SKILLRET: A LARGE-SCALE BENCHMARK FOR SKILL RETRIEVAL IN LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

As LLM agents are increasingly deployed with large libraries of reusable skills, selecting the right skill for a user request has become a critical systems challenge. In small libraries, users may invoke skills explicitly by name, but this assumption breaks down as skill ecosystems grow under tight context and latency budgets. Despite its practical importance, skill retrieval remains underexplored, with limited benchmarks and little understanding of retrieval behavior on realistic skill libraries. To address this gap, we introduce SKILLRET, a large-scale benchmark for skill retrieval in LLM agents. SKILLRET contains 16,129 public agent skills, organized with structured semantic tags and a two-level taxonomy spanning 6 major categories and 18 sub-categories. It provides 63,259 training samples and 4,392 evaluation queries with disjoint skill pools, enabling both benchmarking and retrieval-oriented training. Across a diverse set of retrievers, we find that skill retrieval remains far from solved. Off-the-shelf models struggle on realistic large-scale skill libraries, and prior skill-retrieval models still leave substantial headroom. Task-specific fine-tuning on SKILLRET improves NDCG@10 by +12.9 points over the strongest prior retriever and by +16.2 points over the strongest off-the-shelf retriever. Our analysis further suggests that these gains arise because fine-tuned models better focus on the small skill-relevant signals within long and noisy queries. These results establish SKILLRET as a strong benchmark and foundation for future research on retrieval in large-scale agent systems. We publicly release the benchmark, code, and model checkpoints for embeddings at 0.6B and 8B, and for reranking at 0.6B.

1 INTRODUCTION

LLM agents increasingly rely on reusable skills: long-form procedural modules containing prompts, scripts, workflows, and execution policies (Xu & Yan, 2026; Jiang et al., 2026b; Zhou et al., 2026; Wang et al., 2023a). In small libraries, users can often invoke a skill by name. As skill ecosystems grow, however, exposing every skill in the agent's context or expecting users to identify the right one becomes impractical. An agent instead needs a retrieval layer that selects a small, task-relevant subset under context and latency constraints (Li et al., 2025). This pattern is already visible in systems such as MetaClaw, XSkill, and WebXSkill, which retrieve relevant skills or knowledge at inference time to guide downstream execution (Xia et al., 2026; Jiang et al., 2026a; Wang et al., 2026).

The resulting challenge is to identify the right skills from a large library for a given user request. Despite its practical importance, skill-retrieval evaluation remains underdeveloped. As shown in Table 1, prior skill benchmarks mainly assess end-to-end execution, where outcomes depend on retrieval, skill quality, orchestration, and execution together (Li et al., 2026b; Han et al., 2026; Li et al., 2026a). ToolRet evaluates retrieval of tools rather than long-form skills (Shi et al., 2025). SkillRouter is the closest prior skill-retrieval benchmark, but provides only 75 evaluation queries and does not publicly release its training data (Zheng et al., 2026). These gaps motivate a larger benchmark with public training data and a held-out evaluation set that measures skill retrieval directly.

We introduce SKILLRET, a large-scale benchmark for skill retrieval in LLM agents. Starting from 22,795 public skill listings, a five-stage filtering process retains 17,810 eligible skills. Query construction and evaluation audits yield a final benchmark of 16,129 skills: 10,123 in the training pool and 6,006 in the disjoint evaluation pool. The benchmark provides 63,259 training queries and 4,392 evaluation queries. Generated query–skill pairs undergo leakage filtering, quality review, and

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.

Table 1: Comparison with related benchmarks. *End-to-End* measures agents' final task performance when using skills, whereas *Retrieval* measures whether relevant skills are found. SKILLRET provides public training and evaluation splits. †SkillRouter's 37,979 training samples are not public.

Benchmark	Task	Target	# Eval Samples	Train	# Train Samples
ToolRet (Shi et al., 2025)	Retrieval	Tool	7,615	✓	>200K
SkillsBench (Li et al., 2026b)	End-to-End	Skill	86	×	–
SWE-Skills-Bench (Han et al., 2026)	End-to-End	Skill	565	×	–
AgentSkillOS (Li et al., 2026a)	End-to-End	Skill	30	×	–
SkillRouter (Zheng et al., 2026)	Retrieval	Skill	75	✓	37,979†
SKILLRET (ours)	**Retrieval**	**Skill**	**4,392**	✓	**63,259**

human validation. We also provide semantic tags and a two-level taxonomy with 6 major categories and 18 sub-categories. These resources support controlled training and analysis of retrieval over long skill documents, multi-skill requests, and imbalanced domains.

Evaluating a broad range of retrievers on SKILLRET, we find that task-specific fine-tuning enables smaller models to outperform much larger off-the-shelf models. Our strongest embedding model reaches 86.44 NDCG@10, improving by 12.9 points over the strongest prior skill retriever and by 16.2 points over the strongest off-the-shelf retriever. Reranking adds only a modest gain once first-stage retrieval is strong, while sentence erasure analysis suggests that fine-tuning helps models focus on skill-relevant sentences within long, compositional requests. Finally, on Terminal-Bench (Merrill et al., 2026), retrieved skills preserve agent success while reducing cost by 10%. Together, these results establish SKILLRET as a benchmark for improving retrieval in large-scale agent systems.

2 Related Work

2.1 Agent Skills

Recent work increasingly treats skills as reusable components of LLM agents. MetaClaw jointly evolves an LLM policy and a skill library, synthesizing new skills from failure trajectories (Xia et al., 2026). XSkill studies continual learning in multimodal agents by retrieving and adapting reusable knowledge to the current visual context (Jiang et al., 2026a). WebXSkill combines parameterized action programs with step-level language guidance and organizes executable web skills in a URL-based graph for context-aware retrieval (Wang et al., 2026). Other work examines skill organization and downstream usefulness: AgentSkillOS studies selection and orchestration through capability trees and DAG-based multi-skill pipelines (Li et al., 2026a); SkillsBench finds gains from curated skills but no average benefit from self-generated skills across 86 tasks in 11 domains (Li et al., 2026b); and SWE-Skills-Bench finds that most public software engineering skills provide little or no pass-rate improvement (Han et al., 2026). These studies show both the growing use of skill libraries and the variability of downstream benefits. Because end-to-end outcomes also depend on skill usefulness, orchestration, execution, and contextual fit, they cannot isolate retrieval quality. SKILLRET measures that stage directly.

2.2 Skill Retrieval Benchmarks and Skill Routing

ToolRet evaluates 7.6K retrieval tasks over 43K tools and finds that strong conventional IR models struggle (Shi et al., 2025), establishing retrieval as an agent bottleneck. Tools, however, do not capture the long-form procedural content and compositional structure of skills. Skill documents can contain detailed instructions, decision logic, and usage constraints that must be matched to the relevant part of a user request. SkillFlow proposes a complementary multi-stage pipeline for retrieving and selecting skills from a large community library (Li et al., 2025). The closest prior work, SkillRouter, studies two-stage retrieval over roughly 80K skills and evaluates on 75 expert-verified queries (Zheng et al., 2026). Its finding that full skill bodies contain important routing signals supports evaluation over complete skill documents. SkillRouter focuses on router design, whereas SKILLRET provides public training and evaluation splits with 63,259 training queries and 4,392 evaluation queries over disjoint skill pools. The disjoint pools test whether retrievers generalize to skills unseen during training. This larger evaluation set supports controlled comparisons across retrievers, skill categories, and multi-skill requests.

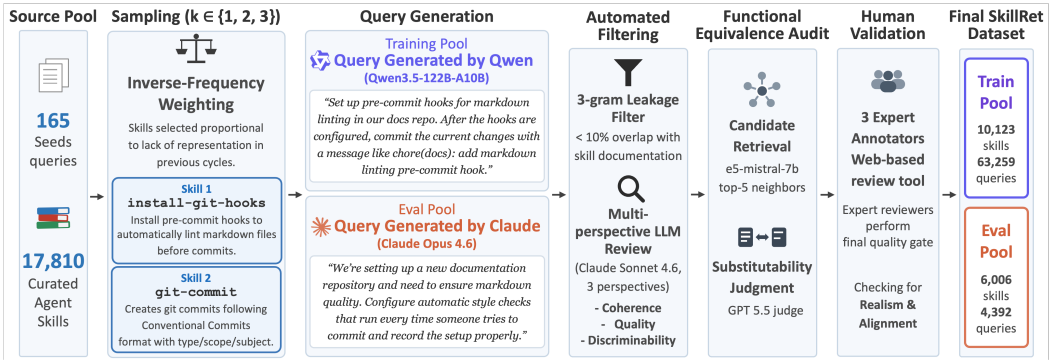

Figure 1: Overview of the SKILLRET data generation pipeline. Using the 17,810 skills retained after the five-stage filtering process and 165 GAIA seed queries, an LLM synthesizes user messages that require the sampled skills. Training queries use Qwen3.5-122B-A10B and evaluation queries use Claude Opus 4.6. Automatic filtering, LLM review, the functional-equivalence and capability-leakage audit, and human validation yield a final evaluation set of 6,006 skills and 4,392 queries alongside 63,259 training queries.

3 SKILLRET BENCHMARK

Fig. 1 illustrates the full data construction pipeline. We curate a corpus of publicly available agent skills as described in Section 3.1. We then generate natural-language queries that each require one or more skills from the library and validate the resulting query–skill pairs as described in Section 3.2.

3.1 DATA COLLECTION AND QUALITY FILTERING

Raw corpus. We start from a snapshot of 22,795 agent skills crawled from `claude-plugins.dev`[1], a community-maintained, open-source marketplace that auto-indexes all public agent skills on GitHub. Each record contains a skill identifier, name, natural-language description, the full skill body (`SKILL.md`), and marketplace metadata including GitHub stars, platform-specific install counts, author, namespace, and license.

Five-stage filtering. We apply a pipeline to remove noise and redundancy. It has two phases. *Content eligibility* (Steps 1–3) ensures that each skill meets basic quality and legal requirements, while *deduplication* (Steps 4–5) removes redundant entries. *(1) Description recovery and pruning.* Listings with missing or stub descriptions (< 10 characters) are recovered via YAML frontmatter parsing or first-paragraph extraction. Unrecoverable entries are removed (3 skills). *(2) Language filtering.* Skills whose bodies contain more than 3% non-Latin characters are removed (1,319 skills), retaining only English-language skills. *(3) License filtering.* Skills declaring a license other than MIT or Apache-2.0 are excluded (255 skills). License-undeclared near-duplicates of these entries are identified using normalized content hashes and also removed (1,249 skills in total). *(4) Content deduplication.* Each skill body is normalized by stripping YAML frontmatter, converting text to lowercase, and removing non-alphanumeric characters, and is then hashed using SHA-256. Among duplicate entries, we retain the entry with the highest star and install counts (1,547 skills removed). *(5) Search-target deduplication.* Skills sharing an identical normalized name–description pair are deduplicated using a hash of the concatenated fields, again retaining the most popular entry (867 skills removed). Both deduplication steps rely on exact hashes and therefore cannot detect skills that implement the same capability with different wording. We therefore apply an LLM-based functional-equivalence audit to the evaluation pool, detailed at the end of Section 3.2.

After the five-stage filtering process, 17,810 skills remain, representing 78.1% of the raw corpus. The per-step attrition is reported in Table 9 in Appendix B.1. Query generation and validation, described in the following subsection, retain only skills that are linked to at least one query, and the functional-equivalence and capability-leakage audit removes further evaluation skills. The final benchmark contains 16,129 skills, comprising a training pool of 10,123 and a held-out evaluation pool of 6,006 paired with 4,392 queries.

[1] https://claude-plugins.dev

3.2 SKILL–QUERY PAIR GENERATION

We construct the evaluation set using a self-instruct-style pipeline (Wang et al., 2023b) that generates natural-language queries requiring one or more skills from the library.

Seed examples. For each generation call, we provide a random subset of the 165 GAIA validation questions (Mialon et al., 2024) as style seeds. These examples illustrate a range of tones, lengths, and request structures. We instruct the model to reflect this diversity rather than follow a fixed template.

Skill sampling. For each generation call we sample $k \in \{1, 2, 3\}$ skills, with k drawn with equal probability. Skills are selected via inverse-frequency weighted sampling (Kang et al., 2019; Cui et al., 2019). Each skill's probability is proportional to $1/(\text{freq} + 1)$, where freq is the number of queries already generated for that skill. A first pass exhausts all skills with zero coverage before any skill is repeated, so every skill in the library is represented at least once.

Query generation. Each generation call receives the name and description of the sampled skills (without the full skill body) and is instructed to produce a single user message that naturally requires all selected skills. The prompt explicitly forbids mentioning any skill name in the generated query, forcing the task need to emerge from the scenario description rather than from lexical overlap with the skill identifier. Previously generated queries for the same skill are shown to the model to suppress near-duplicate outputs. Evaluation queries are generated with Claude Opus 4.6 (Anthropic, 2026), while training queries are generated with Qwen3.5-122B-A10B (Qwen Team, 2026). If the model judges the skill combination to be unrealistic, it may output a designated null token, and that combination is discarded. Full prompt details are provided in Appendix C.1.

Automatic quality filtering. Generated queries pass through a two-stage automatic filter. This procedure is an open rejection-sampling process in which a generator output is not accepted as ground truth merely because it was conditioned on the sampled skills. Failed candidates are discarded and regenerated. *(1) Leakage detection.* We compute the 3-gram overlap between each query and its associated skill documentation. Queries whose overlap ratio exceeds a threshold of 10% are flagged as leaking skill content and discarded. *(2) Multi-perspective LLM review.* A second LLM call evaluates each query from three independent reviewer perspectives: skill coherence (does the query genuinely require the skill?), query quality (is the request specific and realistic?), and benchmark discriminability (would a model without the skill fail to answer it?). A query is rejected if two or more of the three perspectives return an invalid verdict. A single invalid verdict routes the query to human review rather than discarding it outright. Full prompts are provided in Appendix C.2.

Functional-equivalence and capability-leakage audit. Exact hashes cannot detect differently worded skills that implement the same capability. We therefore retrieve the five nearest neighbors of every evaluation skill with e5-mistral-7b-instruct, a model outside the Qwen family used for our retrievers. We retain candidate pairs with cosine similarity of at least 0.80 and ask GPT-5.5 to judge functional substitutability from the complete skill bodies. Within the evaluation pool, union–find groups substitutable pairs into equivalence classes. We retain one representative per class, remove the redundant entries, and remap their qrels so that no query loses a valid answer. We also identify evaluation skills with a functional counterpart in the training pool. Because these cross-split cases have no held-out representative, we remove the affected evaluation skills and drop any query that relies on one rather than shrinking its gold set. This correction converts confirmed equivalent retrievals from false negatives into positives and removes all detected capability-level train–evaluation overlap. Appendix D reports the full audit and graded-relevance robustness checks.

Human expert validation. Queries that pass automatic filtering and the audit are reviewed by three expert annotators using a custom web-based tool that presents each query alongside the associated skill name, description, and the LLM pre-judgment rationale, so that annotators can assess skill–query alignment, realism, and discriminability. Annotators cast a binary valid/invalid mark. This stage is the final quality gate, catching queries that are plausible in isolation but do not genuinely depend on the paired skill.

Training and evaluation splits. To construct query sets for both splits, we generate training queries using Qwen3.5-122B-A10B (Qwen Team, 2026) and evaluation queries using Claude Opus 4.6 (Anthropic, 2026). Using different generator families reduces the opportunity to exploit a single model's

stylistic artifacts. We use the open-weight model for large-scale generation of 63,259 training queries. For evaluation, we use the higher-capacity model to prioritize query quality.

The final split comprises a training pool of 10,123 skills and 63,259 queries, and an evaluation pool of 6,006 skills and 4,392 queries, with no exact skill overlap between the two sets. Fig. 6 in Appendix B.4 shows the distributions before the functional-equivalence audit. Each split differs from the 17,810-skill curated corpus by less than 1 percentage point per major category.

4 Benchmark Analysis

4.1 Taxonomy Overview

We build the taxonomy over the 17,810 skills retained after the five-stage filtering process. Table 2 summarizes its categories. Every skill in the final benchmark retains its major and sub-category label. The taxonomy is constructed through a five-stage pipeline. (1) Tag Discovery: an LLM annotates each skill with three structured tags (*primary_action*, *primary_object*, *domain*), similar to (Gilardi et al., 2023; Ziems et al., 2024). (2) Clustering: k-means over tag vectors at multiple resolutions reveals *stable clusters*, *i.e.,* groups that persist across values of k. (3) Taxonomy Construction: stable clusters seed a draft that experts iteratively refine into 6 Major categories and 18 Sub-categories. (4) LLM-based Assignment: because three-axis tags capture only surface-level attributes, Claude Sonnet 4.6 classifies all 17,810 skills using their names and the first 300 characters of their descriptions. (5) Human Validation: a stratified sample of 200 skills is independently verified by experts, yielding 95.5% accuracy for major categories and 92.2% for sub-categories, with full three-way agreement on 91.0% and 84.5% of items. Appendix B.4 details each stage, and Appendix B.5 reports the tag-rule failures that motivate stage (4). Software Engineering accounts for 62.2% of the curated corpus while Information Retrieval comprises only 3.1%, mirroring the natural composition of public agent skill ecosystems, as shown in Fig. 5 in the Appendix.

4.2 Skill & Taxonomy Statistics

Each skill is represented as the composite text `name | description | skill_md`, which is the actual retrieval target used by all models in our evaluation. The `skill_md` component contains the full Markdown body including instructions, decision logic, usage constraints, and implementation details. Measured in `cl100k_base` tokens, this composite text has a median length of 1,583 tokens, a mean of 2,083, a 95th percentile of 5,531, and a maximum of 47,412. The corpus retained after the five-stage filtering process contains approximately 37.1 M tokens. Fig. 2(a) shows the length distribution. This is an order of magnitude longer than typical tool descriptions in existing benchmarks (Shi et al., 2025), making skill retrieval a fundamentally long-document matching problem. Fig. 2 (b) shows the per-Major length distributions. Data & ML skills are the longest (median 1,795 tokens). Information Retrieval skills are the shortest, reflecting their comparatively concise search-oriented instructions.

Table 2: Taxonomy overview.

Major Category	Sub-Category	Skills	% Total
Software Eng.	Development	4,423	24.8
	Analysis & Testing	2,320	13.0
	Infra. & DevOps	1,970	11.1
	Documentation	889	5.0
	Version Control	756	4.2
	Security	727	4.1
AI Agents	Agent Development	1,194	6.7
	Agent Orchestration	607	3.4
	Agent Evaluation	273	1.5
Business & Planning	Business Analysis	821	4.6
	Project Mgmt.	788	4.4
Data & ML	ML Development	477	2.7
	Data Engineering	418	2.3
	Data Analysis	416	2.3
Content Creation	Writing & Text	687	3.9
	Visual & Media	489	2.7
Info. Retrieval	General Search	357	2.0
	Technical Search	198	1.1

4.3 Query Statistics

We summarize the distributional properties of the generated queries before the functional-equivalence audit. In Fig. 2(c), evaluation queries, generated by Claude Opus 4.6, are substantially longer than training queries generated by Qwen3.5-122B-A10B (Qwen Team, 2026), with a median length of 170 words versus 72 words and a 95th percentile of 270 versus 108 words. This difference likely reflects generation style, as Opus 4.6 tends to produce more detailed, scenario-rich requests that make evaluation queries harder for lexical matching methods. In terms of the number of required skills

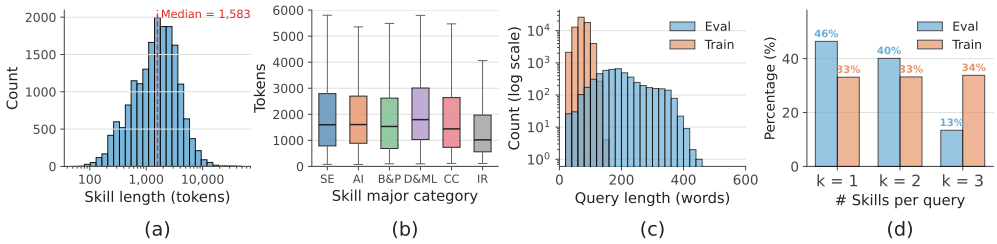

Figure 2: Skill and query length statistics. (a) Distribution of document length across the 17,810 skills retained after the five-stage filtering process. (b) Box plots of document length by major category. (c) Query length distributions for the pre-audit evaluation set and training set. (d) Distribution of the number of skills per query, k, in the same sets. Training queries are sampled uniformly across k, whereas evaluation queries are concentrated on k=1 and k=2.

per query, training queries are distributed uniformly across $k \in \{1, 2, 3\}$, whereas evaluation queries are concentrated on lower values of k, with 46% single-skill queries, 40% two-skill queries, and 13% three-skill queries, as shown in Fig. 2(d). Overall, 54% of evaluation queries require at least two skills. Retrieving just one relevant skill is therefore insufficient for many requests. Section 5.3 evaluates how well models retrieve all required skills on the SKILLRET evaluation set.

5 EVALUATION

5.1 EXPERIMENTAL SETUP

Setup. We use the final set of 6,006 skills and 4,392 queries after the functional-equivalence audit and validation as the default SKILLRET evaluation set. Experiments conducted before the audit are explicitly marked pre-audit. We adopt a two-stage retrieve-then-rerank pipeline where an embedding model retrieves the top-k candidates via cosine similarity and a reranker re-scores each query–candidate pair. Larger k yields better coverage but increases reranking cost. We evaluate $k \in \{10, 20, 50\}$ and set k=20 considering the trade-off between retrieval quality and computational cost. Ablation results are in Appendix E. Encoding the full document text, including the name, description, and Markdown body, consistently outperforms encoding name and description only, as shown in Appendix F. We therefore encode each document up to the model's maximum sequence length, with per-model limits listed in Appendix G. We use each model's officially recommended prompts. Only the Harrier, Qwen3-Embedding, and Qwen3-Reranker families have their default web-search instruction replaced with a skill-retrieval instruction we authored. Full specifications are in Appendix H.

Models. For first-stage retrieval, we evaluate 19 systems across three categories: the sparse BM25 baseline (Robertson & Walker, 1994); encoder-only models from the E5 (Wang et al., 2022), BGE (Xiao et al., 2024), and Snowflake Arctic (Merrick et al., 2024) families; and decoder-only models including F2LLM-v2 (Zhang et al., 2026), pplx-embed-v1 (Eslami et al., 2026), NV-Embed-v1 (Lee et al., 2024), KaLM-Gemma3 (Zhao et al., 2025), Qwen3-Embedding (Zhang et al., 2025), jina-embeddings-v5 (Akram et al., 2026), Harrier-OSS (Huang et al., 2026), and Octen-Embedding (Team, 2025). These systems range from sub-100M to 12B parameters; the full list is in Table 3. For reranking, we evaluate jina-reranker-v2-base-multilingual (Jina AI, 2024) and Qwen3-Reranker (Zhang et al., 2025) at 0.6B and 8B scales. We also evaluate the publicly released SkillRouter-Embedding-0.6B and SkillRouter-Reranker-0.6B checkpoints as external fine-tuned baselines (Zheng et al., 2026). We refer to models fine-tuned on SKILLRET training data collectively as the SKILLRET model family: SkillRet-Embedding-0.6B, SkillRet-Embedding-8B, and SkillRet-Reranker-0.6B. Although Harrier-OSS outperforms Qwen3-Embedding off-the-shelf, we use Qwen3-Embedding as our fine-tuning base because Harrier-OSS is itself a fine-tuned derivative of Qwen3-Embedding. Fine-tuning from either base yields comparable results on the pre-audit evaluation set (Appendix I).

Training details. We fine-tune Qwen3-Embedding-0.6B and Qwen3-Embedding-8B using MultipleNegativesRankingLoss with in-batch negatives on 127,190 positive query–skill pairs derived from the training split. SkillRet-Reranker-0.6B is fine-tuned from Qwen3-Reranker-0.6B using binary cross-entropy on the `yes`/`no` token probability at the final decoding position. Reranker training uses

Table 3: First-stage retrieval results on the SKILLRET evaluation set with 6,006 skills and 4,392 queries. Best result per metric is **bolded**.

Type	Model	Params	NDCG			Recall			Completeness		
			@5	@10	@15	@5	@10	@15	@5	@10	@15
Sparse	BM25	–	49.31	51.69	52.75	53.03	59.41	62.92	38.21	44.56	47.93
Encoder	e5-small-v2	118M	42.63	44.66	45.66	46.03	51.59	54.89	32.61	37.45	40.35
	e5-large-v2	335M	51.30	53.41	54.44	54.88	60.55	63.89	40.16	45.45	48.68
	bge-small-en-v1.5	33M	52.57	54.51	55.45	54.73	60.01	63.07	38.96	43.97	47.11
	snowflake-arctic-embed-s	33M	54.48	56.39	57.35	56.95	61.96	65.02	41.37	46.22	49.32
	bge-large-en-v1.5	335M	57.04	59.00	59.80	59.19	64.37	66.95	42.96	48.34	50.80
Decoder	F2LLM-v2-80M	80M	46.26	48.22	49.27	49.51	54.86	58.32	34.77	39.82	43.06
	pplx-embed-v1-0.6b	0.6B	51.71	54.26	55.52	56.72	63.60	67.70	41.51	48.27	52.32
	NV-Embed-v1	7B	54.62	57.03	57.96	58.16	64.64	67.64	42.12	48.00	50.87
	KaLM-Gemma3-12B	12B	56.31	58.95	60.11	60.48	67.56	71.35	44.49	52.37	56.42
	Qwen3-Embedding-0.6B	0.6B	59.92	61.94	62.98	62.59	67.87	71.25	45.88	51.09	54.67
	jina-embeddings-v5-text-small	0.6B	60.85	62.96	63.90	63.46	68.99	72.02	46.93	53.26	56.74
	Qwen3-Embedding-8B	8B	61.35	63.64	64.73	64.24	70.29	73.76	47.43	54.14	58.15
	harrier-oss-v1-270m	270M	62.42	64.56	65.33	65.12	70.80	73.27	49.00	55.37	58.06
	Octen-Embedding-8B	8B	64.00	65.93	66.85	65.98	71.07	73.99	49.32	55.35	58.99
	harrier-oss-v1-0.6b	0.6B	68.16	70.26	71.09	70.64	76.16	78.76	54.80	61.61	64.82
	SkillRouter-Embedding-0.6B	0.6B	71.80	73.54	74.33	73.73	78.18	80.68	57.22	62.70	65.96
	SkillRet-Embedding-0.6B (ours)	0.6B	79.02	81.12	81.87	82.38	87.74	90.16	70.83	78.94	82.65
	SkillRet-Embedding-8B (ours)	8B	**84.58**	**86.44**	**86.95**	**88.55**	**93.25**	**94.84**	**80.35**	**88.11**	**90.80**

Table 4: Oracle top-20 analysis on the SKILLRET evaluation set. The embedding-only row is the shared first-stage baseline. "+Oracle" inserts missing gold skills before applying the same reranker.

Reranker	Setting	NDCG@5	NDCG@10	NDCG@15
SkillRet-Embedding-8B, no reranking		84.58	86.44	86.95
SkillRouter-Reranker-0.6B (Zheng et al., 2026)	+Rerank	84.69	86.78	87.33
	+Oracle	85.57	88.05	88.92
SkillRet-Reranker-0.6B (ours)	+Rerank	86.10	87.74	88.21
	+Oracle	86.77	88.89	89.63

15 negatives per query, with mining based on four retrievers. All models are trained for one epoch. Negative construction and full hyperparameters are detailed in Appendix J.

Evaluation metrics. We report three primary metrics at k=10. **NDCG@10** (Järvelin & Kekäläinen, 2002) measures ranking quality, **Recall@10** measures the fraction of ground-truth skills retrieved, and **Completeness@10** (Qu et al., 2024) measures the fraction of queries where all ground-truth skills are retrieved, i.e., Recall@10=1. All evaluations are run on a single NVIDIA B200 GPU (180 GB VRAM) per model to ensure reproducibility.

5.2 EXPERIMENTAL RESULTS

Embedding Retrieval. Table 3 reports the retrieval performance of all evaluated models on the SKILLRET benchmark. The best encoder-only model, bge-large-en-v1.5 (Xiao et al., 2024), reaches 59.00 NDCG@10, while the strongest off-the-shelf model, harrier-oss-v1-0.6b (Huang et al., 2026), reaches 70.26. Decoder-only models support substantially longer input sequences than the 512-token limit of the evaluated encoder-only models, allowing them to encode more of each skill document. Model scale alone is insufficient. NV-Embed-v1 (Lee et al., 2024) at 7B scores 57.03, below the 270M Harrier model at 64.56. What matters more is exposure to domain-relevant training data. SkillRouter-Embedding-0.6B (Zheng et al., 2026) reaches 73.54, while SkillRet-Embedding-0.6B and 8B reach 81.12 and 86.44. Thus, the 8B model leads the strongest prior retriever by 12.9 points and the strongest off-the-shelf retriever by 16.2 points.

Reranking. Table 4 separates first-stage recall from reranker quality. In the +Rerank setting, the reranker rescores the 20 candidates returned by SkillRet-Embedding-8B, whose embedding-only scores are shown in the first row. In *+Oracle*, every gold skill missing from that pool is first inserted by displacing an equal number of lowest-ranked non-gold candidates, holding the pool at 20, and the

same reranker then rescores it. Starting from SkillRet-Embedding-8B at 86.44 NDCG@10, SkillRet-Reranker reaches 87.74. The oracle pool raises this only to 88.89. SkillRouter-Reranker (Zheng et al., 2026) reaches 86.78 on the same candidate pool. These small oracle gains suggest limited benefit from expanding candidate coverage under the rerankers tested.

5.3 Analysis

Multi-skill queries amplify the fine-tuning gain. Figure 3 compares retrieval performance by the number k of required skills. Qwen3-Embedding-0.6B degrades sharply across both metrics as k grows: its Completeness@10 falls from 83.8% at k=1 to 25.8% at k=2 and 1.8% at k=3. SkillRet-Embedding-0.6B achieves 92.1%, 74.2%, and 42.1% on the same groups, respectively. Its NDCG@10 advantage over Qwen3 widens from 10.2 points for single-skill queries to 33.1 points for three-skill queries. Appendix N reports the breakdown by major category.

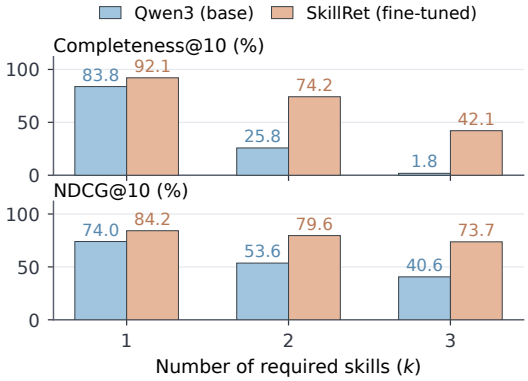

Figure 3: Retrieval performance by required-skill count.

End-to-end agent utility. We integrate SkillRet-Embedding-0.6B into a Claude Code agent with Claude Opus 4.6 and evaluate 80 Terminal-Bench tasks (Merrill et al., 2026) over five independent trials. A `UserPromptSubmit` hook retrieves from the pre-audit skill library and injects the top three skill bodies when cosine similarity is at least 0.30. As Table 6 shows, retrieval matches the no-retrieval baseline within 0.3 percentage points while reducing mean cost per trial by approximately 10%. Because Pass@1 measures task completion rather than agreement with query–skill labels, this experiment provides a downstream check independent of the benchmark's relevance labels.

Deployment efficiency. We measure retrieval latency on a single NVIDIA B200. The Qwen3 backbones underlying SkillRet-Embedding-0.6B and SkillRet-Embedding-8B take 21.25 and 26.25 ms per query at batch size 1, respectively. Their batched throughputs are 381 and 129 queries/s (Table 5). Fine-tuning does not change the architecture, so these measurements estimate the runtime of the corresponding SkillRet checkpoints. Reranking one query's top 20 candidates takes 831.4 ms with Qwen3-Reranker-0.6B and 1,875.5 ms with Qwen3-Reranker-8B (Table 21 in Appendix O). SkillRet-Embedding-8B alone reaches 86.44 NDCG@10; SkillRet-Reranker-0.6B raises this to 87.74, a gain of 1.30 points. Latency-sensitive deployments can therefore use first-stage retrieval alone. Appendix O details the measurement protocol and throughput batch sizes.

Table 5: First-stage latency (ms) and batched throughput (queries/s).

Model	ms	q/s
BM25 (CPU)	0.73	1,405
bge-small-en-v1.5	6.57	624
bge-large-en-v1.5	13.57	142
SkillRet-Emb.-0.6B	21.25	381
SkillRet-Emb.-8B	26.25	129
KaLM-Gemma3-12B	60.58	43

Training improves sensitivity to skill-relevant sentences. We hypothesize that fine-tuning sharpens sensitivity to the few sentences that signal skill intent within long requests, rather than to background context, output requirements, and other constraints. We test this hypothesis through sentence erasure, rather than interpreting attention weights directly. To test this, we conduct a sentence erasure analysis (Barkan et al., 2024; Li et al., 2016) on the 2,319 single-skill queries from the pre-audit evaluation split, comparing Qwen3-Embedding-0.6B with SkillRet-Embedding-0.6B. For each sentence s_i in query q, we replace it with `[MASK]`, re-encode the masked query $q_{\setminus s_i}$, and compute $\text{importance}(s_i) = \text{sim}(q, d^+) - \text{sim}(q_{\setminus s_i}, d^+)$. We then mask the top-$k$ most important sentences and re-run retrieval, with results shown in Table 7. On the full query, the trained model outperforms the base model by 7.5 NDCG@10 points, yet removing the single most important sentence causes a larger performance drop. This supports our hypothesis that the trained model concentrates its retrieval signal on a small set of skill-relevant sentences, whereas the base model relies more diffusely on the entire query. A qualitative visualization of this pattern is shown in Fig. 7 in Appendix M.

Table 6: End-to-end evaluation on 80 Terminal-Bench tasks over five trials with Claude Code and Claude Opus 4.6, using the pre-audit skill library, reporting Pass@1 and per-trial cost.

Skill retrieval	Pass@1	Cost/trial
None (baseline)	65.5%	$0.86
SkillRet (ours)	65.8%	$0.78
vs. baseline	*+0.3 pp*	*9.3% lower*



Figure 4: MTEB Retrieval score vs. SKILLRET. Circle size is proportional to parameter count.

Table 7: NDCG@10 after masking important query snippets on 2,319 pre-audit single-skill queries. Base and Trained denote Qwen3-Embedding-0.6B and SkillRet-Embedding-0.6B.

Condition	Base	Trained
Full query	73.41	80.89
mask_top_1	56.34 (-23.3%)	57.29 (-29.2%)
mask_top_2	40.39 (-45.0%)	38.92 (-51.9%)
mask_top_3	26.66 (-63.7%)	24.00 (-70.3%)

Table 8: Per-major-category NDCG@10 on the SKILLRET evaluation set for Qwen3-Embedding (Base) and SkillRet-Embedding (Ours), hardest first. A query is counted in every major category among its gold skills.

	0.6B		8B		Δ	
Major	Base	Ours	Base	Ours	0.6B	8B
Info. Retr.	44.1	72.8	42.8	79.0	+28.7	+36.2
AI Agents	46.6	73.0	47.5	81.2	+26.4	+33.7
Bus. & Plan.	54.2	79.7	55.2	84.9	+25.5	+29.7
Content Cr.	55.6	82.7	56.6	87.7	+27.1	+31.1
Software Eng.	62.1	81.2	64.7	86.9	+19.1	+22.2
Data & ML	67.9	85.7	72.0	90.4	+17.8	+18.4
Overall	61.9	81.1	63.6	86.4	+19.2	+22.8

MTEB Retrieval rankings do not consistently transfer to skill retrieval. Figure 4 plots MTEB Retrieval score (Hugging Face, 2026) against SKILLRET NDCG@10. MTEB Retrieval score shows a moderate positive correlation at Spearman $\rho = 0.70$, yet ranking inversions are common, with models that score highly on MTEB often underperforming on SKILLRET. For example, KaLM-Gemma3-12B leads on MTEB with 75.66 but scores 58.95 on SKILLRET. In contrast, harrier-oss-v1-0.6b ranks fourth on MTEB at 70.75 and achieves the best off-the-shelf SKILLRET score of 70.26. These inversions suggest that skill retrieval demands a form of query understanding distinct from general semantic matching, requiring models to identify specific capability signals within long, multi-sentence queries. Task-specific fine-tuning, as demonstrated by the SkillRet model family, is the most effective way to bridge this gap. Full scores are in Appendix K.

Per-category performance. Table 8 breaks down NDCG@10 on the SKILLRET evaluation set by the six major categories. The SkillRet models improve over the Qwen3-Embedding baselines in every category, with gains ranging from 18.4 to 36.2 points for the 8B variant. Information Retrieval and AI Agents consistently score lowest, and an 11.4 point gap between the easiest and hardest categories persists even for SkillRet-Embedding-8B. This category-level disparity is invisible to the aggregate NDCG@10 of 86.4 and can only be surfaced through taxonomy-based stratification. Finer-grained sub-category results in Appendix L expose within-major variance of up to 11.5 points.

6 CONCLUSION AND LIMITATIONS

SKILLRET provides public training and evaluation splits and a two-level taxonomy for studying long-document skill retrieval. SkillRet-Embedding-8B reaches 86.44 NDCG@10, compared with 70.26 for the strongest off-the-shelf retriever and 73.54 for the strongest prior skill retriever. Because its queries are synthetic, they may under-represent terse, conversational, or context-dependent requests despite leakage filtering and validation. Moreover, retrieval performance does not guarantee successful skill use: the Terminal-Bench study offers limited downstream evidence, and retrieved skills must still be interpreted and executed correctly. Evaluation on real agent traffic is needed to connect retrieval gains with downstream task success.

AI USE STATEMENT

Generative AI is a component of the research pipeline described in this paper, and we disclose its use as follows. *Synthetic data generation.* All benchmark queries are generated by large language models: 63,259 training queries with Qwen3.5-122B-A10B and 4,392 evaluation queries with Claude Opus 4.6, conditioned on sampled skills and GAIA-style seed examples (Section 3.2, with the complete prompts in Appendix C.1). *Qualitative and thematic data analysis.* Claude Sonnet 4.6 performs the two-pass structured tagging of the corpus (Appendix B.2), assigns all 17,810 skills retained after the five-stage filtering process to the taxonomy (Appendix B.5), and carries out the three-perspective review of generated queries (Appendix C.2). GPT-5.5 judges functional substitutability in the functional-equivalence and capability-leakage audit (Appendix D). *Evaluated systems.* Claude Opus 4.6 serves as the agent backbone in the end-to-end Terminal-Bench evaluation (Section 5). We did not use generative AI to develop theoretical models or conceptual frameworks, to formulate or prove mathematical claims, or to translate material; these categories do not apply to this work. Beyond the pipeline above, we used AI assistants to edit software code and to improve the readability of the manuscript. All AI-produced artifacts were verified before use. Generated queries passed 3-gram leakage filtering and the three-perspective review, and were then adjudicated valid or invalid by the three authors; taxonomy assignments were checked against a stratified sample of 200 skills, yielding 95.5% major-category and 92.2% sub-category accuracy (Appendix B.6). We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The benchmark construction pipeline is described in Section 3, with per-step filtering attrition in Appendix B.1, the structured tagging and taxonomy procedure in Appendices B.2, B.4 and B.5, the complete query generation and review prompts in Appendices C.1 and C.2, and the functional-equivalence and capability-leakage audit in Appendix D. The evaluation protocol, metrics, and hardware are given in Section 5.1; per-model maximum sequence lengths, retrieval prompts, reranking depth, and document representation choices are reported in Appendices G, H, E and F; and the fine-tuning base model choice and complete hyperparameters are in Appendices I and J. All models are evaluated on a single NVIDIA B200 GPU. The benchmark, the disjoint training and evaluation splits, the code, and the fine-tuned checkpoints are publicly released (Appendix A).

ETHICS STATEMENT

Data provenance and licensing. SKILLRET is built from publicly available agent skills indexed by `claude-plugins.dev` from public GitHub repositories. We retain only skills that declare an MIT or Apache-2.0 license and remove all others together with their license-undeclared near-duplicates (Section 3.1). Queries are synthetic and were not collected from users or from live agent traffic. *Personal data.* Marketplace records include author and namespace fields. The dataset is released for retrieval evaluation and model development, and is not intended for profiling individual skill authors or inferring personal attributes. Human validation was performed by the three authors of this paper rather than by recruited annotators, on public technical documents and synthetic queries, and no personal or sensitive data was collected. *Representativeness and bias.* The corpus mirrors the natural composition of public skill ecosystems and is therefore imbalanced, with Software Engineering accounting for 62.2% of skills and Information Retrieval for 3.1%. The taxonomy makes this imbalance explicit and supports per-category analysis. The corpus and the queries are English-only, and query style reflects the two generator model families, which we report in Section 4. *Intended use and misuse.* Strong retrieval performance on SKILLRET does not imply safe or correct end-to-end agent behavior. Retrieved skills may be outdated, unsafe, misapplied, or incorrectly composed with other skills, and SKILLRET should not be used as evidence that a deployed agent system is safe or reliable. Practical deployments should add provenance checks, permission controls, sandboxing, and human oversight for high-impact actions. Appendix P discusses broader impacts in more detail. We have read and adhere to the ICLR Code of Ethics.

REFERENCES

Mohammad Kalim Akram, Saba Sturua, Nastia Havriushenko, Quentin Herreros, Michael Günther, Maximilian Werk, and Han Xiao. jina-embeddings-v5-text: Task-targeted embedding distillation. *arXiv preprint arXiv:2602.15547*, 2026.

Anthropic. Claude opus 4.6. https://www.anthropic.com/claude, 2026.

Oren Barkan, Yonatan Toib, Yehonatan Elisha, Jonathan Weill, and Noam Koenigstein. Llm explainability via attributive masking learning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 9522–9537, 2024.

Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9268–9277, 2019.

Sedigheh Eslami, Maksim Gaiduk, Markus Krimmel, Louis Milliken, Bo Wang, and Denis Bykov. Diffusion-pretrained dense and contextual embeddings. *arXiv preprint arXiv:2602.11151*, 2026.

Fabrizio Gilardi, Meysam Alizadeh, and Maël Kubli. Chatgpt outperforms crowd workers for text-annotation tasks. *Proceedings of the National Academy of Sciences*, 120(30):e2305016120, 2023.

Tingxu Han, Yi Zhang, Wei Song, Chunrong Fang, Zhenyu Chen, Youcheng Sun, and Lijie Hu. Swe-skills-bench: Do agent skills actually help in real-world software engineering? *arXiv preprint arXiv:2603.15401*, 2026. doi: 10.48550/arXiv.2603.15401.

Aolong Huang, Liang Wang, Furu Wei, et al. Harrier-oss-v1. https://huggingface.co/microsoft/harrier-oss-v1-0.6b, 2026.

Hugging Face. MTEB leaderboard. https://huggingface.co/spaces/mteb/leaderboard, 2026.

Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

Guanyu Jiang, Zhaochen Su, Xiaoye Qu, and Yi R. Fung. Xskill: Continual learning from experience and skills in multimodal agents. *arXiv preprint arXiv:2603.12056*, 2026a. doi: 10.48550/arXiv.2603.12056.

Yanna Jiang, Delong Li, Haiyu Deng, Baihe Ma, Xu Wang, Qin Wang, and Guangsheng Yu. Sok: Agentic skills–beyond tool use in llm agents. *arXiv preprint arXiv:2602.20867*, 2026b.

Jina AI. jina-reranker-v2-base-multilingual. https://huggingface.co/jinaai/jina-reranker-v2-base-multilingual, 2024.

Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. *arXiv preprint arXiv:1910.09217*, 2019.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Nv-embed: Improved techniques for training llms as generalist embedding models. *arXiv preprint arXiv:2405.17428*, 2024.

Fangzhou Li, Pagkratios Tagkopoulos, and Ilias Tagkopoulos. Skillflow: Scalable and efficient agent skill retrieval system. *arXiv e-prints*, pp. arXiv–2504, 2025.

Hao Li, Chunjiang Mu, Jianhao Chen, Siyue Ren, Zhiyao Cui, Yiqun Zhang, Lei Bai, and Shuyue Hu. Organizing, orchestrating, and benchmarking agent skills at ecosystem scale. *arXiv preprint arXiv:2603.02176*, 2026a. doi: 10.48550/arXiv.2603.02176.

Jiwei Li, Will Monroe, and Dan Jurafsky. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*, 2016.

Xiangyi Li, Wenbo Chen, Yimin Liu, Shenghan Zheng, Xiaokun Chen, Yifeng He, Yubo Li, Bingran You, Haotian Shen, Jiankai Sun, Shuyi Wang, Binxu Li, Qunhong Zeng, Di Wang, Xuandong Zhao, Yuanli Wang, Roey Ben Chaim, Zonglin Di, Yipeng Gao, Junwei He, Yizhuo He, Liqiang Jing, Luyang Kong, Xin Lan, Jiachen Li, Songlin Li, Yijiang Li, Yueqian Lin, Xinyi Liu, Xuanqing Liu, Haoran Lyu, Ze Ma, Bowei Wang, Runhui Wang, Tianyu Wang, Wengao Ye, Yue Zhang, Hanwen Xing, Yiqi Xue, Steven Dillmann, and Han-chung Lee. Skillsbench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*, 2026b. doi: 10.48550/arXiv.2602.12670.

Luke Merrick, Danmei Xu, Gaurav Nuti, and Daniel Campos. Arctic-embed: Scalable, efficient, and accurate text embedding models. *arXiv preprint arXiv:2405.05374*, 2024.

Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. *arXiv preprint arXiv:2601.11868*, 2026.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: a benchmark for general AI assistants. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=fibxvahvs3.

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, 2024.

Qwen Team. Qwen3.5. https://qwen.ai/blog?id=qwen3.5, 2026. Accessed: 2026-04-22.

Stephen E Robertson and Steve Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR'94: Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, organised by Dublin City University*, pp. 232–241. Springer, 1994.

Zhengliang Shi, Yuhan Wang, Lingyong Yan, Pengjie Ren, Shuaiqiang Wang, Dawei Yin, and Zhaochun Ren. Retrieval models aren't tool-savvy: Benchmarking tool retrieval for large language models. *arXiv preprint arXiv:2503.01763*, 2025. doi: 10.48550/arXiv.2503.01763.

Octen Team. Octen series: Optimizing embedding models to #1 on rteb leaderboard, 2025. URL https://octen-team.github.io/octen_blog/posts/octen-rteb-first-place/.

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*, 2023a.

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

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 13484–13508, 2023b.

Zhaoyang Wang, Qianhui Wu, Xuchao Zhang, Chaoyun Zhang, Wenlin Yao, Fazle Elahi Faisal, Baolin Peng, Si Qin, Suman Nath, Qingwei Lin, Chetan Bansal, Dongmei Zhang, Saravan Rajmohan, Jianfeng Gao, and Huaxiu Yao. Webxskill: Skill learning for autonomous web agents. *arXiv preprint arXiv:2604.13318*, 2026. doi: 10.48550/arXiv.2604.13318.

Peng Xia, Jianwen Chen, Xinyu Yang, Haoqin Tu, Jiaqi Liu, Kaiwen Xiong, Siwei Han, Shi Qiu, Haonian Ji, Yuyin Zhou, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. Metaclaw: Just talk – an agent that meta-learns and evolves in the wild. *arXiv preprint arXiv:2603.17187*, 2026. doi: 10.48550/arXiv.2603.17187.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pp. 641–649, 2024.

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

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

Ziyin Zhang, Zihan Liao, Hang Yu, Peng Di, and Rui Wang. F2llm-v2: Inclusive, performant, and efficient embeddings for a multilingual world. *arXiv preprint arXiv:2603.19223*, 2026.

Xinping Zhao, Xinshuo Hu, Zifei Shan, Shouzheng Huang, Yao Zhou, Xin Zhang, Zetian Sun, Zhenyu Liu, Dongfang Li, Xinyuan Wei, et al. Kalm-embedding-v2: Superior training techniques and data inspire a versatile embedding model. *arXiv preprint arXiv:2506.20923*, 2025.

YanZhao Zheng, ZhenTao Zhang, Chao Ma, YuanQiang Yu, JiHuan Zhu, Baohua Dong, and Hangcheng Zhu. Skillrouter: Skill routing for llm agents at scale. *arXiv preprint arXiv:2603.22455*, 2026. doi: 10.48550/arXiv.2603.22455.

Huichi Zhou, Siyuan Guo, Anjie Liu, Zhongwei Yu, Ziqin Gong, Bowen Zhao, Zhixun Chen, Menglong Zhang, Yihang Chen, Jinsong Li, et al. Memento-skills: Let agents design agents. *arXiv preprint arXiv:2603.18743*, 2026.

Caleb Ziems, William Held, Omar Shaikh, Jiaao Chen, Zhehao Zhang, and Diyi Yang. Can large language models transform computational social science? *Computational Linguistics*, 50(1): 237–291, 2024.

APPENDIX

A DATA, CODE, AND MODELS.

We release SKILLRET publicly at https://huggingface.co/datasets/anonymous-ed-benchmark/SKILLRET. Code is available at https://anonymous.4open.science/r/SKILLRET-608D. Fine-tuned model weights are also publicly released: SkillRet-Embedding-0.6B at https://huggingface.co/anonymous-ed-benchmark/SKILLRET-Embedding-0.6B, SkillRet-Embedding-8B at https://huggingface.co/anonymous-ed-benchmark/SKILLRET-Embedding-8B, and SkillRet-Reranker-0.6B at https://huggingface.co/anonymous-ed-benchmark/SKILLRET-Reranker-0.6B. The dataset is derived from public GitHub-hosted skills and synthetic queries. It is intended for skill retrieval evaluation and model development, not for profiling individual authors, inferring personal attributes, or certifying end-to-end agent safety.

B DATASET CONSTRUCTION DETAILS

This appendix provides details of skill collection, filtering, and taxonomy construction. It expands on Sections 3 and 4. The following subsections cover the per-step filtering attrition (§B.1), the two-pass LLM tagging procedure (§B.2), consensus clustering over action–object combinations (§B.3), the iterative taxonomy design process (§B.4), LLM-based skill assignment (§B.5), and human validation (§B.6).

B.1 PER-STEP FILTERING ATTRITION

Table 9 reports the per-step attrition of the five-stage filtering pipeline described in §3.1. The two largest reductions come from content deduplication in Step 4 and language filtering in Step 2.

Table 9: Five-stage skill filtering pipeline, with 17,810 skills remaining after filtering. The largest reductions come from content deduplication in Step 4 and language filtering in Step 2.

Step	Filter	Removed	Remaining
–	Raw corpus	–	22,795
Phase 1: Content eligibility			
1	Description recovery / pruning	3	22,792
2	Language filter ($>3\%$ non-Latin)	1,319	21,473
3	License filter + propagated cleanup	1,249	20,224
Phase 2: Deduplication			
4	Content deduplication (normalized hash)	1,547	18,677
5	Search-target deduplication (name+desc)	867	**17,810**

B.2 STRUCTURED TAGGING

To characterize each skill along interpretable dimensions, we assign three structured tags per skill: **primary_action** (what the skill *does*), **primary_object** (what the skill *acts on*), and **domain** (the technical field it belongs to). We use a two-pass procedure with Claude Sonnet 4.6.

B.2.1 PASS 1: CATEGORY DISCOVERY

For all 17,810 skills retained after the five-stage filtering process, names and descriptions truncated to 100 characters are submitted in a single prompt. The model is instructed to discover natural, non-overlapping categories for each dimension at an appropriate granularity (roughly 8–15 categories). This yields 13 actions, 14 objects, and 13 domains. The categories are discovered by the LLM from the corpus rather than being predefined by the authors, though the target granularity (8–15 per dimension) is specified in the prompt. The resulting label sets were manually reviewed by the authors to verify semantic coherence and adjust ambiguous or overlapping categories.

System prompt.

```
You are a skill taxonomy analyst. You will receive a list
of ~17,000 AI coding skill names with short descriptions.

Your task: analyze ALL skills and discover the natural
categories that exist across three dimensions.

For each dimension, identify **distinct, non-overlapping
categories** at an appropriate granularity level (roughly
8-15 categories per dimension). Each category should have
a short lowercase label (1-2 words, snake_case) and a
brief description.

Dimensions:
1. **primary_action**: What the skill DOES
   (the core verb/activity)
2. **primary_object**: What the skill acts ON
   (the target/subject)
3. **domain**: What technical field the skill belongs to

Output strict JSON with this structure:
{
  "primary_action": [
    {"label": "...", "description": "..."}
  ],
  "primary_object": [
    {"label": "...", "description": "..."}
  ],
  "domain": [
    {"label": "...", "description": "..."}
  ]
}

No markdown fences, no explanations outside the JSON.
```

User message format.

```
Here are all the skills:

{skill_name_1}: {description_first_100_chars}
{skill_name_2}: {description_first_100_chars}
...
{skill_name_17810}: {description_first_100_chars}
```

B.2.2 PASS 2: BATCH CLASSIFICATION

The discovered categories are injected into the system prompt as a closed label set. Skills are then classified in batches of 100, with the model selecting exactly one label per dimension for each skill. The output is a structured JSON record per skill. After deduplication of any double-tagged entries, we obtain a clean set of 17,810 (id, action, object, domain) tuples.

System prompt.

```
You are a skill taxonomy classifier. For each AI coding
skill, assign exactly 3 labels.

**primary_action** -- choose ONE from:
  - implement: Writing, building, or creating new code,
    features, components, or systems
  - debug: Finding, diagnosing, and fixing bugs, errors,
    or unexpected behavior
  - review: Evaluating, auditing, or assessing code,
    documentation, or designs for quality
```

```
  - test: Writing, running, or managing automated tests
    and test strategies
  - design: Architecting systems, designing APIs,
    planning schemas, or defining specifications
  - document: Creating, updating, or generating
    documentation, comments, or explanations
  - refactor: Restructuring or improving existing code
    without changing behavior
  - configure: Setting up, installing, or configuring
    tools, environments, or services
  - deploy: Building, packaging, releasing, or deploying
    software to environments
  - analyze: Investigating, researching, profiling, or
    extracting insights from code or data
  - generate: Producing artifacts like images, content,
    reports, or boilerplate automatically
  - orchestrate: Coordinating, routing, or managing
    multiple agents, tasks, or workflows
  - search: Finding, discovering, or retrieving
    information from code, docs, or the web

**primary_object** -- choose ONE from:
  - code: Source code files, functions, classes, modules
  - api: REST, GraphQL, gRPC interfaces and endpoints
  - database: Database schemas, queries, migrations
  - ui_component: Frontend components, pages, layouts
  - test_suite: Unit tests, integration tests, E2E tests
  - documentation: READMEs, API docs, guides, changelogs
  - pipeline: CI/CD pipelines, data pipelines, build
    workflows
  - infrastructure: Cloud resources, containers, K8s,
    and infrastructure-as-code
  - agent_skill: AI agent skills, prompts, system
    prompts, and LLM configurations
  - data: Datasets, data files, spreadsheets, reports
  - project: Entire projects, repositories, codebases
  - dependency: Packages, libraries, version management
  - security: Vulnerabilities, authentication, secrets
  - content: Text content, blog posts, marketing copy

**domain** -- choose ONE from:
  - web_frontend: Browser-based UI development (React,
    Vue, Angular, HTML/CSS)
  - backend_api: Server-side development, REST/GraphQL
    APIs, microservices
  - devops_infra: CI/CD, cloud infrastructure,
    containers, Kubernetes
  - data_ml: Data engineering, machine learning, AI
    model training, analytics
  - mobile: iOS, Android, cross-platform mobile apps
  - security: Application security, penetration testing,
    vulnerability management
  - database: Relational and NoSQL databases, query
    optimization
  - ai_agents: LLM applications, agent frameworks, RAG
    systems, prompt engineering
  - developer_tools: CLI tools, IDE extensions, code
    generation, developer productivity
  - testing_qa: Test automation, quality assurance
  - product_design: UI/UX design, product management,
    user research
  - systems: Operating systems, embedded systems,
    compilers
  - business_ops: Project management, marketing, sales,
    finance, legal
```

Table 10: Consensus clustering with 10 stable groups at a threshold of 5/5. *Binding* indicates whether members share a common object or action. 19 singletons (5,776 skills) are omitted for brevity.

Group	Members (action × object)	Skills	Bind
G1	implement × code, impl. × api, impl. × data	2,404	action
G2	document × doc, generate × doc, review × doc	1,309	object
G3	orchestrate × agent, impl. × agent, config. × agent	1,499	object
G4	configure × infra, deploy × infra	637	object
G5	analyze × code, analyze × project	627	action
G6	configure × pipeline, impl. × pipeline	475	object
G7	orchestrate × pipeline, deploy × pipeline	414	object
G8	design × project, design × api, design × code	441	action
G9	design × agent, analyze × agent, doc. × agent	444	object
G10	configure × code, configure × project	298	action

```
Respond ONLY with a JSON array. Each element:
{"id": "...", "primary_action": "...",
 "primary_object": "...", "domain": "..."}.
No explanations, no markdown fences.
```

User message format (per batch of 100 skills).

```
Tag these skills:
{id_1}|{name_1}: {description_first_200_chars}
{id_2}|{name_2}: {description_first_200_chars}
...
{id_100}|{name_100}: {description_first_200_chars}
```

B.3 Consensus Clustering over Action–Object Combinations

The action × object product space contains 182 possible combinations, but the distribution is highly concentrated. The top 44 combinations account for 80% of the corpus retained after the five-stage filtering process, covering 14,324 of 17,810 skills. We focus on these 44 combinations to discover stable groupings that seed the initial taxonomy draft.

Embedding. Each combination is represented as the text "`{action} {object}`" and encoded with Qwen3-Embedding-8B, yielding a 4,096-dimensional vector.

Multi-resolution clustering. We run k-means at five resolutions ($k \in \{5, 7, 10, 15, 20\}$) with 20 random initializations each, and build a co-association matrix. Each entry (i, j) records the fraction of runs in which combinations i and j are assigned to the same cluster.

Strict consensus groups. Two combinations are linked if and only if they co-occur in *all five* resolutions at a threshold of 5/5. The pair is always assigned to the same cluster regardless of whether k is 5 or 20. Connected components of this graph yield **10 stable groups** (25 combinations) and **19 singletons** (Table 10).

In *object-bound* groups, diverse actions share a common object. For example, G2 combines document × doc, generate × doc, and review × doc. In *action-bound* groups, one action spans multiple objects. G1 combines implement × code, implement × api, and implement × data. There are six object-bound groups and four action-bound groups. These stable groups seed the initial taxonomy draft in §B.4. The ten groups in Table 10 contain 8,548 skills. Together with 5,776 skills in the 19 omitted singleton combinations, they account for all 14,324 skills in the 44 selected combinations. The remaining 3,486 skills in the curated corpus lie outside these combinations and are not included in this clustering analysis.

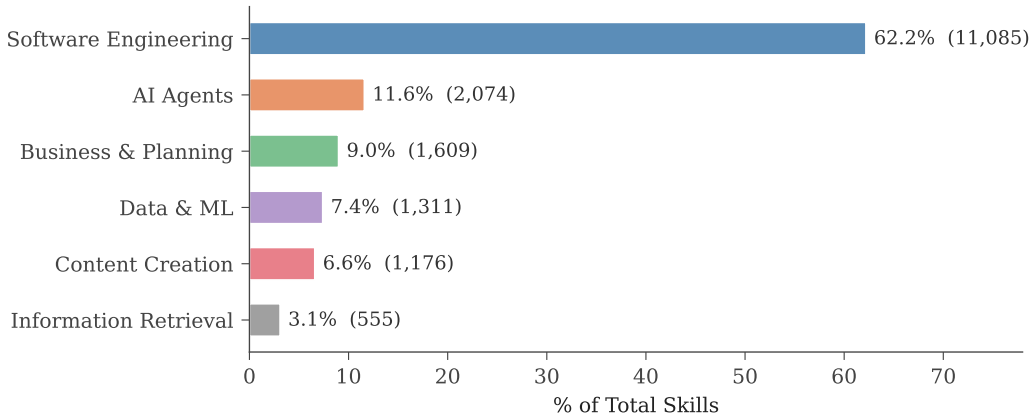

Figure 5: Major-category distribution of the 17,810 skills retained after the five-stage filtering process. Software Engineering dominates (62.2%), creating a 20× imbalance with the smallest category (Information Retrieval, 3.1%).

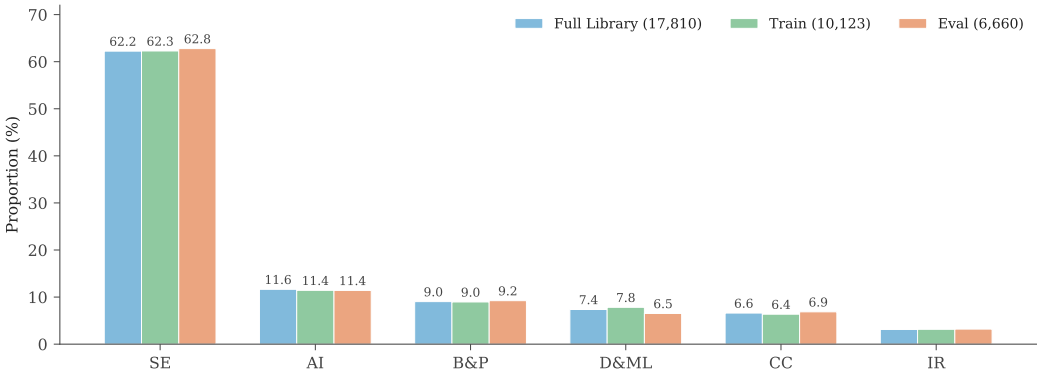

Figure 6: Major-category distribution across the pre-audit data splits. The bars represent the curated corpus of 17,810 skills, the training pool of 10,123 skills, and the pre-audit evaluation pool of 6,660 skills. They show near-identical proportions (< 1 pp deviation), showing that the initial disjoint split preserves the natural category distribution before audit corrections.

B.4 Iterative Taxonomy Design

The final two-level taxonomy shown in Table 2 is the product of an iterative, human-in-the-loop process. The 10 stable groups identified by consensus clustering (§B.3) were used to seed an initial draft taxonomy with 7 Major categories and 21 Sub-categories. Experts then iteratively reviewed stratified samples of 200 skills, identifying structural ambiguities such as an over-broad *Documentation & Knowledge* category, mixed classification axes within Software Engineering, and scattered ML-related skills across Data and SE. Through successive rounds of review and revision, the taxonomy was refined into the final **6 Major / 18 Sub** structure.

B.5 LLM-based Skill Assignment

While tag-based heuristic rules were effective for *discovering* the taxonomy structure, we found them insufficient for precise *assignment* of individual skills. Three-axis tags (action, object, domain) capture only surface-level attributes and cannot distinguish skills whose true purpose is apparent only from the name and description. For example, a skill tagged `implement / content / business_ops` is routed to Software Engineering by tag-based rules, although its description reveals a marketing-campaign planner that belongs in Business & Planning.

To address these limitations, we classify all 17,810 skills retained after the five-stage filtering process, using Claude Sonnet 4.6 via the Anthropic API.

System prompt.

```
You are a taxonomy classifier for AI agent skills.
Each skill is a reusable instruction file that extends
an LLM's capabilities. Given a skill's name and
description, assign it to exactly one (Major,
Sub-category) pair from the taxonomy below.

TAXONOMY:
## Software Engineering
   - Development / Analysis & Testing
   - Infrastructure & DevOps / Security
   - Version Control / Documentation
## AI Agents
   - Agent Development / Orchestration / Evaluation
## Data & ML
   - Data Engineering / Data Analysis / ML Development
## Content Creation
   - Writing & Text / Visual & Media
## Business & Planning
   - Business Analysis / Project Management
## Information Retrieval
   - Technical Search / General Search

CLASSIFICATION PRINCIPLE:
- Classify by the DOMAIN in which the skill's
  capability is used.
- Every skill extends an agent's capabilities,
  but classify by WHAT the extended capability
  is about, not the fact that an agent uses it.
- Technical docs (README, API docs) -> SE / Docs.
- Product planning (PRD, sprints, Jira)
  -> Business & Planning / Project Management.
- Pure business analysis (market research)
  -> Business & Planning / Business Analysis.
- Text/media as final product -> Content Creation.
- AI Agents is ONLY for the agent system itself
  (prompts, routing, MCP servers, evaluation).
- Information Retrieval is ONLY when the PRIMARY
  output is found/retrieved content.

OUTPUT: a JSON array, one object per skill.
  {"id": "...", "major": "...", "sub": "..."}
No markdown fences. No explanations.
```

User message format (per batch of 50 skills).

```
Classify these skills:

{id_1}|{name_1}: {description_first_300_chars}
{id_2}|{name_2}: {description_first_300_chars}
...
{id_50}|{name_50}: {description_first_300_chars}
```

B.6 Human Validation of Assignment

To verify the quality of LLM-based assignment, a stratified random sample of 200 skills is drawn from the classified corpus, preserving the corpus-level distribution across all six Major categories. Experts independently judge whether each skill's assigned Major and Sub-category are appropriate.

The average accuracy across the reviewers is **95.5%** for major categories and **92.2%** for sub-categories. Full three-way agreement is reached on 91.0% (major) and 84.5% (sub) of the 200 items. Table 11 reports the per-category breakdown.

Table 11: Per-category accuracy of LLM-based taxonomy assignment, averaged over independent reviewers on a stratified sample of 200 skills.

Major Category	n	**Major Acc.**	**Sub Acc.**
Software Engineering	120	97.5%	93.1%
AI Agents	20	91.7%	85.0%
Business & Planning	20	96.7%	96.7%
Data & ML	15	82.2%	82.2%
Content Creation	15	93.3%	93.3%
Information Retrieval	10	100.0%	100.0%
Total	**200**	**95.5%**	**92.2%**

C Query Generation

C.1 Query Generation Prompt

Each generation call receives the name and description of the sampled skill(s) as `{skills_text}`, a random subset of 165 GAIA (Mialon et al., 2024) validation questions as `{seeds_text}`, and up to 30 previously generated queries for the same skill as `{prev_section}` to suppress near-duplicate outputs. There is no system prompt. The entire instruction is issued as a single user turn. If the model judges the skill combination to be unrealistic, it outputs `None` and the combination is discarded.

User prompt.

```
Write one realistic message that a user might send to an AI
coding assistant.

The message must naturally require ALL of the following
skills to fulfill:

{skills_text}

Here are {N} examples of how real users talk to AI
assistants. Notice the variety -- questions, commands,
multi-step requests, short and long. Match this diversity
of tone and structure:

{seeds_text}

## Previously generated queries (DO NOT repeat or
## paraphrase these)
{prev_queries}

RULES:
- Do NOT always start with "I'm" or "I need". Vary the
  opening: use questions ("How do I..."), commands
  ("Set up..."), descriptions ("Our team has..."), etc.
- Do NOT mention skill names. The need must arise from
  the task description itself.
- Do NOT explain, evaluate, or comment on the skills.
  Just write the user message.
- The message must be standalone (no prior conversation
  context needed).
- Your query must be DIFFERENT from any previously
  generated query listed above. Use a different scenario,
```

```
  domain, or framing.
- If this skill combination makes no sense together in
  any realistic scenario, output exactly: None

YOUR OUTPUT (one line only -- either a user message
or None):
```

C.2 MULTI-PERSPECTIVE LLM REVIEW

Each query–skill pair that passes the leakage filter is evaluated by Claude Sonnet 4.6 using three independent reviewer prompts issued in separate API calls. Each prompt adopts a distinct evaluation persona—Skill Coherence, Query Quality, and Benchmark Discriminability—so that each dimension is assessed without anchoring bias from the others. A query is marked *invalid* if two or more reviewers return an invalid verdict. A single invalid verdict routes the query to human expert review rather than discarding it outright.

Reviewer 1 — Skill Coherence.

```
You are a benchmark quality reviewer evaluating skill-query
alignment.

SKILL(S):
{skills_block}

USER QUERY:
{query}

Does this query genuinely require the skill(s) listed above?
- Is there a meaningful semantic connection between the skill
  description and the query?
- If multiple skills are provided, does the query naturally
  require all of them?
- Mark INVALID if the skill and query are unrelated or the
  combination is forced.

Be conservative: only mark INVALID when clearly problematic.
When in doubt, mark VALID.

Respond in JSON only (no markdown):
{"verdict": "valid" or "invalid", "reasoning": "1-2 sentences"}
```

Reviewer 2 — Query Quality.

```
You are a benchmark quality reviewer evaluating query realism
and specificity.

SKILL(S):
{skills_block}

USER QUERY:
{query}

Is this a well-formed, realistic user query?
- Is the request specific and answerable?
- Could this plausibly come from a real user in a professional
  setting?
- Is the content technically coherent?
- Mark INVALID if the query is too vague, unrealistic, or
  technically incoherent.

Be conservative: only mark INVALID when clearly problematic.
```

```
When in doubt, mark VALID.

Respond in JSON only (no markdown):
{"verdict": "valid" or "invalid", "reasoning": "1-2 sentences"}
```

Reviewer 3 — Benchmark Discriminability.

```
You are a benchmark quality reviewer evaluating whether a query
can distinguish models that have access to the skill from those
that do not.

SKILL(S):
{skills_block}

USER QUERY:
{query}

Can this query discriminate between models with and without
the skill?
- Would a model lacking this specific skill fail to answer
  it well?
- Is the query too generic -- answerable by any capable model
  without specialized skill knowledge?
- Mark INVALID if the query can be answered adequately without
  the specific skill.

Be conservative: only mark INVALID when clearly problematic.
When in doubt, mark VALID.

Respond in JSON only (no markdown):
{"verdict": "valid" or "invalid", "reasoning": "1-2 sentences"}
```

D FUNCTIONAL-EQUIVALENCE AND GRADED-RELEVANCE AUDIT

Binary-label correction. For each skill in the pre-audit evaluation pool, we retrieve the five nearest evaluation and training neighbors with e5-mistral-7b-instruct. We retain candidate pairs with cosine similarity of at least 0.80 and ask GPT-5.5, using full skill bodies, whether each pair is functionally substitutable. Within the evaluation pool, union–find produces equivalence groups, and we keep one representative per group while remapping their qrels. We additionally remove evaluation skills with a train-pool functional copy and drop queries that rely on them. Re-evaluating all 19 first-stage systems raises NDCG@10 by 3.29 points on average, with a standard deviation of 0.33 and a range of 2.70–3.91. The ordering of all 171 model pairs is unchanged. Thus, the audit changes absolute scores but does not explain the fine-tuning gains or model ranking.

Graded relevance. We construct a separate diagnostic variant that retains evaluation-internal functional equivalents. Original seed skills receive relevance level 2, functional substitutes receive level 1, and unrelated skills receive level 0. We exclude evaluation skills with functional counterparts in the training pool and discard any query that loses an original seed skill. The resulting variant has 4,461 queries, with 7,307 seed query–skill pairs and 358 substitute query–skill pairs. These are relevance-label counts, not counts of unique skills. This variant differs from the default SKILLRET evaluation set because it retains equivalent skills instead of consolidating them into representatives. Table 12 compares binary and graded NDCG@10 on this variant. Only 6.7% of queries contain a substitute label. Across all 19 systems, the absolute difference between graded and binary NDCG@10 is at most 0.09 points, and the mean graded-minus-binary difference is -0.01 points. Giving partial credit to functional substitutes therefore has little effect on aggregate performance in this diagnostic variant.

Functional twins in training batches. The training set is multi-positive. Its 63,259 queries yield 127,190 positive query-skill pairs, or approximately two skills per query. A functional twin becomes

Table 12: Binary and graded NDCG@10 on the graded-relevance variant.

Model	Binary	Graded	Δ
SkillRet-Embedding-8B	85.75	85.70	−0.05
SkillRouter-Embedding-0.6B	72.91	72.97	+0.06
Qwen3-Embedding-8B	62.86	62.77	−0.09
BM25	51.03	51.02	−0.01
e5-small-v2	44.07	44.09	+0.02

an in-batch negative only when it appears in the same batch. Under strict substitutability judgments and effective batch sizes of 384 and 80, this occurs in approximately 0.3% and 0.06% of instances, and stays below 1.7% and 0.4% under a conservative bound that also counts partially overlapping pairs. The main evaluation measures retrieval after consolidating functional equivalents, while the graded-relevance experiment tests sensitivity to how substitutes are labeled. Terminal-Bench separately evaluates task completion. Together, these experiments examine retrieval gains, label sensitivity, and downstream utility.

E Top-k Reranking Depth Ablation

Table 13 reports NDCG@10 on the pre-audit split for three first-stage retrievers across reranking depths $k \in \{10, 20, 50\}$ using Qwen3-Reranker-0.6B and Qwen3-Reranker-8B. Larger k consistently improves NDCG@10 across all models and both rerankers. We adopt k=20 in the main experiments as a practical trade-off between performance and computational cost.

Table 13: Pre-audit NDCG@10 at varying reranking depths $k \in \{10, 20, 50\}$ for two rerankers across three first-stage retrievers. **Emb. Only** denotes the embedding-only baseline without reranking.

			NDCG@10 after Reranking		
Reranker	**First-Stage Model**	**Emb. Only**	**k=10**	**k=20**	**k=50**
Qwen3-Reranker-0.6B	snowflake-arctic-embed-s	52.99	58.50	61.90	65.40
	harrier-oss-v1-270m	61.17	65.40	67.54	69.90
	Qwen3-Embedding-8B	59.98	64.00	66.79	68.90
Qwen3-Reranker-8B	snowflake-arctic-embed-s	52.99	58.00	61.30	64.70
	harrier-oss-v1-270m	61.17	65.10	67.04	69.70
	Qwen3-Embedding-8B	59.98	63.70	66.44	68.80

F Document Representation Ablation

Table 14 compares two document representation strategies across three embedding models. **Name+Desc** encodes only the skill name and description, while **Full** encodes the complete document text including the name, description, and Markdown body up to the model's maximum sequence length. Full-text encoding consistently outperforms name-and-description only across all models, with gains of 1.5–11.4 NDCG@10 points.

Table 14: Effect of document representation on pre-audit NDCG@10.

Model	Name+Desc	Full
snowflake-arctic-embed-s	51.50	52.99
harrier-oss-v1-270m	55.50	61.17
Qwen3-Embedding-8B	48.60	59.98

G Model Maximum Sequence Lengths

Table 15 reports the maximum input sequence length for each model evaluated in this work. For each model, we use the maximum sequence length specified in the official model card or documentation. When no explicit limit is stated, we use the model's default context window. Encoder-only embedding models are limited to 512 tokens, which truncates the majority of skill documents in the corpus. Decoder-only models support substantially longer contexts of 8K–32K tokens, covering nearly all documents. Detailed document length statistics are in Section 4.2.

Table 15: Maximum supported sequence length per embedding and reranking model.

Method	Model	Max Tokens
Embedding		
	bge-small-en-v1.5	512
	e5-small-v2	512
	snowflake-arctic-embed-s	512
	bge-large-en-v1.5	512
	e5-large-v2	512
	F2LLM-v2-80M	40,960
	harrier-oss-v1-270m	32,768
	pplx-embed-v1-0.6b	32,768
	Qwen3-Embedding-0.6B	32,768
	jina-embeddings-v5-text-small	8,192
	harrier-oss-v1-0.6b	32,768
	NV-Embed-v1	32,768
	Octen-Embedding-8B	32,768
	Qwen3-Embedding-8B	32,768
	KaLM-Gemma3-12B	8,192
Reranking		
	jina-reranker-v2-base-multilingual	1,024
	Qwen3-Reranker-0.6B	32,768
	Qwen3-Reranker-8B	32,768

H Retrieval Prompts for Each Evaluated Model

For each model, we follow the query/document prompts recommended in the official model documentation, including model cards, READMEs, and reference implementations. Three models deviate from their default prompts.

- **Harrier-OSS and Qwen3-Embedding.** Most models use task-neutral prompts such as `query:` or `passage:`, but both of these families default to a web search specific instruction. We replace it with a skill-retrieval instruction authored for this work, shown in Table 16.
- **Octen-Embedding-8B.** Following the official README, we prepend `"- "` to each document to work around a known upstream tokenizer issue.[2]
- **Qwen3-Reranker.** The default web search instruction is replaced with a skill search instruction we authored, shown in Table 16.

Table 16 lists the final query and document prompts used for each model.

I Fine-tuning Base Model Selection

To select the base model for fine-tuning, we compared fine-tuning Qwen3-Embedding-0.6B against fine-tuning harrier-oss-v1-0.6b, which is itself a derivative of Qwen3-Embedding. We evaluate this

[2] See https://huggingface.co/Qwen/Qwen3-Embedding-8B/discussions/21.

Table 16: Query and document prompts for each model. *none* denotes no prompt applied. † Prompt authored for this work.

Model	Query Prompt	Doc Prompt
Embedding		
bge-small/large-en-v1.5	`Represent this sentence for searching relevant passages:`	*none*
snowflake-arctic-embed-s	`Represent this sentence for searching relevant passages:`	*none*
e5-small/large-v2	`query:`	`passage:`
pplx-embed-v1-0.6b	`Query:`	`Document:`
jina-embeddings-v5-text-small	`Query:`	`Document:`
harrier-oss-v1-270m/0.6b	`Instruct: Given a skill search query, retrieve relevant skills that match the query\nQuery:`†	*none*
Qwen3-Embedding-0.6B/8B	`Instruct: Given a skill search query, retrieve relevant skills that match the query\nQuery:`†	*none*
F2LLM-v2-80M	`Instruct: Given a question, retrieve passages that can help answer the question.\nQuery:`	*none*
KaLM-Gemma3-12B	`Instruct: Given a query, retrieve documents that answer the query\nQuery:`	*none*
NV-Embed-v1	*none*	*none*
Octen-Embedding-8B	*none*	–
Reranking		
jina-reranker-v2-base-multilingual	*none*	*none*
Qwen3-Reranker-0.6B/8B	`Given a skill search query, judge whether the skill document is relevant and useful for the query`†	*none*

comparison on the pre-audit evaluation set. Table 17 shows that both fine-tuned variants achieve nearly identical performance across all metrics, with differences well within noise. We therefore choose Qwen3-Embedding as the fine-tuning base to avoid double fine-tuning and to maintain a cleaner experimental provenance. The same rationale applies at the 8B scale, where we fine-tune Qwen3-Embedding-8B in preference to Octen-Embedding-8B, which is also Qwen3-Embedding-based.

Table 17: Fine-tuning base model comparison at 0.6B scale on the pre-audit split. (ft) denotes fine-tuned on SkillRet training data.

	NDCG			Recall			Completeness		
Model	**@5**	**@10**	**@15**	**@5**	**@10**	**@15**	**@5**	**@10**	**@15**
harrier-oss-v1-0.6b	64.24	66.55	67.54	67.06	73.09	76.27	50.57	57.37	61.12
harrier-oss-v1-0.6b (ft)	75.80	78.34	79.34	79.65	86.13	89.28	66.72	76.03	80.83
Qwen3-Embedding-0.6B	56.23	58.35	59.34	59.29	64.89	68.06	41.98	47.27	50.31
SkillRet-Embedding-0.6B (ft)	75.57	78.03	78.87	79.15	85.42	88.09	65.96	75.09	79.03

J SkillRet Fine-tuning Details

We fine-tune all SkillRet models on the released training split, comprising 10,123 skills and 63,259 synthetic queries yielding 127,190 positive query–skill pairs. Training and evaluation skills are disjoint.

Embedding models. We fine-tune Qwen3-Embedding-0.6B and Qwen3-Embedding-8B using MultipleNegativesRankingLoss. Each query is paired with one positive skill document per training instance, so a query with multiple ground-truth skills contributes multiple pairs, with remaining in-batch examples serving as negatives. Skill documents are encoded as `name | description`

| skill_md, matching the evaluation document representation. We apply the same skill-retrieval query instruction listed in Table 16 to anchor queries during training. Both models are trained for one epoch with maximum sequence length 8192, learning rate 2×10^{-5}, warmup ratio 0.1, bf16 precision, and gradient checkpointing on 4 GPUs. The 0.6B model uses per-device batch size 96, effective batch 384, while the 8B model uses per-device batch size 20, effective batch 80.

Reranker model. We fine-tune Qwen3-Reranker-0.6B using the same yes/no token scoring interface used at inference time. For each query–document pair, the model receives a chat-formatted prompt containing the skill-search instruction, query, and candidate skill document, and is trained with binary cross-entropy on the probability of the "yes" token versus the "no" token. Positive pairs come from the ground-truth query–skill labels. We mine negatives from four retrievers, SkillRet-Embedding-0.6B, SkillRet-Embedding-8B, Qwen3-Embedding-8B, and harrier-oss-v1-0.6b. For each retriever, we take the top 60 candidates and exclude the top 20. We pool the remaining candidates across retrievers, remove relevant skills, and sample 15 negatives per query, filling any missing slots with random negatives. The reranker is trained for one epoch with maximum sequence length 8192, learning rate 2×10^{-5}, warmup ratio 0.1, bf16 precision, and gradient checkpointing on 4 GPUs. The per-device batch size is 96, giving an effective batch size of 384.

K MTEB Retrieval vs. SkillRet Performance

Table 18 lists MTEB Retrieval scores alongside SkillRet NDCG@10 for the 14 evaluated models with available MTEB Retrieval scores, sorted by MTEB Retrieval score in descending order. A moderate positive correlation is visible in the overall trend, yet notable exceptions appear in both directions. KaLM-Gemma3-12B, for instance, leads on MTEB at 75.66 but achieves only 58.95 on SkillRet, the largest drop among all models. Conversely, some models with low MTEB scores remain highly competitive on SkillRet. harrier-oss-v1-0.6b ranks 4th on MTEB at 70.75 yet achieves the best off-the-shelf score on SkillRet at 70.26, and encoder-only models with as few as 33M parameters reach SkillRet scores in the mid-50s, comparable to NV-Embed-v1 at 7B which scores 57.03 despite a substantially higher MTEB score of 53.98. Together, these patterns suggest that skill retrieval is a distinct task from general information retrieval, requiring models to identify specific capability signals within long, multi-sentence queries.

Table 18: MTEB Retrieval score vs. SkillRet NDCG@10. Models sorted by MTEB Retrieval score in descending order.

Model	Params	MTEB Retr.	SkillRet NDCG@10
KaLM-Gemma3-12B	12B	75.66	58.95
Octen-Embedding-8B	8B	71.61	65.93
Qwen3-Embedding-8B	8B	70.88	63.64
harrier-oss-v1-0.6b	0.6B	70.75	70.26
harrier-oss-v1-270m	270M	66.38	64.56
jina-embeddings-v5-text-small	0.6B	64.88	62.96
Qwen3-Embedding-0.6B	0.6B	64.65	61.94
NV-Embed-v1	7B	53.98	57.03
F2LLM-v2-80M	80M	50.13	48.22
e5-large-v2	335M	40.06	53.41
snowflake-arctic-embed-s	33M	39.84	56.39
e5-small-v2	118M	39.38	44.66
bge-large-en-v1.5	335M	39.00	59.00
bge-small-en-v1.5	33M	36.26	54.51

L Per-Sub-category Retrieval Performance

Table 19 provides a complete breakdown of NDCG@10 and Recall@10 across all 18 Sub-categories for both base and fine-tuned Qwen3-Embedding models. Sub-categories are grouped by Major category and sorted by fine-tuned 8B NDCG@10 within each group. Each query is counted once in every sub-category represented among its gold skills. The n column therefore sums to 6,834

query–category pairs across the 4,392 evaluation queries. These detailed results complement the major-category analysis in Section 5.3.

Table 19: Per-Sub-category NDCG@10 and Recall@10 on the SKILLRET evaluation set for base and fine-tuned Qwen3-Embedding models. Sub-categories grouped by Major category. A query is counted in every Sub-category represented among its gold skills, and n is the resulting number of evaluation queries per Sub-category.

			NDCG@10				Recall@10			
Major	**Sub-category**	n	**0.6B**	**0.6B-ft**	**8B**	**8B-ft**	**0.6B**	**0.6B-ft**	**8B**	**8B-ft**
Software Eng.	Infra. & DevOps	818	64.4	83.4	66.2	89.3	68.1	90.0	71.7	95.0
	Documentation	367	56.8	80.5	55.8	87.0	60.5	87.2	60.7	92.7
	Development	1509	61.5	81.1	65.6	86.9	66.3	86.9	71.4	92.6
	Security	268	63.3	81.3	68.2	86.2	67.2	87.8	73.2	91.8
	Analysis & Testing	859	54.9	77.9	59.7	84.9	59.1	84.6	64.4	91.9
	Version Control	316	48.3	75.2	50.2	82.8	52.8	81.6	55.6	90.8
AI Ag.	Agent Evaluation	95	52.3	76.3	52.2	84.6	56.1	80.9	57.5	89.1
	Agent Development	422	48.0	74.4	49.5	83.5	52.5	81.0	54.9	91.2
	Agent Orchestration	275	41.9	68.5	41.5	76.0	47.2	72.1	45.6	81.8
B&P	Business Analysis	334	57.0	81.4	59.5	86.9	60.8	87.1	64.0	92.0
	Project Mgmt.	329	51.8	78.2	50.8	83.3	58.4	85.6	56.9	91.2
D&ML	Data Analysis	134	67.8	90.1	71.7	92.7	68.8	93.7	73.3	95.6
	ML Development	163	70.5	86.7	75.8	89.6	72.3	90.3	78.1	93.5
	Data Engineering	180	65.6	81.7	69.0	89.2	71.2	87.1	73.5	94.5
CC	Writing & Text	299	53.8	84.1	55.0	89.5	57.2	89.4	59.8	95.6
	Visual & Media	214	58.5	81.0	59.4	85.6	64.4	87.9	64.6	93.1
IR	General Search	154	48.2	76.9	48.2	83.5	50.2	81.8	52.6	90.6
	Technical Search	98	37.2	66.4	34.0	72.0	39.6	72.1	35.7	81.6

M QUALITATIVE VISUALIZATION OF SENTENCE ERASURE IMPORTANCE

To complement the aggregate masking results in Table 7, we provide a qualitative example from the same pre-audit single-skill query subset in Fig. 7. For each sentence in the query, we measure the similarity drop after replacing that sentence with `[MASK]`. A larger drop indicates that the sentence contributes more strongly to retrieving the gold skill.

N MULTI-SKILL RETRIEVAL BY CATEGORY

Table 20 reports NDCG@10 for Qwen3-Embedding-0.6B and SkillRet-Embedding-0.6B on SKILLRET. In five of the six major categories the fine-tuning gain widens monotonically from single-skill to three-skill queries, and the categories whose base score falls furthest are the ones whose gain grows most. Data & ML falls 41.6 points from k=1 to k=3 and its gain widens by 29.2.

Information Retrieval is the exception because the task is already hard at k=1. Its base score of 44.5, 43.5, and 44.6 barely moves, so the gain stays near its k=1 value of 32.3 points instead of growing. Its skills are the shortest in the corpus and it is the smallest major category, so difficulty comes from how interchangeable the documents are rather than from how many are required.

O RETRIEVAL LATENCY AND THROUGHPUT

Table 5 in the main text reports first-stage latency and throughput. We time 100 fixed queries after 10 warm-up iterations using Hugging Face Transformers, FlashAttention-2, and bfloat16 on a single NVIDIA B200. Embedding latency uses batch size 1. Throughput is measured over 1,024 queries at the largest batch size each model fits: 1,024 for both BGE models, 16 and 8 for the Qwen3 backbones

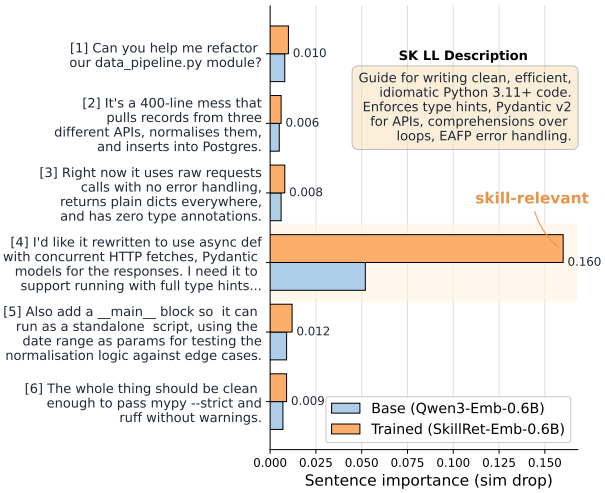

Figure 7: Sentence-level erasure importance for a pre-audit single-skill query. Each bar shows the similarity drop after replacing a sentence with `[MASK]`. The trained model concentrates more importance on the skill-relevant sentence, whereas the base model assigns importance more diffusely across the query.

Table 20: NDCG@10 by major category and number of required skills, shown as base → fine-tuned 0.6B. A query is counted in every major category among its gold skills, and NDCG is computed over its full gold set.

Category	k=1	k=2	k=3
AI Agents	48.8 → 71.2	48.6 → 74.5	39.2 → 72.3
Business & Planning	68.5 → 83.7	49.9 → 79.9	40.2 → 72.2
Content Creation	73.7 → 88.4	52.6 → 82.8	40.4 → 75.7
Data & ML	87.8 → 90.4	63.7 → 86.0	46.2 → 78.0
Information Retrieval	44.5 → 76.8	43.5 → 70.2	44.6 → 72.0
Software Engineering	79.5 → 85.9	54.1 → 79.8	40.1 → 73.4

corresponding to SkillRet-Embedding-0.6B and SkillRet-Embedding-8B, and 4 for KaLM-Gemma3-12B. BM25 runs on CPU. Thus, throughput reflects each model's feasible batching rather than performance at a shared batch size.

Table 21 reports the time to rerank one query's top 20 documents. Qwen3-Reranker-8B processes four candidates per pass, so its reported latency includes five passes. The other rerankers process all 20 candidates in one pass. The fine-tuned SKILLRET and SkillRouter checkpoints use the same architectures as their corresponding Qwen models.

Table 21: Mean latency to rerank one query's top-20 documents.

Reranker	**Params**	**Batch**	**Latency (ms)**
jina-reranker-v2	0.3B	1,024	132.1
Qwen3-Reranker-0.6B	0.6B	32	831.4
Qwen3-Reranker-8B	8B	4	1,875.5

P BROADER IMPACTS

SKILLRET is intended to support research on reliable skill retrieval for LLM agents. By isolating retrieval quality from downstream execution, it provides a controlled benchmark for studying how well models select relevant procedural knowledge from large skill libraries. This may help reduce

context cost, improve reproducibility, and diagnose retrieval failures across domains. At the same time, strong retrieval performance does not imply safe or correct end-to-end agent behavior. Retrieved skills may be outdated, unsafe, misapplied, or incorrectly composed with other skills. Therefore, SKILLRET should not be used as evidence that a deployed agent system is safe or reliable. The dataset is derived from public GitHub-hosted skills and synthetic queries. It is intended for retrieval evaluation and model development, not for profiling individual authors, inferring personal attributes, or certifying downstream agent safety. Practical deployments should include additional safeguards such as provenance checks, permission controls, sandboxing, and human oversight for high-impact actions.