SKILLECTOR: ADAPTIVE CONTEXTUAL LEAN SKILL SELECTOR FRAMEWORK

Anonymous authors
Paper under double-blind review

ABSTRACT

Reusable skills allow language-model agents to acquire specialized business or user-aligned behaviors and capabilities without modifying their underlying models. But, the real future of intelligent agents lies beyond just more skills; it lies in knowing exactly which skills to deploy, when, and why. While both language models and agent harnesses have become increasingly skill-aware, existing skill orchestrators remain mostly naïve and context-insensitive, with limited adaptation of retrieval breadth and ranking strategy to the task at hand. Such rigid orchestration will become progressively ineffective and inefficient (costly) as skill repositories expand, diversify, and accumulate overlapping capabilities. We present `SkillECTOR`, an evolving, adaptive, lean, and model-independent policy that learns how to construct a focused skill set for a given task/query as well as environment/user context with **0 LLM invocation** in its online inference path. Across multiple widely-used benchmarks (s.a. SkillsBench, AgentSkillOS, and Dataverse) our framework significantly ($\delta \approx 12 - 17\%$) improves complete-skill coverage over LLM-based selectors. We observe significant improvement in task success ($\delta \approx 4 - 5\%$) with substantially lower token usage ($\approx 20 - 30\%$) and lower inference latency ($\approx 25\%$) when `SkillECTOR` is integrated with task agents (powered by advanced LLMs). Impacts become more pronounced as skill repositories grow larger. These findings establish adaptive skill orchestration as a compact control problem and offer a scalable yet more robust alternative to placing an additional language model before every agent task.

1 INTRODUCTION

Large language model agents increasingly rely on reusable skills that encode domain knowledge, procedural instructions, and tool-use protocols. As skill libraries grow to thousands of overlapping entries, exposing every skill at inference time becomes infeasible, motivating explicit skill routing (Zheng et al., 2026). Naïve loading increases input tokens, inference latency, and serving cost before the agent begins solving the task. More critically, it creates context bloat: irrelevant or competing instructions can obscure the skills required for the task, induce incorrect skill activation, and produce brittle execution trajectories. LLM-based routing can reduce the loaded context, but it introduces an additional model call, selector-prompt overhead, and inference latency before every agent invocation. Alternatively, expecting the task-solving agents to perform skill selection prove to be equally ineffective leading to lower task success rates and higher token burden with with growing size of the skill repositories. We conjecture that effective agents should adaptively determine not only *which* skills are relevant, but also *how many* skills to expose and *how* complementary retrieval signals should be combined for the current context.

As enterprise agents are increasingly deployed in complex environments, reliability depends less on advances in foundation models and more on the quality of the execution context managed by the harness Bousetouane (2026). This context includes memory, retrieved knowledge, tool schemas, execution traces, and reusable skills that encode domain expertise. Prior work has shown that effective task solving emerges from structured interactions between reasoning and external actions rather than the model alone Yao et al. (2023); Schick et al. (2023b). In modern agent systems, skills act as declarative capability specifications, while the harness implements the skill protocol by loading context, orchestrating tool calls, managing execution state, and integrating intermediate observations. Systems such as Claude and Microsoft Copilot increasingly adopt this separation of concerns,

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.

where skills define reusable expertise and harnesses govern their execution Anthropic (2025); Microsoft (2026b;a). A critical responsibility of the harness is identifying which skills should be exposed to the agent; however, contemporary systems typically rely on matching user requests against skill names, descriptions, or metadata to determine relevance Anthropic (2025); Microsoft (2026b). Missing a necessary skill can deprive the agent of capabilities essential for task completion, whereas retrieving excessive or irrelevant skills inflates context size, increases token costs, and may reduce task performance by diluting attention over non-essential information Liu et al. (2024)

We introduce the *Evolving Contextual Skill Selector* (`SkillECTOR`), an LLM-free, model-agnostic policy for adaptive skill loading. Unlike in-policy approaches that finetune or context-engineer the agent model itself to select skills (Li et al., 2026b), our selector operates externally and makes no LLM call on the online selection path. Inside it is a tri-modal ranker that leverages pre-computed lexical, semantic, and reasoning-aware indexes and ranks the skill library; a lean reinforcement-learning policy predicts retrieval depth K and rank-fusion parameters; and deterministic fusion returns the final Top-K subset. Reflective LLM judge is restricted to an asynchronous offline path that generates delayed rewards from large buffers of completed trajectories. Across skill catalogs/benchmarks containing $\sim$ 200 to $\sim$ 1000 skills, our framework consistently outperforms state-of-the-art LLM-based baselines, achieving up to $\approx 27\%$ higher F1 score of selection and up to $\approx 20 - 30\%$ higher rate of skill coverage. The gains become more pronounced at larger scales, and initial end-to-end evaluation (using agents with most powerful LLMs with longest possible context window) shows that selectively exposing relevant skills via `SkillECTOR`improves actual task success ($\approx 58\%$ vs. $\approx 54\%$) over loading the entire skill catalog, demonstrating benefits for both agent effectiveness and context efficiency ($\approx 2\times$).

We make the following contributions -
(1) We put forward a robust generalizable formalism of the skill subset selection as a contextual decision problem over evolving agent execution state and the *dual* decision space.
(2) We propose `SkillECTOR`, a novel LLM-free retrieval, ranking, and selection framework that adaptively controls skill exposure without online LLM invocation[1].
(3) We validate `SkillECTOR` across diverse skill ecosystems, demonstrating improvements in skill-selection quality, end-to-end task success, and context efficiency.

2 Background and Related Work

Tool-Augmented Agents and Reusable Skills. Language-model agents increasingly rely on external tools and reusable skills. Toolformer learns API invocation through self-supervision (Schick et al., 2023a), Gorilla supports retrieval-augmented API use in evolving tool ecosystems (Patil et al., 2023), and Voyager retrieves executable skills from a growing library (Wang et al., 2023). However, these approaches do not address dynamically adapting the exposed skill set to changing task contexts, library evolution, and cost constraints.

Skill and Tool Retrieval Recent work formulates large-library access primarily as retrieval. ToolReAGT iteratively retrieves tools for decomposed subtasks (Braunschweiler et al., 2025); SkillRouter uses a retrieve-and-rerank pipeline and highlights the importance of full skill bodies (Zheng et al., 2026); and SkillRet provides a large-scale benchmark and task-specific retrievers for skill selection (Kang et al., 2026). These approaches optimize retrieval scores or fixed-depth candidate lists. In contrast, our framework treats both retrieval depth and ranking composition as context-dependent decisions, jointly optimizing task utility, skill-set precision and recall, and serving cost.

Learning and Evaluating Selection Policies SkillGate trains in-policy skill selection by assigning action-local credit to skill-selection tokens, addressing credit starvation in long-horizon trajectories (Li et al., 2026b). Our setting is complementary: the selector is an external, lightweight policy that can evolve without modifying the underlying agent model. It builds on value–advantage decomposition in dueling Q-networks (Wang et al., 2016) and learns from delayed trajectory-level feedback. LLM-as-a-judge offers scalable evaluation but exhibits position, verbosity, and self-enhancement biases (Zheng et al., 2023). We therefore use structured judging together with trace fidelity and with-versus-without-skill ablations, converting counterfactual evidence into delayed rewards for continual selector optimization.

[1] We will publicly release all synthetic tasks, DVB skills and `SkillECTOR` code upon paper acceptance.

3 METHODOLOGY

SkillECTOR combines complementary retrieval signals with a reinforcement-learning controller to adapt skill-selection depth and rank fusion, without modifying the downstream agent, retrieval encoder, or skill definitions.

3.1 THE PROBLEM

Agent Skills, Skill Protocol, and Progressive disclosure An agent skill encapsulates functionality behind a standardized specification, enabling heterogeneous capabilities to be represented uniformly across implementations and execution environments Hong et al. (2026); Agent Skills; Zhang et al. (2025b); Wang et al. (2024; 2025); Mi et al. (2026); Jiang et al. (2026); Xu & Yan (2026). Let $\mathcal{S} = \{s_1, \ldots, s_N\}$ denote the skill catalog, where each skill can be defined as, $s_i = \langle \mathfrak{F}_i, \tau_i, \phi_i, \psi_i, \kappa_i, \rho_i \rangle$.. Here, $\mathfrak{F}_i$ denotes **frontmatter**, τ_i trigger conditions (often τ_i can be subsumed by $\mathfrak{F}_i$), ϕ_i and ψ_i input and output schemas, κ_i the executable capability, and ρ_i operational properties. Together, they form a runtime-independent declarative specification.

Skills are passive abstractions; discovery, loading, invocation, and monitoring are delegated to the agent harness through *progressive disclosure*. Rather than exposing all procedural knowledge upfront, the harness loads skill content incrementally. Anthropic's Agent Skills expose only frontmatter $\mathfrak{F}_i$ initially and retrieve SKILL.md content and bundled resources on demand Zhang et al. (2025a); Anthropic (2025). Microsoft's Agent Framework similarly separates advertisement, loading, resource access, and execution into explicit skill operations, maintaining near-constant baseline context cost while supporting effectively unbounded skill knowledge Microsoft (2026a). Empirical studies show that progressive disclosure substantially influences agent execution trajectories, although a single disclosure stage is often sufficient Chen et al. (2026); He et al. (2026).

Prolem formulation We focus our scope to one of the primary responsibilities of the harness, the skill-subset selection/recommendation problem. Given a query q, we aim to recommend a task-relevant subset $S_q \subseteq \mathcal{S}$ with adaptive cardinality $K_q = |S_q|$. *The objective is to recover required capabilities while limiting unnecessary exposure.* Loading the entire library introduces irrelevant or competing instructions, whereas a fixed, small subset risks omitting required skills.

Asking the controller to directly predict skill identities is risky. We will encounter out-of-bounds issues in the decision space as new skills are added to the skill repositories. Instead, we propose to solve the dual of the problem where we make the configuration parameters and retrieval depth constraint our decision variables.

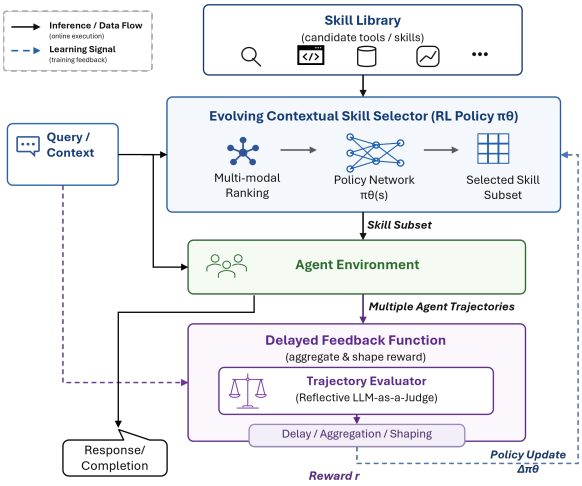


Figure 1: Architecture of SkillECTOR.

We aim to learn a controller policy $\pi_\theta(K, \vec{W}, \vec{\kappa}|\mathbf{x})$ such that we minimize the joint loss, $\arg\min_{K,\vec{W},\vec{\kappa}} \mathcal{L}(\frac{1}{Task\ Success} + LLM\ Cost)$ where, $(K, \vec{W}, \vec{\kappa})$ is our decision space, K is the retrieval depth i.e. subset size, $\vec{W}$ are the fusion weights over the tri-modal ranker, and ς are additional parameters of the each ranker mode and $x \in \mathcal{X}$ is the context or state. Thus the final recommendations are generated as a final transformation, such as $S_q = \text{Top-K*}(UnifiedRank_{\vec{W}^*,\vec{\kappa}^*}(\mathcal{S}))$, where $K^*, \vec{W}^*, \vec{\kappa}^* = \pi^*(K, \vec{W}, \vec{\kappa}|\mathbf{x}_q)$. As is clear, this formulation allows arbitrary extension of $\mathcal{S}$ without change in decision space. *Task Success* or *LLM Cost* does not need separate metrics. They can be directly estimated via agent execution with S_q.

3.2 THE SKILLECTOR FRAMEWORK

SkillECTOR separates retrieval, selection, and learning. Tri-modal retrievers/rankers supply complementary ranks given a query, and a learned controller adapts their fusion and unified retrieval depth K. Figure 1 summarizes the framework and its interfaces to agent execution and feedback. Given a task/query q, the controller's state space comprises the query embedding and summaries from each of the modes of our tri-modal ranker. The controller recommends fusion parameters (and additional parameters of each ranker mode) and subset-size (retrieval depth K); fused/unified tri-modal rank with top-K retrieval returns the selected skills to the downstream agent with zero LLM invocation. The task agent is responsible for reasoning, tool interactions, and response generation for q. Controller policy update with feedback from task executions lies outside the hot/inference path. The backbone reflective LLM judge, emits delayed reward signals based on the buffer of a batch of task execution trajectories. Delayed reward signals leads to $1/\text{buffe size}$ (often $<< 10^{-3}$) LLM invocations per task.

3.2.1 MULTI-MODAL SKILL RANKING

Skill Representation and Index Construction. As described in section 3.1, each skill comprises: is name and desciption $\in \mathfrak{F}_i$, its procedural document. We pre-compute 3 kinds of indexes — lexical, skill-semantic indexes with $\mathfrak{F}_i$ and procedural body resp., and instruction-semantic index from **reasoned** synthetic user query examples. The semantic indexes store frozen sentence-transformer embeddings; the instruction index retains individual example user requests and a centroid per skill. Reasoning over skill content produces example user requests offline, covering different sub-capabilities and formulations. These requests are distinct from the procedural instructions in the skill document; this reasoning *does not occur at query time*. Catalog changes require corresponding index updates, not additional policy outputs for new skill identities.

Tri-modal ranker design. We employ three complementary text-based ranking modals: BM25, Dense, and Instruction-Dense. Figure 2 shows the design, inputs and policy-guided unification.

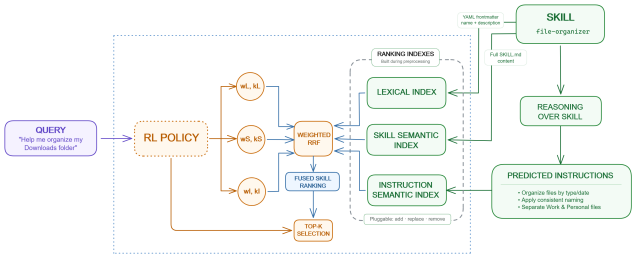


Figure 2: Multi-modal skill ranking and unification.

BM25 $\boxed{\mathfrak{R}_B}$**.** BM25 scores queries against skill names and descriptions after lowercasing, tokenization, stop-word removal, and Porter stemming. It captures lexical matches involving tool names, domain terms, and capability descriptions.

Dense $\boxed{\mathfrak{R}_D}$**.** Dense ranks skills by cosine similarity between the query embedding and precomputed skill-document embeddings. Our configuration uses the full skill-document text as encoder input, supporting semantic matching beyond shared vocabulary.

Instruction-Dense $\boxed{\mathfrak{R}_I}$**.** Let $\mathbf{u}_{s,j}$ be the unit-normalized embedding of synthetic user request j for skill s (examples in Appendix A), and $\mathbf{c}_s$ the unit-normalized mean of these embeddings. For a unit-normalized query embedding $\bar{e}(q)$, the raw instruction-semantic score is $r_{\mathrm{I}}(q,s) = \alpha\,\bar{e}(q)^\top \mathbf{c}_s + (1-\alpha)\max_j \bar{e}(q)^\top \mathbf{u}_{s,j}$. We use $\alpha = 0.5$ to balance typical-use similarity with the strongest example match, helping capture specific use cases that centroid averaging may dilute.

Both dense modals share one query embedding, and each modal ranks the full catalog. Scores are min–max normalized within each modal, preserving their ordering and supporting policy-state construction; constant-score lists are mapped to zero. Unification/fusion is over rank positions rather than score magnitudes, avoiding numerical instability.

3.2.2 STATE AND ACTION SPACES

State representation. The state $\mathbf{x}_q$ combines a 384-dimensional query embedding with four score summaries per modal over a fixed window of leading results: mean and standard deviation divided

by the leading score, leading score divided by its sum with the median, and the nonnegative gap to the top-five mean divided by the leading score. Zero denominators are safeguarded. Pairwise overlap counts at five and ten, plus top-ten Jaccard overlap across all modals, capture ranking agreement. The trimodal configuration yields 403 features (384 embedding values, 12 score statistics, seven agreement features); bimodal and unimodal configurations state dimensions change only wrt score/agreement features and is independent of catalog size. (Details in Appendix B).

Action representation. The decision/action space $a_q = (K_q, \vec{W}_q, \vec{\kappa}_q)$ specifies subset-size and vector of unification weights and per mode reciprocal-rank fusion (RRF) smoothing constants respectively. For a particular rank mode $f \in$, the unification weights and modal RRFs are discretized into bins as $w_{q,f} \in \{0.2, 0.5, 0.8\}$, and $\kappa_{q,f} \in \{10, 60, 200\}$. Subset size $K_q \in \{1, \ldots, N\}$ is discrete integer by default (In our experiments $N = 10$ suffices). This yeilds $\approx 7K$ actions for the tri-modal configuration and lower for bi/uni-modal. Weights have no sum-to constraint. This identity-independent action space stays fixed as the library grows.

3.2.3 Adaptive Fusion and Top-K Selection

Given the selected action, `SkillECTOR` unifies the modal rankings using weighted reciprocal-rank fusion. The fused/unified rank of skill s is,

$$\boxed{\mathfrak{R}_U}(s \mid \vec{W}_q) = \sum_{f \in \mathcal{F}} \frac{w_{q,f}}{\kappa_{q,f} + \mathfrak{R}_f(q, s)}, \tag{1}$$

where $\mathcal{F} \subseteq \{B, D, I\}$ is the set of active ranking modes (Sec. 3.2.1). Top-$K_q(\boxed{\mathfrak{R}_U}(\mathcal{S}))$ where $K_q \leftarrow a_q$, yields the selected subset S_q, and $\kappa_{q,f}$ shapes per-modal ranking.

3.2.4 Selection Feedback and Reward Model

Reward modeling is a crucial factor in learning a good policy in this setting. Response quality or task completion are not sufficiently representative non-degenerate reward functions (they do not subsume the implicit objective). Hence, our reward combines selection quality and action calibration with domain-adjustable hyperparameters (Appendix C). Now, we assume that we may not have enough ground truth to train from, which is true in most enterprise scenarios. Thus, we use a generative reward model. We leverage one-shot GPT-4.1-mini as an LLM judge to (A.) evaluate skill usefulness and (B.) to predict required-skill coverage (check Appendix D to verify how GPT-4.1-mini is the most stable judge). If enough is available, we can choose to leverage ground truth and not a judge.

Selection feedback. Let the LLM-judge predicted required-skills for a query q be the pseudo ground truth $GT_q^{(judge)}$. We now compute, $\boxed{\text{meta-precision}}$ $Prec_q = (|z| \text{ s.t. } z \in S_q, z \in GT_q^{(judge)})/|S_q|$ measures the useful fraction of selected skills; $\boxed{\text{meta-recall}}$ $Rec_q = (|z| \text{ s.t. } z \in S_q, z \in GT_q^{(judge)})/|GT_q^{(judge)}|$ measures required-skill coverage for a given task q. Their harmonic mean is $F1_{,q}$ (zero when both vanish). We also compute $\boxed{\text{ranking fidelity}}$ $\mathcal{F}_{\text{rank},q}$, which is the reciprocal mean distance of misses ($GT_q^{(judge)} \setminus S_q$) beyond K_q, or $\mathcal{F}_{\text{rank},q} = 1$ without misses. The selection error is, $E_{\text{sel}}(q) = \lambda_F(1 - F1_q) + \lambda_P(1 - Prec_q) + \lambda_R(1 - Rec_q) + \lambda_{\text{rank}}(1 - \mathcal{F}_{\text{rank},q})$. The nonnegative weights $\lambda_F, \lambda_P, \lambda_R, \lambda_{\text{rank}}$ correspond to F1, meta-precision, meta-recall, and ranking-fidelity errors, respectively; they sum to one, with zero weights omitting terms.

Action calibration feedback. Let $E_K(q)$ denote the absolute mismatch between the predicted subset cardinality K_q and reference cardinality $|GT_q^{(judge)}|$, normalized by the configured upper and lower bound for K. Let $E_W(q)$ is the mean absolute deviation between predicted ranker mode weights and their targets. Each target is the reciprocal of 1 plus the mean distance of the missed skills, $GT_q^{(judge)} \setminus S_q$, beyond that mode's cutoff K_q; distances are zero for skills within the cutoff, and with no misses all targets $= 1$. So, the action error is $E_{\text{act}}(q) = \eta E_K(q) + (1 - \eta)E_W(q)$; $\eta \in [0, 1]$. Here, η is the trade-off between cardinality error and mode-weight error.

Composite reward. Both errors contribute equally to the exponential reward:

$$\text{Reward}(q, a_q) = \exp\left[-\tau\left(E_{\text{sel}}(q) + E_{\text{act}}(q)\right)\right]. \tag{2}$$

The hyperparameter $\tau > 0$ controls reward sharpness. RRF constants receive feedback indirectly through their effects on ranking and selection.

3.2.5 POLICY OPTIMIZATION AND INFERENCE

SkillECTOR uses a Rainbow-based deep Q-network (DQN) controller Hessel et al. (2018) combining a dueling distributional network, prioritized experience replay, and NoisyNet exploration. Two shared 256-unit hidden layers feed mean-centered value and advantage streams, producing distributions over 51 uniformly spaced reward atoms in $[0, 1]$. Actions maximize discounted expected reward. Each query is a one-decision episode without bootstrapping. After warm-up, reward-prediction errors prioritize replay. Rewards are projected onto the categorical support; cross-entropy loss is importance-weighted and normalized by the weight sum. We use Adam optimizer with gradient clipping and learned parameter noise drives exploration. The encoder and downstream agent remain unchanged. Inference freezes controller parameters and disables noise, yielding deterministic greedy selection without annotations or reward computation.

4 EVALUATIONS

We evaluate SkillECTOR in against zero-shot context-engineered LLM-based selectors of varying reasoning power (GPT-5.4-nano, GPT-5.6-sol, and GPT-6 Astra). We examine (1) skill-selection quality and selection/loading cost across 3 domains/benchmarks, (2) task-level success and AI token foot footprint using powerful task agent (with high context window backbone LLM) on a unified benchmark with and without SkillECTOR, and (3) ablation studies on impact of skill diversity on selection quality/cost and on retrieval depth bloating and impact of ranking modes on quality. We aim to answer the following evaluation questions: **[Q1]** Does SkillECTOR lead to higher selection quality? **[Q2]** What is the cost impact of contextual skill selection (How lean are we)? **[Q3]** Is SkillECTOR even needed for modern agents (with SOTA LLM backbone)?

4.1 EXPERIMENTAL SETUP

Benchmarks and catalogs. The evaluation spans three skill ecosystems. SkillsBench contains 94 tasks and 209 canonical skills (Li et al., 2026c). AgentSkillOS contains 30 tasks, evenly distributed across five categories, and is evaluated with 50-, 500-, and 1,000-skill catalogs (Li et al., 2026a). The Dataverse benchmark comprises 183 synthetically constructed tasks mapped to a catalog of 16 original Dataverse skills. Unlike other open generic benchmarks, **Microsoft Dataverse** is the cloud data platform underpinning Power Platform and Dynamics 365, acting as the business system of record for entities such as accounts, contacts, cases, and custom tables in a secure, relational, metadata-driven store with role-based security, business rules, and auditing Microsoft (2026d). It is the enterprise data layer grounding Copilot Microsoft (2026b). Dataverse is more demanding than a plain database: relationships, option sets, and permission boundaries constrain every access Microsoft (2026c). Skills supply the knowledge to navigate them— locating tables, reading schemas, honoring roles—across operations spanning discovery, filtered reads and aggregations, create, update, flow invocation, environment-scoped governance etc..

Retrieval engines and ensembles. We evaluate four modal configurations of rankers: Tri-modal/full (BM25 + Dense + Instruction-Dense), and 3 Bi-modals (BM25 + Dense, BM25 + Instruction-Dense, and Dense + Instruction-Dense). The full tri-modal ranking is used for all primary evaluations, while bi-modal configurations are used for ablation study.

Evaluation metrics and baselines. Selection quality is measured by Task-Skill Alignment Rate (**TSAR**) which is the percentage of tasks where predicted/recommended skills cover or align with all ground-truth skills of the given tasks. Presicion, Recall and F1 are per task metrics comparing the match between Top-K recommended skills and ground truth skills. In the tables we report average values over the query/task set. Zero-shot context engineered GPT-5.4-nano, GPT-5.6-sol, and GPT-6 Astra selectors are used as selector baselines across all three benchmarks.

4.2 RESULTS

4.2.1 SELECTION QUALITY ACROSS SKILL ECOSYSTEMS

SkillsBench. Table 1 show SkillECTOR significantly outperforms all baseline selectors in terms of primary quality metric TSAR ($diff \approx 12\%$), mean Recall & F1. Figure 3 additionally shows

Table 1: SkillsBench performance comparison on the original 209-skill catalog.

Selector	Avg. Precision	Avg. Recall	Avg. F1	TSAR (%)
SkillECTOR	0.803	**0.858**	**0.818**	**67.0**
GPT-5.4-nano	0.694	0.672	0.644	42.6
GPT-5.6-sol	**0.878**	0.786	0.808	55.3
GPT-6 Astra	0.866	0.771	0.792	55.3

Figure 3: Learning curves - SkillsBench.

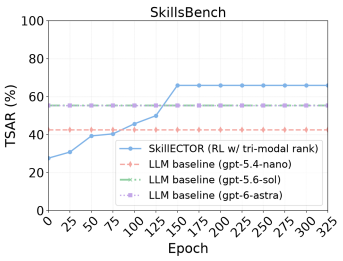
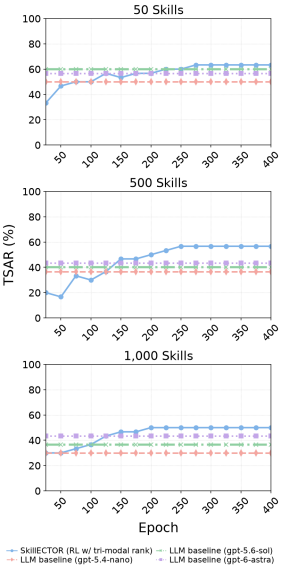


Table 2: AgentSkillOS selection qlty w/ increasing catalog size.

Skills	Selector	Avg. Prec.	Avg. Rec.	Avg. F1	TSAR (%)
50	SkillECTOR	0.711	**0.825**	0.733	**63.3**
50	GPT-5.4-nano	0.628	0.775	0.665	50.0
50	GPT-5.6-sol	0.703	0.778	0.722	60.0
50	GPT-6 Astra	**0.747**	0.756	**0.744**	56.7
500	SkillECTOR	0.591	**0.761**	0.628	**56.7**
500	GPT-5.4-nano	0.410	0.575	0.461	36.7
500	GPT-5.6-sol	0.623	0.686	0.639	40.0
500	GPT-6 Astra	**0.639**	0.678	**0.648**	43.3
1,000	SkillECTOR	0.400	**0.706**	0.460	**50.0**
1,000	GPT-5.4-nano	0.423	0.594	0.479	30.0
1,000	GPT-5.6-sol	0.612	0.647	0.619	36.7
1,000	GPT-6 Astra	**0.628**	0.667	**0.637**	43.3

Figure 4: Learning curves - AgentskillOS



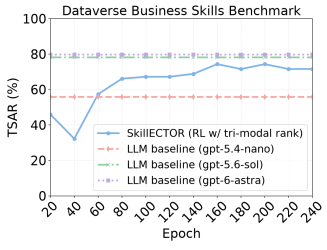
the learning curves for TSAR. SkillECTOR surpasses all three LLM selectors and stabilizes at surpasses GPT-5.4-nano performance quite early on, and reaches peak perf. within 150 epochs.

AgentSkillOS. We evaluate SkillECTOR on the same 30 AgentSkillOS tasks using increasing skill catalog size (50, 500, and 1,000 skills). Table 2 illustrates how SkillECTOR outperforms all selector baselines in terms of TSAR. Interesting thing to note here is that the difference with the baseline TSAR-s are more substantial ($\delta \approx 13\%$) when the the skill catalog is large (500, 1000). *This corroborates out conjecture that our framework will be increasingly indispensable as skill repositories/catalogs grow.* Another interesting aspect is that SkillECTOR achieves higher recall which in turn leads to better coverage (TSAR). Learning curves in Figure 4 align with our understanding. It shows how for small skill catalog (50) SkillECTOR takes longer to actually outperform the strongest selector baselines (GPT5.6-sol in this case).

Table 3: Dataverse selection quality on the 16-skill catalog.

Selector	Precision	Recall	F1	TSAR (%)
SkillECTOR	0.609	0.888	0.691	77.1
GPT-5.4-nano	0.697	0.771	0.697	55.7
GPT-5.6-sol	0.905	0.908	0.893	78.1
GPT-6 Astra	**0.909**	**0.917**	**0.903**	**79.8**

Figure 5: Learning curves DVB.

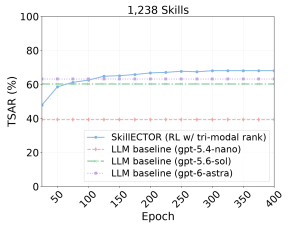


Dataverse. We evaluate SkillECTOR on 183 synthetic Dataverse Business (DVB) tasks using the catalog of 16 existing business skills. Table 3 as well as Figure 5 show that while SkillECTOR clearly outperforms the 'nano' baseline. but falls short against the stronger baselines (sol and astra). Our studies have shown, supported by our conjecture, that when skill catalogs are extremely small (16 here) Strong LLM selectors can easily do the job well, at zero-shot.

Combined Skill Ecosystem. To stress test SkillECTOR on skill and task diversity, we combine SkillsBench, AgentSkillOS, and Dataverse into 307 tasks and 1,238 skills, ranking each query against the shared catalog. We observe SkillECTOR achieves the highest recall (0.825) and TSAR (68.1%), exceeding GPT-6 Astra by 4.9 percentage points in TSAR. Full comparison in Appendix E.

Although cost results in the next section paint the full picture of the necessity of a lean skill selector, the results in this section demonstrate improved quality/coverage of skill selection, affirming **Q1**.

Figure 6: Learning curves - Combined ecosystem.



4.2.2 Cost Impact

Table 4 compares selection tokens, loaded-skill tokens, and latency. SkillECTOR incurs no LLM selection tokens or selector API expenditure and achieves the lowest latency in every setting. Arrow-marked rows additionally show the lowest combined selection + loaded-skill token footprint. Dataverse (no arrow) is the exception: its small 16-skill catalog limits selection-token savings, which do not offset SkillECTOR's larger loaded-skill context. Thus, SkillECTOR reduces latency and combined selector token footprint on every benchmark establishing clear efficiency benefit in **Q2**.

Table 4: Cost comparison across skill ecosystems. K denotes 1,000 tokens. Selectors: **O** = SkillECTOR, **5.4Nano** = GPT-5.4-nano, **5.6Sol** = GPT-5.6-sol, **6Astra** = GPT-6 Astra.

	Selection tokens/task				Loaded-skill tokens/task				Latency (s)			
Benchmark	O	5.4Nano	5.6Sol	6Astra	O	5.4Nano	5.6Sol	6Astra	O	5.4Nano	5.6Sol	6Astra
→SkillsBench (209)	0	13.70K	13.70K	13.70K	4.32K	3.63K	3.67K	3.43K	1.24	2.04	6.46	8.14
Dataverse (16)	0	1.72K	1.72K	1.72K	18.59K	12.56K	10.64K	10.82K	1.18	1.43	3.52	7.72
→AgentSkillOS (50)	0	4.24K	4.24K	4.24K	8.09K	6.80K	6.61K	6.47K	1.21	1.67	2.85	8.25
→AgentSkillOS (500)	0	37.00K	37.00K	37.00K	9.92K	9.17K	6.96K	6.84K	1.33	2.02	4.53	9.80
→AgentSkillOS (1,000)	0	75.51K	75.51K	75.51K	8.95K	9.59K	6.62K	6.90K	1.32	4.36	5.85	11.34
→Combined (1,238)	0	90.86K	90.86K	90.86K	14.92K	9.66K	8.11K	7.94K	1.28	8.17	6.53	8.23

4.2.3 Deployment Study

We evaluate 81 SkillsBench tasks in the GitHub Copilot (GHCP) Task Agent harness using GPT-6 Astra with maximum context length and reasoning, and the Benchflow BenchFlow Team (2026) verifier. We compare, Native and SkillECTOR configurations with 209 and 1209 skills (merging 1000 skills from AgentSkillOS). Oracle (exact GT skills) and No-skill (0 skills) establish upper and lower bounds. Native exposes the ***full skill catalog***; SkillECTOR recommends a task-aware subset. Table 5 reports TSAR, verified task success rate (TSR), and execution costs (tokens & money). **TSR measures actual task execution by task agent**. SkillECTOR improves task success and efficiency over Native-1209 in the GHCP harness, affirming **Q3**. *Note, as catalog size increases from 209 to 1209, Native starts degrading substantially but SkillECTOR maintains good performance.*

Table 5: E2E Task execution Experiment; Quality and Cost (token pricing details in Appendix F)

Experiment	Quality		Cost							
	TSAR	TSR	Avg. Latency (min/task)	Total Tool Calls	Avg. Tokens/ Tool Call	Total Tokens (81 tasks)	P50 (/task)	P90 (/task)	P99 (/task)	Projected Cost (USD)
No-skill	0.00%	49.38%	11.76	2,383	21,836.78	52.037 M	0.374 M	1.274 M	3.499 M	$349.65
Native-209	60.49%	58.02%	13.04	2,430	24,006.95	58.337 M	0.543 M	1.563 M	2.710 M	$366.50
SkillECTOR-209	62.96%	58.02%	13.06	2,392	23,121.51	55.307 M	0.410 M	1.608 M	3.537 M	$354.18
Native-1209	14.81%	54.32%	17.39	2,777	27,605.62	76.661 M	0.544 M	1.776 M	5.301 M	$401.34
SkillECTOR-1209	62.96%	58.02%	12.79	2,396	22,352.37	53.556 M	0.463 M	1.457 M	2.570 M	$343.51
Oracle	87.65%	61.73%	12.69	2,584	23,834.30	61.588 M	0.426 M	1.657 M	3.318 M	$380.06

4.3 Ablation Studies

The following controlled ablations use SkillsBench to examine how skill diversity along with retrieval-depth calibration, and ranker mode composition affect selection performance. V 1 through V 5 use 450-skill catalogs containing canonical skills and partial duplicates (*V*ariants) with varying diversity. Across the five variants, skill representations are progressively enriched with more descriptive metadata, clearer task framing, explicit usage conditions, and increasingly detailed operational guidance.

Skill Representation. Figure 7 compares precision, recall, F1, and canonical TSAR across variants for four retriever configurations of `SkillECTOR`. For the tri-modal setup, and across all the quality metrics there is a sharp drop as soon as variants are introduced. But across increasing diversity levels there is hardly any change.

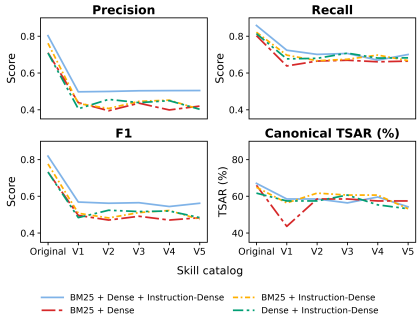


Figure 7: Selection quality across SkillsBench skill-representation conditions for the four retrieval ensembles.

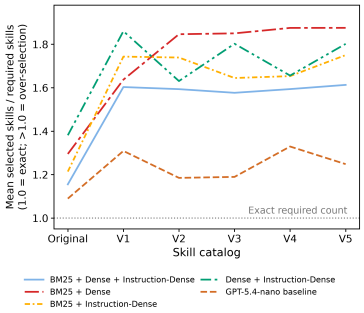


Figure 8: Retrieval-depth bloating across SkillsBench catalog variants.

Retrieval-Depth Bloating. Figure 8 reports the mean bloat factor (predicted subset size relative to the required ground-truth skill count), where 1.0 represents exact selection. On the original catalog, `SkillECTOR` with the tri-modal ranker recorded 1.16, while the GPT-5.4-nano selector recorded 1.09. As variants are introduced (v1) we observe sharp rise in bloat factor; however across the variant catalogs with increasing diversity we see plateaued bloat factor. But we do see strong positive correlation and static difference among `SkillECTOR` w/ different ranker and 'nano' baselines.

Table 6: Ablation on impact of ranker modes.

Ranker config	Precision	Recall	F1	TSAR (%)
Tri-modal	**0.803**	**0.858**	**0.818**	**67.0**
Bi-modal ($\mathfrak{R}_B \oplus \mathfrak{R}_D$)	0.710	0.803	0.732	66.0
Bi-modal ($\mathfrak{R}_B \oplus \mathfrak{R}_I$)	0.761	0.821	0.776	64.9
Bi-modal ($\mathfrak{R}_B \oplus \mathfrak{R}_I$)	0.708	0.814	0.731	61.7

Retriever Contributions. These results clearly indicate that the tri-modal provided the strongest overall performance, while BM25 + Instruction-Dense produced the highest F1 among the two-retriever ensembles.

5 Conclusion

We introduced `SkillECTOR`, a model-independent framework that casts skill orchestration as a contextual control problem rather than an additional language-model call. By jointly adapting retrieval depth and rank fusion, our framework improves required-skill coverage, task success, latency, and context efficiency across expanding skill ecosystems. These results show that reliable agent behavior depends not only on stronger foundation models, but also on construction of the execution context exposed by the harness. More broadly, this work motivates research on continual selector adaptation under catalog drift, richer state representations incorporating memory and execution traces, uncertainty-aware skill exposure, and reward learning from counterfactual feedback. Extending adaptive orchestration across skills, tools, memory, and knowledge may provide a principled foundation for efficient, robust, and personalized agent harnesses.

AI DISCLOSURE

In this work, we used generative AI tools for synthetic task generation for Dataverse benchmark, and error analysis in our software when and if any error occurred. We have not used generative AI tools for any of the other required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for grammar and language check only on sparse occasions. We have reviewed all AI-assisted work. (We checked AI-assisted grammar corrections manually to see if the semantics have changed and LLM-assisted error debugged code was verified and tested for correctness by 3 author). We take responsibility for the final content of this work.

REFERENCES

Agent Skills. Agent skills specification. Open specification. URL https://agentskills.io/specification.

Anthropic. Agent skills: Overview. Claude Platform Documentation, 2025. URL https://platform.claude.com/docs/en/agents-and-tools/agent-skills/overview. Level 1 metadata (always loaded), Level 2 instructions (loaded when triggered), Level 3 resources and code (loaded as needed).

Anthropic. Extending claude's capabilities with skills and mcp servers. https://claude.com/blog/extending-claude-capabilities-with-skills-mcp-servers, 2025. Accessed: 2026-09-23.

BenchFlow Team. Benchflow: framework for rl environments for llm agents. https://github.com/benchflow-ai/benchflow, 2026.

Fouad Bousetouane. Ai agents do not fail alone:the context fails first, 2026. URL https://arxiv.org/abs/2607.14275.

Norbert Braunschweiler, Rama Doddipatla, and Tudor-Catalin Zorila. ToolReAGt: Tool retrieval for LLM-based complex task solution via retrieval augmented generation. In *Proceedings of the 1st Workshop on Knowledge Augmented Methods for Natural Language Processing*, 2025. URL https://aclanthology.org/2025.knowllm-1.7/.

Zhiyu Chen, Zihan Guo, Bo Huang, Bingwei Lu, Jianghao Lin, Yuanjian Zhou, and Weinan Zhang. SkillJuror: Measuring how agent skill organization changes runtime behavior. *arXiv preprint arXiv:2606.11543*, 2026. doi: 10.48550/arXiv.2606.11543. URL https://arxiv.org/abs/2606.11543. 82-task SkillsBench study; distinct skill resources touched per trajectory rise from 1.18 to 3.85 and effective uptake events from 1.33 to 3.92, with +4.1% verifier-passing trials.

Yifeng He, Yinzhe Zhao, Jicheng Wang, and Hao Chen. Is progressive disclosure all you need for long-context agents? *arXiv preprint arXiv:2607.17598*, 2026. doi: 10.48550/arXiv.2607.17598. URL https://arxiv.org/abs/2607.17598. First controlled study of the pattern on InfiniteBench across three harnesses and three model families; one disclosure level is optimal, a second routing level never helps.

Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. doi: 10.1609/aaai.v32i1.11796. URL https://ojs.aaai.org/index.php/AAAI/article/view/11796.

David Boram Hong, Aaron Imani, and Iftekhar Ahmed. From anatomy to smells: An empirical study of SKILL.md in agent skills, 2026. URL https://arxiv.org/abs/2607.01456.

Yanna Jiang, Delong Li, Haiyu Deng, Baihe Ma, Xu Wang, Qin Wang, and Guangsheng Yu. SoK: Agentic skills—beyond tool use in LLM agents, 2026. URL https://arxiv.org/abs/2602.20867.

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

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, 2026a. URL https://arxiv.org/abs/2603.02176.

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

Xiangyi Li et al. SkillsBench: Benchmarking how well agent skills work across diverse tasks, 2026c. URL https://arxiv.org/abs/2602.12670.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

Qirui Mi, Zhijian Ma, Mengyue Yang, Haoxuan Li, Yisen Wang, Haifeng Zhang, and Jun Wang. Skill-pro: Learning reusable skills from experience via non-parametric ppo for llm agents. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. doi: 10.48550/arXiv.2602.01869. URL https://openreview.net/forum?id=9kJQjx2B80.

Microsoft. Harnesses in copilot studio. https://learn.microsoft.com/en-us/microsoft-copilot-studio/harnesses-overview, 2026a. Accessed: 2026-09-23.

Microsoft. Skills overview for agents. https://learn.microsoft.com/en-us/microsoft-copilot-studio/agents-experience/skills-overview, 2026b. Accessed: 2026-09-23.

Microsoft. Agent skills — Microsoft Agent Framework. Microsoft Learn, 2026a. URL https://learn.microsoft.com/en-us/agent-framework/agents/skills. Four-stage progressive disclosure: advertise (∼100 tokens/skill), load_skill, read_skill_resource, run_skill_script.

Microsoft. Microsoft Dataverse — Microsoft Power Platform. Microsoft product page, 2026b. URL https://www.microsoft.com/en-us/power-platform/dataverse. Accessed: 2026-09-24.

Microsoft. Microsoft Dataverse documentation. Microsoft Learn, Power Apps documentation, 2026c. URL https://learn.microsoft.com/en-us/power-apps/maker/data-platform/. Accessed: 2026-09-24.

Microsoft. What is Microsoft Dataverse? Microsoft Learn, Power Apps documentation, 2026d. URL https://learn.microsoft.com/en-us/power-apps/maker/data-platform/data-platform-intro. Accessed: 2026-09-24.

OpenAI. Openai api pricing and models documentation. https://openai.com, 2026. Accessed: 2026-09-26.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*, 2023. URL https://arxiv.org/abs/2305.15334.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, 2023a. URL https://arxiv.org/abs/2302.04761.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessı, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Edouard Grave, Armand Joulin, et al. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36, 2023b.

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. In *NeurIPS 2023 Workshop on Foundation Models for Decision Making*, 2023. URL https://arxiv.org/abs/2305.16291.

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

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48, pp. 1995–2003. PMLR, 2016. URL https://proceedings.mlr.press/v48/wangf16.html.

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

Renjun Xu and Yang Yan. Agent skills for large language models: Architecture, acquisition, security, and the path forward, 2026. URL https://arxiv.org/abs/2602.12430. Accepted at the Agent Skills 2026 Workshop.

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

Barry Zhang, Keith Lazuka, and Mahesh Murag. Equipping agents for the real world with Agent Skills. Anthropic Engineering Blog, October 2025a. URL https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills. Three-level progressive disclosure: YAML frontmatter at startup, SKILL.md body on trigger, bundled files and scripts on demand.

Barry Zhang, Keith Lazuka, and Mahesh Murag. Equipping agents for the real world with agent skills. Anthropic Engineering, October 2025b. URL https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2306.05685.

Yanzhao Zheng, Zhentao Zhang, Chao Ma, Yuanqiang Yu, Jihuan Zhu, Baohua Dong, and Hangcheng Zhu. Skillrouter: Retrieve-and-rerank skill selection for LLM agents at scale. *arXiv preprint arXiv:2603.22455*, 2026. URL https://arxiv.org/abs/2603.22455.

APPENDIX

A INSTRUCTION-DENSE EXAMPLES

Table 7 presents selected reasoned instructions for Flight Search and PowerPoint skills, reproduced from their generated example user requests. These are retrieval examples generated offline from skill content, not procedural steps or observed task outcomes.

Table 7: Example user requests for Instruction-Dense retrieval.

Skill	Reasoned instructions
Flight Search	**1.** I need to find flights from Boston to Miami on March 15th next year so I can finalize my spring break plans with friends and see what airlines have the best prices for that week.
	2. Can you check what flights are available from San Francisco to New York on a Monday in April? I'm planning a quick business trip and want to see if there are morning departures that would work with my meeting schedule.
	3. I'm planning a cross-country road trip starting from the East Coast. Can you search flights from Washington DC to San Diego on May 10th? I want to see if flying makes sense cost-wise compared to driving.
PowerPoint	**1.** Can I extract all the comments that reviewers left on my presentation slides so I can compile their feedback into one document?
	2. Can I convert my PowerPoint slides to image files so I can embed them in a document or share them as standalone graphics?
	3. I'm building a presentation from HTML content and want to use an HTML-to-PowerPoint converter that handles proper formatting and positioning. What's the workflow for that?

The Flight Search examples express price comparison, departure-time preferences, and travel alternatives. The PowerPoint examples express reviewer-comment extraction, slide-to-image conversion, and presentation creation from HTML. They illustrate user intents rather than guarantees of successful execution. Instruction-Dense embeds each request and combines query similarity to the skill's request centroid with the strongest individual match, representing both common uses and specific task contexts without generating requests at query time.

B STATE-SPACE FEATURE DETAILS

Table 8 details the four score summaries computed independently for each ranking modal in the policy state (Section 3.2.2). Within the fixed window of leading results, let t be the highest score, μ the mean, σ the population standard deviation, and m the median of the min–max-normalized scores. Let μ_5 be the mean of the first five scores (or all available scores if fewer than five), and $\epsilon > 0$ a small constant safeguarding division. Empty windows yield four zeros.

Table 8: Per-modal score summaries used as state-space features.

Summary	Computation	Meaning
Normalized mean	$\mu/(t+\epsilon)$	Average score relative to the best score
Normalized standard deviation	$\sigma/(t+\epsilon)$	Score dispersion relative to the best score
Top-score prominence	$t/(t+m+\epsilon)$	Best score relative to the median
Top-five gap	$\max\left(0, \frac{t-\mu_5}{t+\epsilon}\right)$	Separation between the best score and the top-five mean

C REWARD HYPERPARAMETERS

Table 9 gives the reward-component weights used in the reported benchmark runs. Weights are fixed within each run but may differ across domains to reflect the relative importance of precision, coverage, and action calibration. All listed runs use reward sharpness $\tau = 1$. These coefficients weight reward errors, not the modal contributions in RRF.

Table 9: Reward hyperparameters for the reported runs. Selection weights sum to one; action weights are η for cardinality error and $1 - \eta$ for modal-weight error.

	Selection error				Action error	
Benchmark	λ_F	λ_P	λ_R	λ_{rank}	η	$1-\eta$
SkillsBench (209)	0.5	0	0	0.5	0.5	0.5
Dataverse (16)	0.5	0	0	0.5	0.5	0.5
AgentSkillOS (50)	0	0.3	0.5	0.2	0.5	0.5
AgentSkillOS (500)	0	0.3	0.5	0.2	0.5	0.5
AgentSkillOS (1,000)	0	0.2	0.6	0.2	0.2	0.8
Combined (1,238)	0	0.2	0.6	0.2	0.2	0.8

D LLM JUDGE EVALUATION

We evaluated GPT-5.6-sol, GPT-5.6-terra, GPT-5.4-mini, and GPT-4.1-mini on post-execution artifacts from all 94 SkillsBench tasks. The staged judge assesses material task requirements, identifies required skills, and evaluates rank fidelity when applicable. The prompt requires evidence-grounded judgments and complete structured outputs. Ground-truth skill annotations are withheld during judging and used only for subsequent comparison.

A batch size of 15 groups up to 15 independent tasks in each batched LLM request. We conducted five runs per model, and all 20 runs evaluated all 94 tasks without unresolved failures. Judge-inferred required skills are compared with annotated skills using macro precision, recall, and F1. Table 10 reports agreement and evaluation efficiency.

Table 10: LLM-judge agreement with annotated required skills and evaluation efficiency. Values are averaged over five runs; F1 is reported as mean ± population standard deviation.

Judge	Precision	Recall	F1	Calls	Tokens	Latency (s)
GPT-5.6-sol	**0.9037**	0.7408	0.7903 ± 0.0181	19.8	818,240	854
GPT-5.6-terra	0.8827	0.7365	0.7770 ± 0.0122	22.0	1,087,053	803
GPT-5.4-mini	0.8103	0.8130	0.7805 ± 0.0119	17.4	959,848	**588**
GPT-4.1-mini	0.8419	**0.8289**	$\mathbf{0.8175 \pm 0.0056}$	**16.0**	**732,086**	727

GPT-4.1-mini achieves the highest F1 and recall, the lowest F1 variation, and the lowest call and token usage. GPT-5.6-sol achieves the highest precision, while GPT-5.4-mini has the lowest latency. Calls, tokens, and latency include retry and resume attempts.

Table 11: Selection quality on the combined ecosystem benchmark (1,238 skills).

Selector	Precision	Recall	F1	TSAR (%)
`SkillECTOR`	0.513	**0.825**	0.587	**68.1**
GPT-5.4-nano	0.540	0.631	0.544	39.4
GPT-5.6-sol	0.803	0.781	0.774	60.3
GPT-6 Astra	**0.827**	0.803	**0.798**	63.2

E COMBINED SKILL ECOSYSTEM

To examine selection under greater skill and task diversity, we combine SkillsBench, AgentSkillOS, and Dataverse into a shared benchmark of 307 tasks and 1,238 skills. Unlike the separate ecosystem evaluations, each query is ranked against the combined catalog. Table 11 reports the best-F1 results from the recall-weighted trajectory of `SkillECTOR`. Each LLM selector result corresponds to the run used in the learning-curve comparison.

`SkillECTOR` achieves the highest recall (0.825) and TSAR (68.1%), while GPT-6 Astra achieves the highest precision and F1. Its complete skill coverage exceeds GPT-6 Astra by 4.9 percentage points, showing a coverage advantage in this heterogeneous catalog, with lower precision and F1 than the larger LLM selectors. Figure 6 shows the corresponding TSAR learning curves.

F MONETARY COST

Table 12 outlines the public token pricing breakdown for GPT-6-Astra (c.f. OpenAI (2026)) that was used for computation of 'Projected Cost (USD)' of the end to end GHCP experiments in Table 5 in the main paper.

Table 12: Pricing per million tokens.

Token Type	**Default**	**Long Context**
Regular input	$10.00	$20.00
Cache reads	$1.00	$2.00
Cache writes	$12.50	$25.00
Output	$50.00	$75.00

GHCP logging gives us detailed telemetry of input/output/cache tokens as per tasks, per run levels. We used the above pricing table to compute the overall joint cost.