More Skills, Worse Agents? Skill Shadowing Degrades Performance When Expanding Skill Libraries

Hongwen Song
hongwen.song@databricks.com
Databricks

Song (Vinson) Wei
vinson.wei@databricks.com
Databricks

Abstract

Skill libraries allow LLM agents to load task-specific instructions on demand, letting non-expert users solve domain-specific tasks through natural language without knowing which skills exist or how they work. However, performance degrades as libraries grow—by up to 21% when scaling from a small set of helpful skills to a 202-skill library. In this work, we formulate this performance degradation as the pass rate drop between loading a library of known-helpful skills and the full library. Moreover, we propose to decompose the pass rate drop by conditioning on the skill(s) invocation—which skills the agent selects during a trajectory—into two effects: *skill shadowing*, where the agent selects wrong skills more often as the library expands, and *context overhead*, where the enlarged context degrades execution even when selection is correct. We derive upper bounds on both effects to characterize their magnitudes of impacts to the pass rate drop. Our empirical estimates of the effects and their upper bounds both show that the *skill shadowing* effect grows with library size and significantly contributes to the performance degradation, whereas the *context overhead* effect remains small and indistinguishable from zero. This observed asymmetry establishes that the skill selection failure, not the enlarged context, is the primary bottleneck when expanding the skill libraries.

ACM Reference Format:
Hongwen Song and Song (Vinson) Wei. 2026. More Skills, Worse Agents? Skill Shadowing Degrades Performance When Expanding Skill Libraries. In *The ACM CAIS 2026 Workshop AgentSkills (Agent Skills '26), May 26, 2026, San Jose, CA, USA.* ACM, New York, NY, USA, 9 pages.

Agent Skills '26, San Jose, CA, USA

1 Introduction

LLM agents are increasingly extended not only by adding tool calls but also by loading *skills*—self-contained packages of instructions, scripts, and reference documents that inject domain expertise into the agent's context [16]. The abstraction has rapidly become an industry standard: Anthropic shipped Agent Skills for Claude and released the specification as an open format, with partner-built skills from Atlassian, Stripe, and Figma entering a curated directory [17]; OpenAI adopted the same `SKILL.md` convention for Codex shortly after [8]. The appeal is compositional: practitioners assemble dozens of skills into a single agent, and foundational systems demonstrated that capability libraries can be composed and selected at scale through language-based interfaces—Voyager [14] incrementally builds a skill library retrieved by embedding similarity, while HuggingGPT [13] selects among thousands of expert models using their natural-language descriptions. The premise is that a user can describe a task in plain language ("check why this customer's cluster is slow") and the agent autonomously selects the right skill, without the user knowing which skills exist or how they work. In our experiments, a 202-skill library cuts this premise short: pass rate drops by 21%, averaged across both models.

Yet most prior work evaluates skills *one at a time*: how to author a skill, retrieve one, or measure its marginal contribution over a no-skill baseline [3, 9, 11]. Evidence from the more mature tool-use literature suggests this single-skill view is dangerously incomplete. Tool selection accuracy drops from above 90% with fewer than 30 candidates to 13.6% with 11,100 [1], and in-context selection "could completely fail" with numerous options [3]. Skills amplify these failure modes: they rely on natural-language descriptions with broader, more overlapping scope than typed tool schemas, and a wrong skill injects plausible but incorrect reasoning into the agent's context where it is treated as authoritative [16]—whereas a wrong tool call is more likely to produce a detectable error (type mismatch, missing parameter, API error code) due to its structured schema.

Motivations. Despite growing awareness—recent work has documented degradation under progressively realistic skill settings [7] and shown that compressing skill content can yield a "less-is-more" effect [2]. As shown in Figure 1, we evaluate on SkillsBench [6] and observe monotonic pass rate drop when we load progressively larger skill libraries compared to only loading known-helpful skills (i.e., oracle

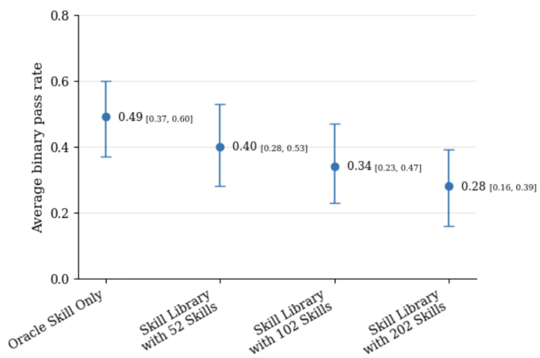

Figure 1. Pass-rate drop when expanding from oracle skills to larger libraries, averaged across 38 (task, model) pairs, with 95% percentile-bootstrap CI.

skills). Our leading hypothesis is that *certain skills in the library shadow the oracle skill.* For example:

> In the `mario-coin-counting` task, the library includes a skill named `video-frame-extraction`, whose description superficially matches the query better than the provided helpful skill. As a result, the agent invokes this wrong one in ALL 26 trajectories.

However, no prior work has formalized the library-induced degradation, nor has tried to decompose it into separable causal mechanisms through controlled experimentation. The challenge is that an agent trajectory—from receiving the task query, through skill selection and invocation, to producing a final output—is largely a black box: we observe the task outcome but not the intermediate decisions that produced it. Without observing full intermediate states, we cannot attribute degradation to specific failure mechanisms.

Contributions. We propose to use *skill invocations* as one such observable intermediate state. We formalize the trajectory structure to define invocation patterns, which enables an exact decomposition of library-induced degradation into two separable effects—*skill shadowing* and *context overhead*—with upper bounds derived on each individual effect (§3). Our experiments show that skill shadowing dominates, accounting for up to 68% of degradation and serving as the only statistically significant effect, while context overhead yields consistently reasonable but insignificant point estimates, remaining indistinguishable from noise (§4).

2 Formulation

We study *skill-augmented* agents—LLM agents that select and load task-specific instruction sets from a shared library. Unlike tool-augmented agents [10, 12], where invoking a tool executes code and returns a computed result, invoking a *skill* injects instructional text into the agent's context window; the agent then drives task execution itself using base tools, guided by the injected instructions.

2.1 Skill-Augmented Agents

Definition 2.1 (Skill)**.** A skill is a triple $S_i = (n_i, d_i, b_i)$ consisting of a name n_i, a natural-language description d_i, and a body b_i (the full instructions and examples that guide task execution). The pair (n_i, d_i) is the skill's *interface*—the only information visible to the agent at selection time. The body b_i is withheld until the skill is invoked.

Definition 2.2 (Skill library)**.** A skill library is a finite set $\mathcal{S} = \{S_1, \ldots, S_\ell\}$.

Our formulation adapts the agentic-skill tuple of Jiang et al. [5], who define a skill by its applicability condition, policy, termination condition, and callable interface. Context-injection skills delegate execution entirely to the host agent, so the applicability condition, policy, and termination condition reduce to properties of the agent rather than the skill; What remains is the callable interface (our (n_i, d_i)) and the instructional payload (our b_i).

Definition 2.3 (Agent)**.** An agent $\mathcal{A}_\theta$ is a large language model with parameters θ that, at each turn, selects an action conditioned on the current observation, prior interaction history, task q, and the skill description $\{(n_i, d_i)\}_{i=1}^{\ell}$. An action is one of: a *skill invocation* $\texttt{Skill}(n_i)$, a *base-tool call* (e.g., file read, shell command), or a *termination signal.*

Definition 2.4 (Agent context)**.** The *agent context* is the complete input the agent receives before acting:

$$C(q, \mathcal{S}) = \big(q,\ \mathcal{D}(\mathcal{S})\big), \quad \mathcal{D}(\mathcal{S}) = \{(n_i, d_i)\}_{i=1}^{\ell}. \tag{1}$$

The agent context is fully determined by the task and library, and is the only input that varies across experiments (§4).

This adapts the controller formulation of Qin et al. [10], who decompose the LLM's decision as $p_\theta(a_t \mid x_t, H_t, q)$ over perceiver output x_t, history H_t, and query q. In our setting, the perceiver output is replaced by the skill description $\mathcal{D}(\mathcal{S})$, and the history H_t is unobservable—we treat the trajectory as a black box from which only the invocation set $\mathcal{I}$ and the final output are recorded.

2.2 Formalization of the Trajectory

Definition 2.5 (Task)**.** A task is a triple $\mathcal{T} = (q, r, v)$ consisting of a query q (a natural-language description of the objective), a reference solution r (a known-correct implementation that demonstrates solvability), and a verifier v (a deterministic test suite of independently evaluable subtasks).

This follows the task structure of Li et al. [6], simplified by folding the environment into v. The agent observes only q; both r and v are withheld. Since each task is uniquely identified by its query, we index by q for brevity throughout. Given $\mathcal{T} = (q, r, v)$ and $\mathcal{S}$, a trajectory proceeds in three phases:

Phase 1: Context loading. The agent receives q and $\mathcal{D}(\mathcal{S})$. Skill bodies are withheld.

Phase 2: Skill invocation to fulfill the task query. The agent acts for a bounded number of turns (or time). At each turn, it emits an action: 2-(1) If the action is $\texttt{Skill}(n_i)$, the body b_i is appended to the agent's context (i.e., skill invocation); 2-(2) If the action is a base-tool call, the system executes it and returns the result.

The agent may invoke multiple skills during a single trajectory. Let $\mathcal{I} \subseteq \mathcal{S}$ denote the set of distinct skills invoked during the trajectory. The intermediate actions, observations, and internal context accumulated during Phase 2 are difficult to observe. We observe only the agent context $C(q, \mathcal{S})$, the invocation set $\mathcal{I}$, and the final output evaluated by v.

Phase 3: Output. The trajectory terminates when the agent signals completion or reaches the turn (or time) limit. The agent produces a final artifact (e.g., modified files, generated code) and the trajectory ends.

2.3 Output Verification

The verifier v, external to the agent trajectory, evaluates the final environment state; the agent cannot observe or interact with v. We define the task completion random variable

$$T_q = \frac{\#\{\text{passed subtasks verified by } v\}}{m} \in \left\{0,\ \tfrac{1}{m},\ \ldots,\ 1\right\}. \tag{2}$$

The *binary pass rate* is the probability that all subtasks pass:

$$p(q, \mathcal{A}_\theta, \mathcal{S}) = \Pr\left[T_q = 1 \mid q,\ \mathcal{A}_\theta,\ \mathcal{S}\right]. \tag{3}$$

The *fractional pass rate* is:

$$\bar{p}(q, \mathcal{A}_\theta, \mathcal{S}) = \mathbb{E}\left[T_q \mid q,\ \mathcal{A}_\theta,\ \mathcal{S}\right]. \tag{4}$$

When $m = 1$, $T_q \in \{0, 1\}$ and the two metrics coincide. Since the agent $\mathcal{A}_\theta$ is fixed throughout our experiments, we write $p(q, \mathcal{S})$ for brevity.

3 Library-Induced Degradation: Decomposition and Upper Bounds

We formalize the two degradation mechanisms studied in this paper. *Skill shadowing* occurs when distractor skills displace oracle skills during selection, reducing the probability that the agent invokes a correct skill despite one being available. *Context overhead* occurs when the enlarged agent context degrades execution quality even when the agent selects correctly. We define oracle and distractor skills, classify trajectories by their invocation pattern, and derive an exact decomposition of the pass-rate drop into these two effects.

Definition 3.1 (Oracle skill set)**.** For a task q and a skill-delta threshold $\tau > 0$, the oracle skill set is the subset of skills whose marginal contribution over the no-skill baseline meets or exceeds τ:

$$\mathcal{S}^\star(q) = \{S_i \in \mathcal{S} \mid p(q, \{S_i\}) - p(q, \varnothing) \geq \tau\}. \tag{5}$$

The oracle set is determined empirically: each skill is tested in isolation against q, and only those whose pass-rate improvement over the no-skill baseline meets the threshold are included. The set may contain multiple skills (when several skills independently improve performance on q) or be empty (when no single skill provides sufficient benefit). The value of τ and the empirical oracle-set sizes are reported in §4.

Definition 3.2 (Distractor skills)**.** For a task q, all non-oracle skills $\mathcal{S} \setminus \mathcal{S}^\star(q)$ are distractors. Distractors are not adversarially constructed; they are simply the skills in the library that do not meet the oracle threshold for q.

Definition 3.3 (Invocation events)**.** For a task q with $\mathcal{S}^\star(q) \neq \varnothing$, recall that a trajectory produces an invocation set $\mathcal{I}$. Every trajectory falls into one of three mutually exclusive events:

- N. **No skill invoked** ($\mathcal{I} = \varnothing$). The agent attempts the task with base tools alone, or fails to act.
- M. **Mixed invocation** ($\mathcal{I} \neq \varnothing$ and $\mathcal{I} \not\subseteq \mathcal{S}^\star(q)$). The agent invoked at least one skill, but not exclusively oracle skills—either no oracle was invoked at all, or an oracle was invoked alongside distractors.
- O. **Oracle-only invocation** ($\varnothing \neq \mathcal{I} \subseteq \mathcal{S}^\star(q)$). Every skill the agent invoked was an oracle skill.

Library-induced pass-rate drop. To quantify the cost of library expansion, we compare the pass rate when only the oracle skill set is available (selection is trivial) against the pass rate under the full library:

Definition 3.4 (Library-induced pass-rate drop)**.** For a task q with $\mathcal{S}^\star(q) \neq \varnothing$, the library-induced pass-rate drop is:

$$\Delta(q, \mathcal{S}) = p\big(q,\ \mathcal{S}^\star(q)\big)\ -\ p(q,\ \mathcal{S}). \tag{6}$$

Decomposition by invocation event. For tasks with $\mathcal{S}^\star(q) \neq \varnothing$, the three invocation events (Definition 3.3) partition the trajectory space. By the law of total probability:

$$p(q, \mathcal{S}) = \pi_{\text{N}}\, \rho_{\text{N}} + \pi_{\text{M}}\, \rho_{\text{M}} + \pi_{\text{O}}\, \rho_{\text{O}}, \tag{7}$$

where $\pi_{\text{N}}, \pi_{\text{M}}, \pi_{\text{O}}$ are the probabilities of events N, M, O given $(q, \mathcal{S})$, and $\rho_{\text{N}}, \rho_{\text{M}}, \rho_{\text{O}}$ are the corresponding conditional pass rates. The probabilities of the three events sum to one, i.e., $\pi_{\text{N}} + \pi_{\text{M}} + \pi_{\text{O}} = 1$.

The same decomposition applies when the library contains only oracle skills, i.e., $\mathcal{S} = \mathcal{S}^\star(q)$. Since no distractors are present, M is impossible, leaving only N and O. We denote the event probabilities and conditional pass rates under $\mathcal{S} = \mathcal{S}^\star(q)$ with a star:

$$p(q, \mathcal{S}^\star) = \pi^\star_{\text{N}}\, \rho^\star_{\text{N}} + \pi^\star_{\text{O}}\, \rho^\star_{\text{O}}, \tag{8}$$

where $\pi^\star_{\text{N}} + \pi^\star_{\text{O}} = 1$. Substituting Eqs. 7 and 8 into Eq. 6:

$$\Delta(q, \mathcal{S}) = \left(\pi^\star_{\text{N}}\, \rho^\star_{\text{N}} + \pi^\star_{\text{O}}\, \rho^\star_{\text{O}}\right)\ -\ \left(\pi_{\text{N}}\, \rho_{\text{N}} + \pi_{\text{M}}\, \rho_{\text{M}} + \pi_{\text{O}}\, \rho_{\text{O}}\right). \tag{9}$$

Adding and subtracting $\pi_{\mathrm{N}}^{\star}\,\rho_{\mathrm{N}} + \pi_{\mathrm{O}}^{\star}\,\rho_{\mathrm{O}}$ yields:

$$\begin{aligned}\Delta(q,\mathcal{S}) &= \underbrace{\pi_{\mathrm{N}}^{\star}\big(\rho_{\mathrm{N}}^{\star}-\rho_{\mathrm{N}}\big)+\pi_{\mathrm{O}}^{\star}\big(\rho_{\mathrm{O}}^{\star}-\rho_{\mathrm{O}}\big)}_{\Delta_{\mathrm{ctx}}} \\ &\quad+\underbrace{\big(\pi_{\mathrm{N}}^{\star}-\pi_{\mathrm{N}}\big)\,\rho_{\mathrm{N}}+\big(\pi_{\mathrm{O}}^{\star}-\pi_{\mathrm{O}}\big)\,\rho_{\mathrm{O}}-\pi_{\mathrm{M}}\,\rho_{\mathrm{M}}}_{\Delta_{\mathrm{shd}}}. \qquad (10)\end{aligned}$$

Context overhead effect (Δ_{ctx}). Event probabilities are held at their $\mathcal{S}^{\star}$ values; only the conditional pass rates change. A positive $\rho_{\mathrm{O}}^{\star} - \rho_{\mathrm{O}}$ indicates that the full-library context degrades execution even when the agent selects the same skills. The sole difference between conditions is the agent context (Definition 2.4): $\mathcal{D}(\mathcal{S}^{\star})$ contains $|\mathcal{S}^{\star}(q)|$ entries, while $\mathcal{D}(\mathcal{S})$ contains ℓ.

Skill shadowing effect (Δ_{shd}). By analogy with variable shadowing in programming, distractor skills whose descriptions overlap with oracle skills can hide the correct choice, diverting the agent toward wrong or no skills. Conditional pass rates are held at their full-library values; only the event probabilities change. Under the full library, probability mass shifts from O into M and N, reducing performance. Since $\pi_{\mathrm{N}}^{\star}+\pi_{\mathrm{O}}^{\star} = 1 = \pi_{\mathrm{N}}+\pi_{\mathrm{M}}+\pi_{\mathrm{O}}$, we have $\pi_{\mathrm{M}} = (\pi_{\mathrm{N}}^{\star}-\pi_{\mathrm{N}})+(\pi_{\mathrm{O}}^{\star}-\pi_{\mathrm{O}})$, and the skill shadowing effect simplifies to:

$$\Delta_{\mathrm{shd}} = \big(\pi_{\mathrm{N}}^{\star}-\pi_{\mathrm{N}}\big)\big(\rho_{\mathrm{N}}-\rho_{\mathrm{M}}\big)+\big(\pi_{\mathrm{O}}^{\star}-\pi_{\mathrm{O}}\big)\big(\rho_{\mathrm{O}}-\rho_{\mathrm{M}}\big). \qquad (11)$$

Assumption 1. *For each invocation event $E \in \{N, O\}$, the conditional pass rate under $\mathcal{S} = \mathcal{S}^{\star}(q)$ is at least as high as under the full library:*

$$\rho_E^{\star} \geq \rho_E. \qquad (12)$$

Lemma 1 (Upper bound on context overhead)**.** *Under Assumption 1, $\Delta_{\mathrm{ctx}} \geq 0$ and*

$$\Delta_{\mathrm{ctx}} \leq \max\!\big(\rho_N^{\star}-\rho_N,\ \rho_O^{\star}-\rho_O\big). \qquad (13)$$

Proof. Both terms in Δ_{ctx} are non-negative by Assumption 1. Since $\pi_{\mathrm{N}}^{\star} + \pi_{\mathrm{O}}^{\star} = 1$, Δ_{ctx} is a convex combination, bounded by the maximum of its components. □

Lemma 2 (Upper bound on skill shadowing)**.** *The skill shadowing effect satisfies*

$$\begin{aligned}\big|\Delta_{\mathrm{shd}}\big| &\leq \big(|\pi_N^{\star}-\pi_N| + |\pi_O^{\star}-\pi_O|\big) \;\cdot\; \max\!\big(|\rho_N-\rho_M|,\ |\rho_O-\rho_M|\big) \\ &\leq |\pi_N^{\star}-\pi_N| + |\pi_O^{\star}-\pi_O|. \qquad (14)\end{aligned}$$

Proof. Apply the triangle inequality to Eq. 11 and factor out the maximum absolute conditional pass-rate difference. The second inequality follows because $|\rho_E - \rho_{\mathrm{M}}| \leq 1$. □

Interpretation. The two bounds serve as structural diagnostics: if the full-library context barely changes execution quality within each invocation event, the bound—and therefore Δ_{ctx} itself—must be small, regardless of sample size. A small bound thus disambiguates "undetectable due to noise" from "genuinely small." The shadowing bound factors into two interpretable components: *selection instability* (how much the invocation-event distribution shifts under the full library) and *selection stakes* (how much pass rate differs across events). Shadowing can be large only when both factors are large: the agent's selection must shift *and* the shift must move mass toward lower-performing events. We evaluate these bounds numerically in Appendix B.2.

4 Experiments

We design controlled experiments to answer the remaining question: what drives the library-induced degradation?

Dataset. We evaluate on SkillsBench [6], a benchmark of 88 tasks spanning document processing, finance, scientific computing, data analysis, etc. Each task has a corresponding skill that could be helpful. We include only tasks whose corresponding skill yields at least a 4% pass-rate uplift over the no-skill baseline, giving us 38 (task, model) pairs with oracle skills (21 tasks from Haiku 4.5, and 17 from Sonnet 4.6). For each pair we evaluate under a baseline where the library contains only oracle skills ($\mathcal{S}{=}\mathcal{S}^{\star}$) and three expanded libraries of total size $|\mathcal{S}| \in \{52, 102, 202\}$. The final dataset comprises 2,545 trajectories. Further details, e.g., task selection, full trajectory counts, library construction, and bootstrap, are deferred to Appendix A.

Decomposition and scaling of the pass-rate drop. For each library size we classify trajectories into the three invocation events (Definition 3.3) and report Δ_{ctx}, Δ_{shd}, and Δ (Eq. 10) in Table 1: Both Δ_{ctx} and Δ_{shd} grow with library size, with Δ_{shd} accounting for an increasing share of Δ. The behavioral shifts underlying this scaling are visible in Table 2, which reports the fraction of trajectories in each of four invocation patterns—"oracle only" (event O), "mixed – oracle invoked" and "mixed – oracle not invoked" (sub-cases of event M), and "no skill invoked" (event N). As the library grows, the event O share collapses, with mass migrating into events N and M. Two complementary failure modes are visible: *abandonment* (mass into event N) and *wrong selection* (mass into event M). Additional shadowing examples appear in Appendix A.4 and empirical studies of invocation pattern shift can be found in Appendix B.4.

Table 1. Decomposition of Δ (Definition 3.4) into Δ_{ctx} and Δ_{shd} via Eq. 10, averaged across both models, with 95% two-stage clustered-bootstrap CIs. $\Delta_{\mathrm{ctx}} + \Delta_{\mathrm{shd}} = \Delta$ exactly. **Bold** marks CIs that exclude zero.

$\lvert\mathcal{S}\rvert$	Δ_{ctx}	Δ_{shd}	Δ
52	.06 [−.11, .22]	.03 [−.01, .07]	**.08 [.02, .15]**
102	.06 [−.11, .23]	**.08 [.02, .15]**	**.14 [.09, .19]**
202	.07 [−.13, .25]	**.14 [.06, .26]**	**.21 [.15, .27]**

Which effect dominates the drop? Δ_{shd} is the only component whose CI excludes zero and accounts for the majority of Δ at the largest library. Δ_{ctx} does not separate from zero at any library size, although its point estimate is consistently positive. Within event o—where the agent selected correctly—the binary pass rate still falls as the library grows (see Tables 7 and 8 in Appendix B.3), suggesting context overhead is real but our sample size is too small to confirm it. Based on the empirical observation, we conclude that shadowing is the dominant effect of the drop, and the context-overhead point estimates as suggestive only. The theoretical bounds of Lemmas 1 and 2 corroborate this conclusion: the context overhead bound remains small and indistinguishable from zero, while the shadowing bound grows sharply and is significantly positive throughout; see Appendix B.2 for complete details.

Table 2. Trajectory-share distribution across the four invocation patterns (refinement of Definition 3.3's partition; see Appendix B.3 for details).

	oracle only	mixed – oracle invoked	mixed – oracle not invoked	no skill invoked	n
$\mathcal{S} = \mathcal{S}^\star$	88.0%	0.0%	0.0%	12.0%	552
$\lvert\mathcal{S}\rvert = 52$	80.3%	0.9%	2.8%	16.1%	542
$\lvert\mathcal{S}\rvert = 102$	64.4%	1.4%	5.9%	28.3%	988
$\lvert\mathcal{S}\rvert = 202$	52.6%	1.6%	7.3%	38.5%	494

5 Discussion

Our results point to skill shadowing as the primary bottleneck when scaling a skill library: once semantically similar skills compete for selection, accuracy degrades regardless of the underlying model. A natural next step is to attack this bottleneck directly through retrieval-based pre-filtering [1, 15], description disambiguation [2], learned routing [4], or other techniques. In addition, our study covers 202 skills on two models from a single provider, a scale at which context overhead remains modest. Whether these patterns—and the mitigations above—hold as libraries grow by an order of magnitude or span diverse architectures remains an open question.

Acknowledgments

We thank Zubair Anwar, Juanyan Li and Martin Valdez-Vivas for their valuable feedback and discussions throughout this work. Moreover, we truly appreciate the Data Science Leaders Sam Shah, Feng Pan, Zubair Anwar, and Divy Menghani for their support.

References

[1] Tiantian Gan and Qiyao Sun. 2025. RAG-MCP: Mitigating Prompt Bloat in LLM Tool Selection via Retrieval-Augmented Generation. *arXiv preprint arXiv:2505.03275* (2025).

[2] Yudong Gao, Zongjie Li, Zimo Ji, Pingchuan Ma, Shuai Wang, et al. 2026. SkillReducer: Optimizing LLM Agent Skills for Token Efficiency. *arXiv preprint arXiv:2603.29919* (2026).

[3] Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. 2023. ToolkenGPT: Augmenting frozen language models with massive tools via tool embeddings. *Advances in neural information processing systems* 36 (2023), 45870–45894.

[4] Jingyi Jia and Qinbin Li. 2026. AutoTool: Efficient tool selection for large language model agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 40. 31265–31273.

[5] Yanna Jiang, Delong Li, Haiyu Deng, Baihe Ma, Xu Wang, Qin Wang, and Guangsheng Yu. 2026. SoK: Agentic Skills—Beyond Tool Use in LLM Agents. *arXiv preprint arXiv:2602.20867* (2026).

[6] Xiangyi Li, Wenbo Chen, Yimin Liu, Shenghan Zheng, Xiaokun Chen, Yifeng He, Yubo Li, Bingran You, Haotian Shen, Jiankai Sun, et al. 2026. SkillsBench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670* (2026).

[7] Yujian Liu, Jiabao Ji, Li An, Tommi Jaakkola, Yang Zhang, and Shiyu Chang. 2026. How Well Do Agentic Skills Work in the Wild: Benchmarking LLM Skill Usage in Realistic Settings. *arXiv preprint arXiv:2604.04323* (2026).

[8] OpenAI. 2025. Agent Skills – Codex. https://developers.openai.com/codex/skills. OpenAI adopted the same SKILL.md format; skills available in Codex CLI, IDE extension, and Codex app.

[9] Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2024. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems* 37 (2024), 126544–126565.

[10] Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Xuanhe Zhou, Yufei Huang, Chaojun Xiao, et al. 2024. Tool learning with foundation models. *Comput. Surveys* 57, 4 (2024), 1–40.

[11] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2024. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. In *The Twelfth International Conference on Learning Representations*.

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

[13] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. HuggingGPT: Solving AI tasks with ChatGPT and its friends in Hugging Face. *Advances in Neural Information Processing Systems* 36 (2023), 38154–38180.

[14] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An Open-Ended Embodied Agent with Large Language Models. *Transactions on Machine Learning Research* (2023).

[15] Renxi Wang, Xudong Han, Lei Ji, Shu Wang, Timothy Baldwin, and Haonan Li. 2025. ToolGen: Unified Tool Retrieval and Calling via Generation. In *The Thirteenth International Conference on Learning Representations*.

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

[17] Barry Zhang, Keith Lazuka, and Mahesh Murag. 2025. Equipping agents for the real world with agent skills. *Anthropic Engineering Blog* (2025).

Table 3. Frequent shadowing pairs at $|\mathcal{S}|$=202 (494 trajectories across the 38 matched (task, model) pairs). $\#_{\text{shd traj}}$ is the count of trajectories in which the agent first-picks the distractor skill.

Primary task	Oracle bundle	distractor skill	Origin task(s) of distractor skill	$\#_{\text{shd traj}}$
`mario-coin-counting`	`ffmpeg`, `image_editing`, `object_counter`	`video-frame-extraction`	`jpg-ocr-stat`, `pedestrian-traffic-counting`	51
`trend-anomaly-causal-inference`	`data_cleaning`, `did_causal_analysis`, `feature_engineering`, `time_series_anomaly_detection`	`senior-data-scientist`	`powerlifting-coef-calc`	37
`manufacturing-fjsp-optimization`	`fjsp-baseline-repair-with-downtime-and-policy`	`reflow_machine_maintenance_guidance`	`manufacturing-equipment-maintenance`	14
`econ-detrending-correlation`	`timeseries-detrending`	`csv-processing`	`adaptive-cruise-control`	4
`energy-market-pricing`	`dc-power-flow`, `economic-dispatch`, `locational-marginal-prices`, `power-flow-data`	`casadi-ipopt-nlp`	`energy-ac-optimal-power-flow`	3

Table 4. Library-induced pass-rate drop Δ (Definition 3.4) by model and library size, with 95% percentile-bootstrap CI. **Bold** marks CIs excluding zero.

$\lvert\mathcal{S}\rvert$	Haiku 4.5	Sonnet 4.6
52	**.10 [.03, .18]**	.06 [−.03, .15]
102	**.22 [.16, .29]**	.05 [−.03, .13]
202	**.26 [.20, .33]**	**.15 [.06, .24]**

A Experiment Configurations

Definition 3.1 defines the oracle set at the individual-skill level, but in practice skills are often bundled together: multiple skills may jointly serve as the oracle set for a given task. In Table 3, `trend-anomaly-causal-inference` is a good example.

In SkillsBench [6], we have *skill bundle* (or just bundle for brevity) authored by task creators—they are hand-crafted set of skills that the task author considers helpful for solving the given task. We call this set the task's *authored primary skill bundle*, or just *authored (skill) bundle* for brevity. Bundles range from 1 to 7 skills (median 2, mean 2.7); they are provided in the benchmark, not by this work.

A.1 Tasks included in our experiments

We only include a (task, model) pair into our experiment if every individual skill in the authored bundle clears the empirical threshold $\tau = 0.04$ per our Definition 3.1, so that we can "break the bundle structure" and ensure each individual skill helps solve the task. The rationale is simple: if the skill does not help, one probably could not observe performance degradation even with more skills in the library. This filtering leaves us 38 (task, model) pairs: 21 Haiku and 17 Sonnet. From now on, the authored bundle serves as the oracle skill set $\mathcal{S}^\star(q)$, and we use "oracle skill set" and "oracle bundle" interchangeably below. Our experiments consist of 2,545 trajectories: with 303/297/544/273 from Haiku and 233/233/441/221 from Sonnet for $\mathcal{S} = \mathcal{S}^\star$, and $|\mathcal{S}| \in \{52, 102, 202\}$, respectively.

A.2 Library construction

The largest library ($|\mathcal{S}|$=202) is the union of all 88 authored bundles after byte-identical de-duplication and numeric-suffix disambiguation for same-name skills with divergent bodies.

A.3 Bootstrap

Confidence intervals come from 95% percentile of the bootstrap distribution. For the decomposition (Eq. 10): at each bootstrap trial, we first resample (task, model) pairs with replacement, then resample trajectories within each sampled pair.

A.4 Examples with skill shadowing effect

Table 3 lists the most frequent empirical examples with skill shadowing effect at $|\mathcal{S}|$=202, ranked by the number of trajectories in which the distractor skill is the agent's first pick. The first row is the `mario-coin-counting` example in our introduction §1.

B Additional Experiments

B.1 Per-model pass rate drops

Table 4 presents the overall pass rate drops with a per-model breakdown, where we can observe both models degrade as

Table 5. Per-model decomposition of Δ (Definition 3.4) into Δ_{ctx} and Δ_{shd} (Eq. 10), with 95% joint two-stage clustered-bootstrap CIs (2,000 trials, random seed is 42). $\Delta_{\text{ctx}} + \Delta_{\text{shd}} = \Delta$ on every point estimate. **Bold** marks CIs excluding zero. Pooled estimates appear in Table 1.

View	$\lvert\mathcal{S}\rvert$	Δ_{ctx}	Δ_{shd}	Δ
Haiku 4.5	52	.09 [−.10, .27]	.02 [−.04, .07]	.10 [−.08, .28]
	102	.15 [−.04, .34]	.07 [.00, .16]	**.22 [.05, .39]**
	202	.09 [−.18, .36]	**.17 [.02, .37]**	**.26 [.09, .43]**
Sonnet 4.6	52	.05 [−.22, .33]	.01 [−.07, .10]	.06 [−.21, .33]
	102	.03 [−.23, .30]	.02 [−.06, .10]	.05 [−.21, .31]
	202	.11 [−.16, .40]	.04 [−.04, .15]	.15 [−.11, .41]

Table 6. Upper bounds $\Delta^{\text{sup}}_{\text{ctx}}$ and $\Delta^{\text{sup}}_{\text{shd}}$ from Lemmas 1 and 2 (tighter form), with 95% joint two-stage clustered-bootstrap CIs from the same iterations as Table 5 (2,000 trials, random seed is 42). **Bold** marks CIs excluding zero.

	Haiku 4.5		Sonnet 4.6		Pooled	
$\lvert\mathcal{S}\rvert$	$\Delta^{\text{sup}}_{\text{ctx}}$	$\Delta^{\text{sup}}_{\text{shd}}$	$\Delta^{\text{sup}}_{\text{ctx}}$	$\Delta^{\text{sup}}_{\text{shd}}$	$\Delta^{\text{sup}}_{\text{ctx}}$	$\Delta^{\text{sup}}_{\text{shd}}$
52	.10 [−.06, .36]	**.04 [.01, .19]**	.06 [−.20, .33]	.16 [.00, .44]	.08 [−.08, .29]	**.06 [.01, .18]**
102	.17 [.00, .41]	**.09 [.02, .32]**	.04 [−.21, .30]	**.17 [.01, .43]**	.07 [−.08, .31]	**.14 [.04, .29]**
202	.10 [−.04, .43]	**.35 [.08, .74]**	.12 [−.14, .41]	**.15 [.01, .37]**	.07 [−.06, .34]	**.24 [.09, .46]**

the library grows. Haiku's drop separates from zero already at $\lvert\mathcal{S}\rvert$=52; Sonnet's only at $\lvert\mathcal{S}\rvert$=202.

Table 5 presents the estimates on both effects with a per-model breakdown: For Haiku, Δ_{shd} dominates and its CI excludes zero at the largest library, whereas for Sonnet, Δ_{ctx} matches or exceeds Δ_{shd} at every size, but we do not have necessary statistical power to distinguish from zero.

B.2 Theoretical upper bounds

We substitute the empirical $(\pi^\star_E, \rho^\star_E, \pi_E, \rho_E)$ from the decomposition into the right-hand sides of Lemma 1 (Eq. 13) and Lemma 2 (tighter form of Eq. 14). CIs are obtained by recomputing each bound on every bootstrap trial and taking the proper percentiles.

Table 6 reports the two upper bounds, where we denote $\Delta^{\text{sup}}_{\text{ctx}} = \max(\rho^\star_{\text{N}} - \rho_{\text{N}},\ \rho^\star_{\text{O}} - \rho_{\text{O}})$, and $\Delta^{\text{sup}}_{\text{shd}} = \lvert\pi^\star_{\text{N}} - \pi_{\text{N}}\rvert + \lvert\pi^\star_{\text{O}} - \pi_{\text{O}}\rvert$. We observe $\Delta^{\text{sup}}_{\text{ctx}}$ remains small across all conditions, and no CI separates from zero, confirming our finding that context overhead is not distinguishable from noise. $\Delta^{\text{sup}}_{\text{shd}}$ grows sharply with library size and excludes zero under nearly all scenarios.

B.3 Pass rates for different invocation events

A finer partition of the invocation events. Definition 3.3's mixed event M admits two types of events: the agent invoked at least one oracle skill alongside distractors, or it invoked only distractors. Combined with N and O, this gives a four-way partition:

- "oracle only" (O);
- "mixed – oracle invoked";
- "mixed – oracle not invoked";
- "no skill invoked" (N).

We report the binary pass rate for all four types of invocation events in Table 7 and the fractional pass rate in Table 8. CIs are Wilson 95% for binary and 95% percentile bootstrap for fractional.

- **Context overhead is visible within the oracle-only bucket.** For "oracle only" (event O)—trajectories where the agent invoked only oracle skills—the binary pass rate falls monotonically with library size on both models. This is the empirical signature of context overhead: the same invocation pattern produces a lower pass rate under a larger in-context library.
- **Fractional pass rate falls less than binary.** Within "oracle only", the fractional pass rate drops by less than the binary pass rate at every library size (e.g. pooled $\bar{p}$: .87 → .80 vs. pooled p: .52 → .45 at $\lvert\mathcal{S}\rvert$=202). Context overhead removes whole-task correctness more than partial progress.
- **Mixed and no-skill buckets are sample-limited.** The two "mixed" buckets and "no skill invoked" draw on small samples in many conditions; their high-variance pass rates should be read alongside the cell counts in Table 10.

B.4 Invocation event distributions

We call π_{M} (Definition 3.3) the empirical shadowing rate: the fraction of trajectories in which the agent invokes at least one distractor skill.

Table 9 reports π_{M} (with Wilson 95% CIs) on the selected (task, model) pairs, and we find that Sonnet accumulates more wrong-selection mass than Haiku at every library size;

Table 7. Binary pass rates for different invocation events and library sizes, with Wilson 95% CIs. Cells with fewer than 5 valid trajectories are "—".

Invocation pattern	View	p ($\mathcal{S}^\star$)	p ($\lvert\mathcal{S}\rvert$=52)	p ($\lvert\mathcal{S}\rvert$=102)	p ($\lvert\mathcal{S}\rvert$=202)
oracle only	Haiku 4.5	.43 [.37, .50]	.33 [.28, .40]	.26 [.21, .32]	.34 [.24, .45]
	Sonnet 4.6	.62 [.56, .68]	.57 [.50, .63]	.58 [.53, .63]	.51 [.43, .58]
	Pooled	.52 [.48, .57]	.44 [.40, .49]	.45 [.41, .49]	.45 [.39, .52]
mixed – oracle invoked	Haiku 4.5	—	—	.33 [.12, .65]	—
	Sonnet 4.6	—	.00 [.00, .43]	.20 [.04, .62]	.13 [.02, .47]
	Pooled	—	.00 [.00, .43]	.29 [.12, .55]	.13 [.02, .47]
mixed – oracle not invoked	Haiku 4.5	—	.00 [.00, .35]	.05 [.01, .22]	.00 [.00, .23]
	Sonnet 4.6	—	.00 [.00, .32]	.14 [.06, .29]	.13 [.05, .32]
	Pooled	—	.00 [.00, .20]	.10 [.05, .21]	.08 [.03, .22]
no skill invoked	Haiku 4.5	.12 [.06, .24]	.11 [.06, .22]	.07 [.04, .11]	.03 [.02, .07]
	Sonnet 4.6	—	1.00 [.77, 1.00]	1.00 [.87, 1.00]	.80 [.49, .94]
	Pooled	.12 [.06, .24]	.27 [.18, .38]	.16 [.12, .20]	.07 [.04, .12]

Table 8. Fractional pass rate by invocation event type and library size, with 95% percentile-bootstrap CIs. Cells with fewer than 5 valid trajectories are "—".

Invocation pattern	View	$\bar{p}$ ($\mathcal{S}^\star$)	$\bar{p}$ ($\lvert\mathcal{S}\rvert$=52)	$\bar{p}$ ($\lvert\mathcal{S}\rvert$=102)	$\bar{p}$ ($\lvert\mathcal{S}\rvert$=202)
oracle only	Haiku 4.5	.83 [.81, .86]	.77 [.73, .80]	.77 [.73, .80]	.80 [.73, .85]
	Sonnet 4.6	.91 [.88, .93]	.89 [.87, .92]	.91 [.89, .93]	.80 [.74, .84]
	Pooled	.87 [.85, .89]	.83 [.80, .85]	.85 [.83, .86]	.80 [.75, .83]
mixed – oracle invoked	Haiku 4.5	—	—	.92 [.87, .98]	—
	Sonnet 4.6	—	—	—	.89 [.81, .94]
	Pooled	—	—	.91 [.85, .97]	.89 [.81, .94]
mixed – oracle not invoked	Haiku 4.5	—	—	.70 [.60, .79]	—
	Sonnet 4.6	—	—	.85 [.79, .90]	.64 [.40, .87]
	Pooled	—	—	.80 [.75, .85]	.64 [.40, .87]
no skill invoked	Haiku 4.5	.48 [.36, .59]	.54 [.45, .64]	.45 [.40, .49]	.42 [.37, .48]
	Sonnet 4.6	—	1.00 [1.00, 1.00]	1.00 [1.00, 1.00]	.80 [.50, 1.00]
	Pooled	.47 [.36, .59]	.62 [.53, .71]	.50 [.45, .55]	.44 [.39, .50]

Haiku's π_{M} stays low because its mass shifts to "no skill invoked" events instead of selecting wrong skills as shown in Table 10 later on.

Table 9. Empirical shadowing rate π_{M} (Definition 3.3) by library size, with Wilson 95% CIs. $\mathcal{S} = \mathcal{S}^\star$ row is omitted because $\pi_{\mathrm{M}} = 0$ by construction.

$\lvert\mathcal{S}\rvert$	Haiku 4.5	Sonnet 4.6	Pooled
52	.02 [.01, .05]	.06 [.03, .09]	.04 [.02, .06]
102	.06 [.04, .08]	.09 [.07, .12]	.07 [.06, .09]
202	.05 [.03, .08]	.14 [.10, .19]	.09 [.07, .12]

Table 10 reports the per-bucket trajectory shares. The two models exhibit qualitatively different transitions as the library grows. On Haiku, "oracle only" (event O) collapses and the displaced mass goes almost entirely into "no skill invoked" (event N); the two mixed buckets stay small at every size. Haiku's failure mode under expansion is therefore *abandonment*: rather than picking the wrong skill, the agent picks no skill. On Sonnet, event O declines more gently, event N stays small, and the displaced mass goes into the two "mixed" sub-cases of event M. Sonnet's failure mode is *wrong selection*: the agent keeps invoking skills, but increasingly the wrong ones. This per-model contrast underlies the decomposition pattern in Table 1.

Table 10. Trajectory-share distribution across the four invocation patterns (refinement of Definition 3.3's N/M/O partition), by library size.

	$\lvert\mathcal{S}\rvert$	oracle only	mixed — oracle invoked	mixed — oracle not invoked	no skill invoked	n
Haiku 4.5	$\mathcal{S}^\star$	80.9%	0.0%	0.0%	19.1%	314
	52	74.5%	0.0%	2.3%	23.2%	306
	102	47.8%	1.6%	4.0%	46.5%	546
	202	29.3%	0.0%	4.8%	65.9%	273
Sonnet 4.6	$\mathcal{S}^\star$	97.5%	0.0%	0.0%	2.5%	238
	52	87.7%	2.1%	3.4%	6.8%	236
	102	84.8%	1.1%	8.1%	5.9%	442
	202	81.4%	3.6%	10.4%	4.5%	221
Pooled	$\mathcal{S}^\star$	88.0%	0.0%	0.0%	12.0%	552
	52	80.3%	0.9%	2.8%	16.1%	542
	102	64.4%	1.4%	5.9%	28.3%	988
	202	52.6%	1.6%	7.3%	38.5%	494