MEASURING THE MARGINAL UTILITY OF SKILLS AT THE EXECUTION STATE

Anonymous authors
Paper under double-blind review

ABSTRACT

Skill libraries package procedural knowledge for agents, but retrieval ranks relevance rather than testing execution. We define the marginal utility $\Delta V(s_t, k)$ of a fixed package-plus-announcement protocol at execution state s_t as its change in expected verified reward. At each checkpoint we replay one realised prefix, apply a protocol arm, independently continue it, and compare verifier rewards. Across 22 screened realised states from eleven SkillsBench tasks under one model and agent (516 runs), the health-gated own-package contrast was +0.147. Conditional on these fixed states, a runs-only 95% interval was [+0.082, +0.213] and within-state randomization gave $p < 0.001$; an 11-cluster t interval crossed zero [-0.020, +0.315] ($p = 0.079$), making cross-task mean inference method-sensitive. Utility did not separate checkpoint rules, but uptake differed by +39.9 points between first-write-rule and three-fifths checkpoints; two no-write fallbacks and one reversed pair limit that label. Only two tasks had a foreign top-1. The contribution is a signed, state-conditioned protocol and auditable finite-sample map, not a router.

1 INTRODUCTION

Coding agents can now load *skills* — packaged procedural knowledge: a `SKILL.md`, conventions, often scripts — at run time, and a benchmark exists on which their effect can be scored (BenchFlow AI, 2026). The question of *which* skills to load has been posed as retrieval: rank a library by relevance to the task description and inject the top few. Done well this works on average; the strongest such router recovers a large share of the gap between an agent with no skills and one handed the gold ones on 75 SkillsBench-derived tasks (Zheng et al., 2026).

Two observations, neither ours, say that relevance is not the quantity being paid for. First, the benchmark's own authors report that 16 of its 84 evaluated tasks are made *worse* by the skills curated for them, the worst by 39.3 points (BenchFlow AI, 2026). A relevance score is non-negative and induces only an order; it has no place for harm. Second, inside the router paper the ordering by ranking quality and the ordering by execution success come apart on some agents: a retriever with 74.0% Hit@1 completes fewer tasks than a markedly weaker one, and lengthening the shortlist lowers success where it should raise it (Section 2). At three trials per task that is noise-level, and we treat it as such — a reason to measure, not a result.

This paper measures. The *marginal utility* $\Delta V(s_t, k)$ of a skill k at an execution state s_t — the agent's context and sandbox after t exchanges — is the change in expected verified reward that injecting k there causes under a fixed package-plus-announcement protocol, with the empty injection an ordinary arm so that ΔV is free to be negative (Section 3). We estimate it by *forking a live rollout*: a baseline is recorded exchange by exchange; at a checkpoint its prefix is replayed byte for byte into fresh sandboxes, one per arm; each arm receives a different injection — nothing, the task's own package, a package written for an unrelated task, the single package a retriever ranks first — and is run to completion under the benchmark's own verifier, repeatedly. The difference in mean reward is ΔV. The contrast conditions on one shared realised prefix; live continuations are independent, not outcome-paired, and the uncertainty does not cover alternative baseline trajectories.

Figure 1 summarizes the fork, its interventions and the verified contrast.

Across 22 screened realised task–state cells, the health-gated own-package protocol contrast was +0.147; conditional on those fixed states, a runs-only 95% interval was [+0.082, +0.213] and within-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

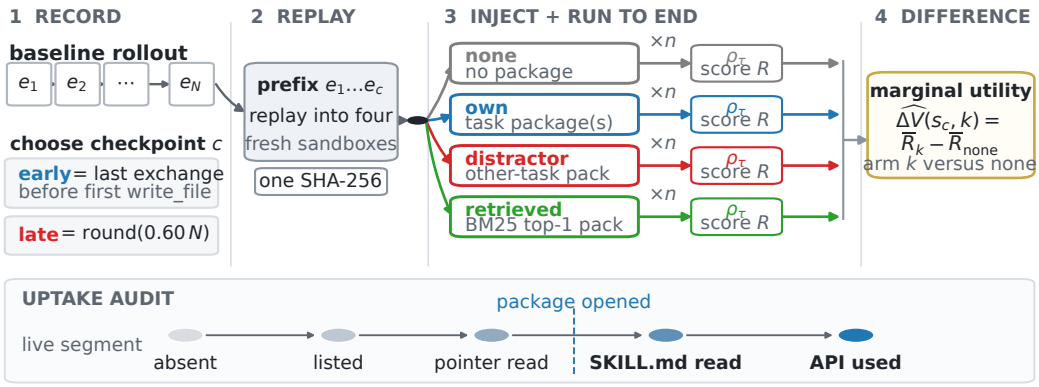


Figure 1: Forked-arm measurement of skill utility. A recorded rollout defines an execution state; its byte-identical prefix is replayed into fresh sandboxes, the arms receive different package-plus-announcement protocols, and independent continuations are scored by the task verifier. Mean reward against the uninjected arm estimates $\Delta V(s_c, k)$; an uptake ladder records whether the agent opened or used the package.

state randomization gave $p < 0.001$. Cross-task mean inference is less certain: an 11-cluster t interval was [-0.020, +0.315] ($p = 0.079$). Reward did not separate the two checkpoint rules. Uptake did: 84.1% versus 44.1%, with the direction unchanged after removing one reversed pair and both no-write fallbacks. Ranking against utility was only weakly probed: just two tasks had a foreign top-ranked package.

Our contributions are:

1. a signed, state-conditioned utility estimand for procedural knowledge;
2. a byte-audited fork-and-replay instrument that measures it under a fixed package-plus-announcement protocol; and
3. a 516-run finite-sample map with clustered sensitivity analyses and a machine-checked evidence ledger.

The estimator itself is standard counterfactual credit assignment (Meng, 2026), and abstention is standard in adaptive retrieval. Our claim is their use together at an agent's execution state. The scope is eleven adaptively screened tasks (two analysed first, five added next and four later), one model, one agent and one benchmark. This is not a task-population estimate; the map supplies labels for future routing, not evidence that a router already improves reward.

2 Related Work

Agent skills and skill routing. SkillsBench (BenchFlow AI, 2026) made the effect of skills — packaged procedural knowledge a coding agent loads into its context — measurable: 87 container tasks scored by verifiers, run with and without the skills curated for them. Skill routing is then posed as retrieval: SkillRet (Cho et al., 2026) scores ranking over 17,810 skills with Hit@K and nDCG, Wang et al. (2026) supervise a retriever with *compatibility* rather than topical match, and Su et al. (2026) scale to 26,262 skills with end-task execution but a static pipeline. **SkillRouter (Zheng et al., 2026) is the closest prior work and we do not claim its position**: it retrieves and reranks over roughly 80K skills and evaluates end to end on 75 SkillsBench-derived tasks, where success rises from 14.89% without skills to 27.56% at top-1 against 32.67% with oracle skills. Its Appendix M is where ranking and execution come apart: on `glm-5` its own retriever at 74.0% Hit@1 completes 23.11% of tasks while a markedly weaker Qwen3 baseline completes 24.89%, and widening that baseline's shortlist to top-10 moves it *down*, to 20.89%. At three trials over 75 tasks that is within noise, and we use it as the reason to measure, not as a result. The benchmark's

authors add that 16 of its 84 evaluated tasks are made *worse* by their own skills, the worst by 39.3 points; a ranking objective, whose scores are non-negative, has no way to express this.

From relevance to utility, and deciding not to retrieve. That a retriever should optimise downstream usefulness is common ground (Zhang et al., 2026); the utility signal comes from an LLM annotator (Zhang et al., 2025), the perplexity drop a passage induces (Sandhu et al., 2026), process supervision on query rewriting (Kim et al., 2025) or utility judgements iterated at inference time (Zhang et al., 2024) — always a *proxy*, because the true signal is expensive. That abstaining should be an available action is equally established: SKR (Wang et al., 2023), Self-RAG (Asai et al., 2023), Adaptive-RAG (Jeong et al., 2024), FLARE (Jiang et al., 2023), DRAGIN (Su et al., 2024) and the comparisons of Moskvoretskii et al. (2025) and Marina et al. (2025) all *predict*, from the query or a partial answer, whether retrieval will help. None *measures* the difference by running both, on a unit — a multi-turn execution inside a container, scored only after a verifier runs — or decides from an execution state that already holds the agent's tool calls, outputs and failures.

Counterfactual credit assignment for agents. Meng (2026) intervene on an agent trajectory — delete a step, substitute evidence, perturb a tool output — resume from a frozen reference policy and difference the outcomes; structurally that is our estimator. Dong et al. (2026), Bonagiri et al. (2026) and Qian et al. (2025) correct credit, explain failures and design rewards with related counterfactuals. All of them intervene retrospectively on what the agent has already produced; we intervene prospectively on an *external capability not yet supplied*, and a Meng (2026)-style attribution is a baseline we owe the reader. Toolformer (Schick et al., 2023), Gorilla (Patil et al., 2023) and ToolLLM (Qin et al., 2023) established that models can find and call tools; whether a tool should have been offered at all is downstream of that.

Position. What we claim is the conjunction: skill utility *measured* rather than predicted, at the execution state where the skill would be consumed rather than at the query, for procedural knowledge whose failure mode is a misleading process rather than a wrong fact.

3 METHOD

This section defines the quantity the paper measures, gives the estimator and what it assumes, and restates the ranking-versus-utility question in terms of the quantity. Operational detail — how checkpoints are chosen, how packages are injected, what counts as a healthy run — is in Section 4.

3.1 THE MARGINAL UTILITY OF A SKILL AT A STATE

A task τ consists of an instruction, a sandbox and a verifier ρ_τ that scores the sandbox's final state in $[0, 1]$. An agent runs it as a sequence of *exchanges*, one model request and response each, the response's tool calls being executed in the sandbox in between. After t exchanges the agent is at an *execution state* s_t: the context the model will be prompted with next, together with the state of the sandbox. A *skill* k is a package — a directory with a `SKILL.md` and optional scripts and references — drawn from a library $\mathcal{K}$. An *injection* $\iota(s_t, k)$ is a fixed package-plus-announcement protocol: it copies k into the agent's addressable workspace, adds a pointer and tells the agent to inspect it. The empty injection $\varnothing$ does none of these; there is no sham announcement. Thus the estimand is the protocol effect, not the effect of procedural content alone.

Continuing the agent from a state s to completion is a random experiment — the model samples — whose outcome is the verifier's score. The *value* of a state is its expected score, and the *marginal utility* of k at s_t is the change in value the injection causes:

$$V(s) = \mathbb{E}[\rho_\tau(\text{final sandbox}) \mid \text{continue from } s]\,, \qquad \Delta V(s_t, k) = V(\iota(s_t, k)) - V(\iota(s_t, \varnothing))\,. \tag{1}$$

This expectation is conditional on the one realised state and the fixed protocol; there is no second baseline rollout from which to estimate across-prefix variation. For a binary verifier this is the change in the probability that the task is solved. Three properties follow from the definition and shape the design.

It is signed. $\Delta V \in [-1, 1]$, and nothing in equation 1 favours the positive half: a skill that sends the agent down a wrong path, or whose reading consumes the budget the task needed, has $\Delta V < 0$. A

relevance score $r(k \mid \tau)$ is non-negative and induces only an order among skills; it has no place for harm, which is what the benchmark's own report of tasks made worse by the skills curated for them describes (BenchFlow AI, 2026).

It is a function of the state, not of the pair (τ, k). Two mechanisms make the same skill worth different amounts at different points of one rollout. Headroom: by equation 1, $\Delta V(s_t, k) \leq 1 - V(s_t)$, so once the agent has done most of the work no injection can be worth much, and a cell whose baseline already sits at the verifier's ceiling is uninformative about k by construction. Uptake: whether the agent opens a package it is offered depends on where in the task it is — an agent still exploring reads what it is given, an agent committed to an implementation does not (Section 5.1) — and an unread skill acts only through the announcement that it exists. Relevance is scored from the instruction alone and is the same at every t; it can represent neither mechanism.

It is relative to a protocol. The intervention in equation 1 is one specific way of making k available: where the files go, what the agent is told, and when. Change the protocol and the estimand changes, because uptake changes with it. We fixed one protocol before the first treated run (Section 4); every ΔV in this paper is a ΔV under it.

3.2 Estimation by forking a live rollout

Fork and difference. The two values in equation 1 are expectations over continuations from the *same* state, so we realise that state once and continue from it many times. A baseline rollout is recorded exchange by exchange. At a checkpoint c the first c exchanges are replayed verbatim into a fresh sandbox, which reconstructs s_c; the injection for arm a is applied; the model generates from exchange $c + 1$ until the agent stops or a clock expires; and the task's verifier scores the sandbox. Each arm $a \in \{\varnothing, \text{own}, \text{distractor}, \text{retrieved}\}$ is continued n times independently, giving rewards $R_1^a, \ldots, R_n^a$, and

$$\widehat{\Delta V}(s_c, a) \;=\; \frac{1}{n}\sum_{i=1}^{n} R_i^a \;-\; \frac{1}{n}\sum_{i=1}^{n} R_i^{\varnothing}. \tag{2}$$

The design is blocked on one realised fork state: every arm of a cell shares the prefix byte for byte, so variation before c does not enter equation 2. The continuations are independent rather than observation-paired. Repeats average sampling in the live segment; they do not measure variation across alternative baseline prefixes.

What it assumes. Equation 2 is unbiased for equation 1 under three conditions, each made checkable rather than assumed. (A1) *Fidelity*: continuing from the replayed prefix is continuing from s_c. This is a property of the harness, tested directly by replaying a recording to its end and checking that reward, tool calls and verifier output are reproduced (Section 4, *Fidelity of the fork*). (A2) *Exchangeability*: the arms of a cell differ in their assigned protocol and nothing else — same prefix, model, clocks and image, launched in an order that cycles through the arms repeat by repeat, so that no arm is run as a block against a different hour of the gateway. The shared prefix is verified by hash; the rest is fixed by the plan before any run starts. (A3) *The outcome is not ours*: reward is what the task's verifier returns, and a run is declared healthy or not by criteria written before the first treated run, so that a run cut short by a clock is not read as a skill that did not help.

Assignment and uptake. Among runs that pass the fixed health gate, every continuation counts under its assigned arm whether or not the agent read the package. This is the intervention a practitioner controls — injecting k at s_t, not making the agent read it. One assigned continuation in this batch timed out and is excluded by that gate, so the reported contrast is not strict intention to treat over every launched run; Section 5 gives a worst/observed-reward sensitivity bound. With $u \in \{0, 1\}$ for whether the package was read, $\pi = P(u = 1)$ its uptake rate and $\Delta V_{u=1}$, $\Delta V_{u=0}$ the effects among continuations that did and did not read it, $\Delta V = \pi\, \Delta V_{u=1} + (1 - \pi)\, \Delta V_{u=0}$. The treatment-on-treated contrast estimates $\Delta V_{u=1}$ only if $\Delta V_{u=0} = 0$, which our protocol does not guarantee — the announcement is part of the intervention — and uptake is an outcome of the live segment, so conditioning on it conditions on a post-treatment variable. We report that contrast as a description of mechanism and draw no causal claim from it; uptake is graded on a ladder fixed in advance (Section 4).

Uncertainty and budget. Rewards lie in $[0, 1]$, so the standard error of equation 2 is at most $1/\sqrt{2n}$: the pilot's six repeats per arm can sign an effect of the order of the outcome's range and cannot resolve a small one. We report percentile bootstrap intervals over runs within each arm and flag a cell as degenerate when every run on both sides returned the same reward; its interval collapses and it says only that the state was at a ceiling or a floor. These per-cell intervals are exploratory; pooled results separately report fixed-state continuation uncertainty and task-cluster sensitivities. The budget is set by cost: a run is a full agentic rollout in its own sandbox, a few sandboxes run at a time, and a cell of four arms is $4n$ rollouts, a task with two checkpoints $8n$. The paper's claims are therefore about how ΔV behaves across a handful of states per task, not about any one of them to the second decimal.

3.3 Ranking versus utility

A retriever scores skills from the instruction, $r(k \mid \tau)$, and skill routing as practised injects the top of the order π_τ this induces. The implicit claim is that π_τ orders utility: if k is ranked above k' then $\Delta V(s_t, k) \geq \Delta V(s_t, k')$ at the states the agent passes through. The arms test three points on π_τ at each state: *own*, the package written for the task, the top of any ideal order; *retrieved*, the retriever's top-ranked package over the whole library; *distractor*, a package written for a different task, near the bottom of any relevance order. If ranking ordered utility, every informative cell would satisfy $\Delta V(\text{own}) \geq \Delta V(\text{retrieved}) \geq \Delta V(\text{distractor})$, with the first two equal wherever the retriever ranks the own package first.

That last clause locates the information. On a task whose own package is ranked first, *retrieved* duplicates *own* and is not run; such a task shows only whether relevance separates the task's own skill from an unrelated one, which any ranker gets right. Whether ranking quality orders execution utility is decidable only on tasks where the retriever's top choice is not the task's own package, and the results (Section 5) are to be read with that in mind: how many such tasks a batch contains is a property of the retriever and the library, not a choice of ours. The distractor arm carries a second question ranking cannot pose at all, whether ΔV is ever negative: a ranker that is wrong forgoes the benefit of the right skill, while an injected skill that harms costs more than withholding it would have. How every number in this paper is sourced and checked is stated in the Reproducibility Statement.

4 Experimental Setup

Benchmark and harness. All runs are on SkillsBench v1.1 (BenchFlow AI, 2026): 87 agentic tasks, each shipping its own skill packages (232 in all; a package is a directory with a `SKILL.md` and optional scripts and references) and its own verifier, which scores the final state of the sandbox in $[0, 1]$. Tasks run under the benchmark's harness, `benchflow` v0.6.6 (the release that both reads the v1.1 task format and supports the sandbox we can run), in apple/container sandboxes on a single Apple-silicon laptop, three at a time; this bounds the scale of everything below. The harness carries eleven patch hunks across four files, two of which repair defects that had silently invalidated tasks of an earlier round (Appendix D). The agent is `deepagents` v0.7.15, installed from a pre-staged wheelhouse image so that no run depends on a package download. The model is DeepSeek-V4.1-Flash in its non-thinking configuration (`deepseek-chat`), called directly on DeepSeek's platform; every baseline and every fork of Section 5.1 uses that one model. Formal runs started on 21–23 September 2026 against `https://api.deepseek.com` and record experiment-code commit `bd5e3a8a31510c4364593f850ec6fd47c1a6b8e5`. The archive does not record temperature, top-p, maximum output tokens or a sampling seed, and pins the agent to a package version and mutable image tag rather than an immutable artifact; exact sampling replay is therefore unavailable. The uptake probes that fixed the early-checkpoint rule and the replay test below ran earlier on the previous generation (`deepseek-v4-flash`) and are never pooled with the experiment: a recording made by one model cannot be replayed in front of another, so the change of model re-recorded every baseline and re-cut every checkpoint.

Baselines and fork points. For each task we record one baseline rollout with the same model and clocks as the arms. Its trajectory is a sequence of *exchanges*, one model request and response each; a checkpoint is an exchange count c, and forking at c replays the first c exchanges from the recording and lets the model generate from exchange $c+1$. Each task is forked at two rule-defined checkpoints.

The *first-write-rule* checkpoint (labelled early in the artifacts) is the last exchange before the agent's first `write_file` or `edit_file` call — the state in which it has explored but not committed to an implementation, chosen because of the uptake probes of Section 5.1 — with a fallback to a quarter of the trajectory when the baseline never writes; the *three-fifths-trajectory* checkpoint (labelled late) is 60% of the baseline's N exchanges, rounded. Both are clamped so that at least two exchanges are replayed and at least three remain live. The cut is made once per (task, checkpoint) into one folder per arm; the folders' replayed prefixes are checked to be byte-identical (one shared SHA-256) and the exchange count is taken from the harness's own trajectory loader, so the arms of a cell differ only in their assigned protocol. The rule that chose each checkpoint and the number it produced are written beside the cut. The labels are neither universally chronological nor universally pre-/post-write: for `tictoc-unnecessary-abort-detection` the first write occurs after the 60% point, so its first-write-rule cut is numerically later; the `financial-modeling-qa` and `travel-planning` baselines never write, so their first-write-rule labels use quarter-trajectory fallbacks.

Arms and the injection protocol. Four conditions are defined. *None* resumes without injection. *Own* injects the task's own package or packages, the benchmark's native configuration. *Distractor* injects the packages of another task in the batch, assigned by a fixed rotation of the planned task list, so no assignment is chosen after seeing a result. *Retrieved* injects the single package a retriever ranks first for the task's instructions over all 232 packages of the benchmark; the retriever is BM25, chosen because it is deterministic and dependency-free, and the arm is not planned for a task whose top-ranked package is its only own package, where it would duplicate *own*. The fourth arm is the one the ranking-versus-utility question requires; Section 5.1 runs it on every task where it is defined. Injection happens after the sandbox is up and before the agent session opens, over the harness's own upload path: the packages are placed under `/skills`, mirrored into the agent's working directory (its file tools cannot address absolute paths outside it), a pointer file listing them is written beside them, and a short block naming the packages and asking the agent to read a relevant `SKILL.md` before writing its own implementation is appended to the task prompt. The replayed prefix is served from the recording by turn index, so the first turn that can act on the block is the first live one, and the prefix is identical across arms by construction. This protocol was fixed after the two lighter ones — files mounted, and files mounted and announced — were measured to leave the agent's uptake at the bottom rungs of the ladder below. Thus ΔV identifies the whole package-plus-announcement protocol, not the package contents alone; the distractor partially controls the announcement but not package length or scripts.

Clocks, health and failed runs. Four clocks are set explicitly, because a run cut short by any of them ends with reward zero and a nonzero tool count, which is exactly what a useless skill looks like: sandbox setup 120 s, agent install 900 s, and 3600 s each for the agent's wall clock and its idle watchdog; the verifier has 900 s. A run is *healthy* when it has a reward, made at least one tool call, made at least one live model request, terminated normally, and its verifier log shows a test session — a pytest summary, a CTRF report, or the task's own tally (Appendix D gives the reason for the last). Unhealthy runs stay in the manifest and count in n but in no mean; a lane that produces two consecutive runs with zero tool calls stops; no run is retried or deleted.

Outcome and estimator. The outcome is the task verifier's reward. For a cell (task, checkpoint) and arm k, the primary estimate is the assigned-arm difference between mean rewards among runs meeting the fixed health gate: every admitted run counts whether or not the agent read the package. The sole excluded run was an own-arm continuation that timed out after returning reward 1.0; it remains in the manifest, and Section 5 bounds its effect. Cell intervals are descriptive 95% percentile bootstraps with $B = 10\,000$ resamples under a fixed seed. The pooled analysis in Section 5 reports continuation uncertainty conditional on the fixed states and task-cluster sensitivities. A side with fewer than two healthy runs gets a point estimate and no interval, and a cell in which every run on both sides returned the same reward is flagged as degenerate. Uptake is graded from the live segment's tool outputs against the injected package on a ladder fixed before the first treated run — absent, listed in a directory, pointer file read, `SKILL.md` read, skill's API used — with the pass mark at `SKILL.md` read. The treatment-on-treated contrast that conditions on it is reported as an auxiliary and is not causal.

Tasks and scale. Candidates were first drawn from the tasks that ship skills, whose images build on arm64 and whose category is software engineering, excluding the benchmark's hard tier; later rounds widened the pool to the rest of the library except seven tasks this machine cannot run (five amd64-only, two on base images the harness cannot replace). Each candidate is screened on rules fixed before the forks they admit. The first, applied throughout: a task enters only if its baseline reward lies strictly inside $(0, 1)$ and its verifier log shows the tests ran, since a task the model always fails or always solves can say nothing about ΔV, and a zero the verifier wrote without testing says nothing at all. A second rule was added for a later round, for tasks whose verifier returns only 0 or 1 and which the first rule can therefore never admit: a task also enters if at least three recordings run their tests without a harness error and do not all return the same reward, the earliest recording still being the one forked and the others serving only the screen. Tasks entered in rounds, each admitted after the previous round had been analysed: two in the first, five in an extension round (for which the second rule was added) and four in later ones. Reapplying these rules to surviving baselines identifies fourteen passing tasks: the eleven in the plan and three passing-but-unrun tasks. This is a deterministic reconstruction because the final machine-readable screening decision file was not archived; no temporal claim is made from it. Each of the eleven was forked at two checkpoints with six repeats per arm — nine for `crystallographic-wyckoff-position-analysis`, which took a later round of three more — and the retrieved fourth arm on the eight tasks where it is defined, 516 runs in all; Section 5.1 declares the extension where it reports the cells. A task whose environment is found defective after the fact is excluded whole, with the evidence filed beside the manifest, and its runs stay graded and declared. The screen forks one recording per task, so it is a coarse instrument: a task can pass or fail it on the luck of one rollout, and a task that fails it is not shown to be uninformative, only not shown to be informative. The inferential target is the finite map over these adaptively screened realised states, not task prevalence in SkillsBench.

Fidelity of the fork. The estimator rests on a resumed arm reconstructing the state it was forked from. On `dialogue-parser`, under the gateway model, a full replay of a recorded baseline reproduced its reward of 0.667, all 30 tool calls in order and the verifier's test results with zero divergences, while cutting the agent's execution time from 540 s to 30 s and making no live request; the same recording cut at exchange 15 and continued with the live model also finished at 0.667; the replay log is kept with the repository.

5 Results

5.1 The forked-arm experiment

We forked each of the eleven tasks at two execution states — the last exchange before the agent's first file write (first-write-rule, labelled *early*) and 60% of the way through the baseline trajectory (three-fifths-rule, labelled *late*) — and ran every fork to completion under three arms: no injection, the task's own skill package, and a package taken from an unrelated task, plus a fourth on the tasks where it is defined, the pack a retriever ranks first (Figure 2). Four of them, `edit-pdf`, `hvac-control`, `pptx-reference-formatting` and `travel-planning`, entered in later rounds, after the first seven had been analysed, under the same screen and protocol. Each cell has six repeats (nine for `crystallographic-wyckoff-position-analysis`: after the first six had been analysed, three more were run in every arm at both checkpoints to tighten the contrast between the retriever's pack and the task's own set; a sequential extension, and the intervals of those cells are narrower for it); 516 runs were planned and 515 met the fixed health gate; the one run that failed it is excluded from every mean but kept in the manifest. All arms use the same model — DeepSeek-V4.1-Flash in its non-thinking configuration (`deepseek-chat`), called directly on DeepSeek's platform — the same 3600 s clocks and the same replayed prefix, verified byte for byte; the reported ΔV is the assigned-arm difference in mean verified reward among runs meeting the fixed health gate, and the intervals are percentile bootstraps of that difference.

Where skills help and where they hurt. The own-package protocol lowered the mean verified reward in 2 of 22 cells (9%). That figure needs its denominator read carefully: in 6 of the 22 cells every available arm had the same mean (four at the verifier's ceiling, one at its floor and one at an intermediate score); in 5 of those cells every observed reward was identical, while the remaining tie in means retained within-arm variation (Table 1). Those mean-flat cells carry no directional

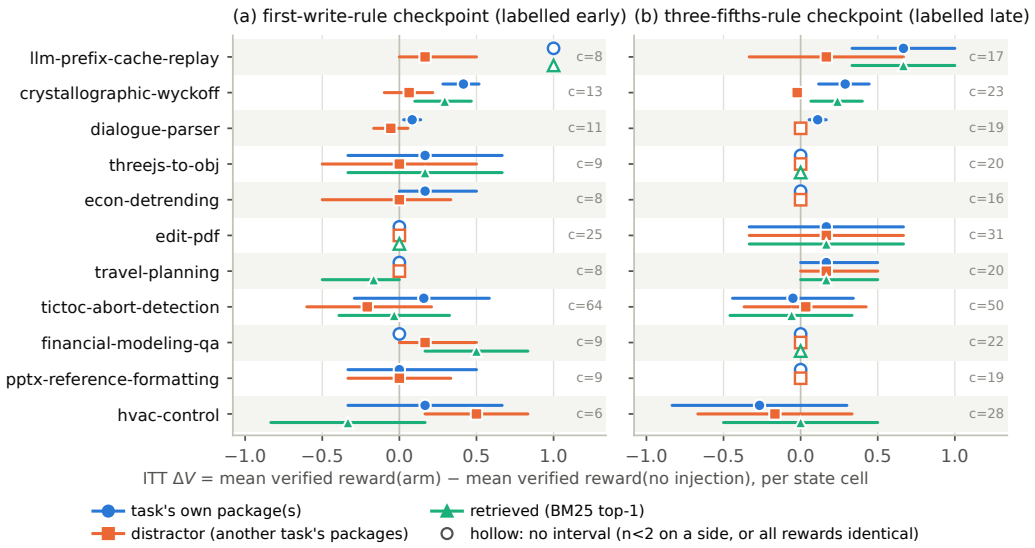


Figure 2: Assigned-arm ΔV per task and checkpoint for the own-package protocol, a distractor protocol and, where it was run, the retriever's top-ranked pack, with 95% bootstrap intervals. Hollow markers carry no interval: either fewer than two healthy runs on a side, or every run on each side returned the same reward, so the bootstrap collapses to the point. Source: `results/ds_20260921/analysis`.

information about ΔV from their point estimates, although the skill was opened in 8 of their 36 treated runs. In the 17 cells with any runwise reward variation, the own-package protocol raised the mean in 12, lowered it in 2 and left it unchanged in 3, with the interval clear of zero in 5 of the 12 positive cells. At six to nine repeats per cell an interval that straddles zero is what a real effect of a few tenths of a point looks like as much as it is what no effect looks like; the sign pattern across cells, not any single cell, is what this experiment can support.

State dependence. For the own-package protocol, the first-write-rule checkpoint's ΔV exceeds the three-fifths-rule one in six of eleven tasks, falls below it in three and equals it in two. The largest rule-labelled gap is 0.433. For `tictoc-unnecessary-abort-detection` the baseline agent's first write came at exchange 64 of 84, after the 60% mark at exchange 50, so its first-write-rule checkpoint is the later of its two. `financial-modeling-qa` and `travel-planning` never wrote in the baseline, so their first-write-rule checkpoints use the quarter-trajectory fallback. At this scale the rule-labelled differences point both ways, so whether ΔV itself moves with the state is not resolved here. The rule label does, however, separate package uptake in this screened map. Pooled over tasks, the agent read the injected `SKILL.md` in 84% of first-write-rule own-arm runs (58/69) and 44% of three-fifths-rule runs (30/68). These are rule-stratified counts, not a universal before/after-writing contrast. Under the assigned-arm estimand, an injection the agent never reads has ΔV of zero, not a missing observation: it is the value of injecting at that state.

Distractors. The distractor protocol lowered mean reward in 4 of 22 cells (18%). The agent opened the irrelevant package in 23 of 69 first-write-rule runs and 8 of 69 three-fifths-rule runs. None of the four drops has its interval below zero; the largest is `tictoc-unnecessary-abort-detection` at exchange 64 (-0.208 [-0.600, +0.208]). A relevance score has no sign with which to express such a drop, and ΔV does. In the other direction, an unrelated package raised reward with its interval clear of zero in one cell: `hvac-control` at exchange 6, +0.500 [+0.167, +0.833], where 6 of 6 distractor runs solved the task against 3 of 6 uninjected ones, a difference an exact test does not separate from chance (two-sided Fisher $p = 0.18$). An injection is also an announcement that help exists and a break in the agent's trajectory, either of

which could move an agent; this is why the own arm is read against the distractor as well as against no injection.

Ranking against utility. The `bm25` top-1 arm is defined on 16 states from eight tasks. 12 states compare one own pack with the full own set; only 4 states from two tasks use a foreign pack. Top-1 is worth more in two states, the same in eight and less in six. This is chiefly a top-one-versus-full-set contrast and only a two-task glimpse of relevance–utility disagreement (Figure 4, Appendix C).

What this experiment does and does not establish. Most cells have six repeats, so their intervals are exploratory. `crystallographic-wyckoff-position-analysis` has nine repeats after a sequential extension whose first round had already been read. Conditioning on uptake is post-treatment, so the auxiliary treatment-on-treated contrast is not causal. Every cell is in Table 1 (Appendix A), with manifests under `results/ds_20260921/analysis`.

Cross-cell inference. Across the fixed 22 screened realised states, the health-gated own-package protocol contrast is +0.147; a runs-only 95% interval conditional on those states is [+0.082, +0.213], and within-state randomization rejects the sharp no-effect null ($p < 0.001$). Cross-task mean inference is method-sensitive: preserving each task's state pair gives a task-first interval [+0.010, +0.319], but the 11-cluster t interval [-0.020, +0.315] crosses zero ($p = 0.079$); the task signs are 8 positive, 1 negative and 2 tied ($p = 0.039$). Distractor, top-1/full-set and first-write-rule-minus-three-fifths reward contrasts remain unresolved. Uptake differs by +39.9 points [+15.8, +64.8] (task-stratified $p < 0.001$), and by +44.3 [+18.9, +70.8] after removing the reversed pair and both no-write fallbacks (Figure 6, Appendix C). Table 2 gives the estimands. The sole health-gate failure is an own-arm HVAC continuation with reward 1.0; exclusion, retaining 1, or coding 0 gives its state contrasts -0.267, -0.167 and -0.333 and pooled effects +0.147, +0.152 and +0.144, respectively.

6 Conclusion

We measure a skill's value at the state where an agent would consume it by forking a live rollout and treating no injection as an arm. Across 22 screened realised task–state cells, the health-gated own-package protocol contrast is +0.147 [fixed-state 95% CI +0.082, +0.213]. Its 11-cluster t interval [-0.020, +0.315] crosses zero, so cross-task mean inference is method-sensitive. Uptake differs by +39.9 points and remains positive after removing all checkpoint exceptions. Reward timing, distractor and top-1/full-set effects remain unresolved.

Limitations. This finite map covers one model, one agent and one benchmark; most cells have six repeats, six are mean-flat (four at the ceiling), and each task has one realised baseline. Tasks entered adaptively in rounds of two, five and four; one received extra repeats. BM25 yields only two foreign top-1 tasks. The intervention is a package-plus-announcement protocol, and a health gate excludes one timeout; three reward codings preserve the pooled direction without making the primary analysis strict ITT.

The result is a signed, auditable finite map and protocol. It can label state-aware abstention, routing and diagnosis; reward gains remain future work.

AI use statement

Large language models are the object of study: every rollout in this paper is an LLM agent, and the quantity measured is the change in its verified outcome when a skill package is injected. Separately from that, in this work we used generative AI tools (LLM-based coding assistants) for implementing and debugging the harness patches, the fork-and-replay tooling and the analysis code, for running and monitoring experiments, and for drafting and revising text and figures, in every case under the authors' direction and review. Every number in the paper is produced by scripts from the run manifests and is matched against raw output on disk by the evidence-ledger checker described in the Reproducibility Statement, so the provenance of a claim does not depend on who, or what, wrote the sentence around it; AI-written code is covered by the same regression checks as the rest, and all AI-assisted text was reviewed by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

The agents studied here execute code with tool access. Every run is confined to the benchmark's own sandbox — an apple/container image built from the task's definition, with no credentials and no reach into the host beyond the harness's upload and result paths — and its only external dependency is the model endpoint. The tasks, verifiers and skill packages are the public SkillsBench release; no data was collected from people, and no run produces artefacts that leave the sandbox other than the verifier's score and the logs the analysis reads. The paper measures whether injecting procedural knowledge into an agent helps or harms it on a benchmark; we see no direct path from that measurement to misuse, and the finding that a mismatched skill is not harmful at this scale should not be read as licence to inject skills without measuring, on tasks or models this paper did not run.

REPRODUCIBILITY STATEMENT

The numerical analysis is traceable, but exact execution replay is not fully pinned. The harness is `benchflow` v0.6.6 with eleven patch hunks and checksums; formal runs were started on 21–23 September 2026 against `https://api.deepseek.com` through endpoint alias `deepseek-chat`, and record experiment-code commit `bd5e3a8a31510c4364593f850ec6fd47c1a6b8e5`. Each manifest row records its arm, checkpoint, health verdict, reward, uptake, request counts and source paths. Each fork carries a shared prefix SHA-256, and scripts regenerate the calibration report, tables and figures.

The archived configs do not record temperature, top-p, maximum output tokens or a sampling seed, and pin `deepagents` only to v0.7.15 and a mutable image tag rather than an immutable historical artifact. The final screening list is a deterministic reconstruction from surviving baselines because its machine-readable decision file was not archived. These omissions limit exact replay and are not replaced with guessed defaults.

An anonymized reviewer artifact accompanies the submission as supplementary material. It contains a redacted manifest row for every one of the 516 formal runs (task, checkpoint, arm, injected packages, reward, termination, health verdict, uptake evidence, clocks, model endpoint and experiment-code commit), the declared arms of every state cell, the per-cell, pooled and sensitivity analyses as JSON, the analysis code that produced them, a checksum manifest covering every member, and the builder that emitted the archive from a fixed repository revision; path-valued fields are relative, and no raw trajectories, credentials or host paths are included. Re-running the released per-cell analysis on the released manifests reproduces every cell reported here, and the two omissions above are stated in the artifact rather than inferred.

Every quantity in this paper is a row in an evidence ledger under one of four tiers — *measured here*, with a path to the raw output; *reported by prior work*, pinned to the arXiv version that reports it; *internal observation*, a number we saw and do not argue from; *unverified hypothesis*, a mechanism nobody has tested — and a checker refuses the build if a number in the text is missing from the ledger, a measured row points at output that does not exist, a prior-work row is unpinned, or a row is filed under a fifth tier. It does not check that a number is filed under the right tier or that the sentence around it is true; it makes those questions answerable by a reader.

REFERENCES

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*, 2023. arXiv:2310.11511v1.

BenchFlow AI. SkillsBench. Software release v1.1, commit `b63b7b2`, 2026. `https://github.com/benchflow-ai/skillsbench`.

Akash Bonagiri, Devang Borkar, Gerard Janno Anderias, Setareh Rafatirad, and Houman Homayoun. CausalFlow: Causal attribution and counterfactual repair for llm agent failures. *arXiv preprint arXiv:2605.25338*, 2026. arXiv:2605.25338v1.

Hongcheol Cho, Ryangkyung Kang, and Youngeun Kim. SkillRet: A large-scale benchmark for skill retrieval in llm agents. *arXiv preprint arXiv:2605.05726*, 2026. arXiv:2605.05726v1.

Chengqi Dong, Chuhuai Yue, Hang He, yandong liu, Fenghe Tang, S Kevin Zhou, Xiaohan Wang, Jiajun Chai, and Guojun Yin. TAPO: Tool-aware policy optimization via credit transfer for multimodal search agents. *arXiv preprint arXiv:2606.05784*, 2026. arXiv:2606.05784v1.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C. Park. Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. *arXiv preprint arXiv:2403.14403*, 2024. arXiv:2403.14403v2.

Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. *arXiv preprint arXiv:2305.06983*, 2023. arXiv:2305.06983v2.

Jaeyoung Kim, Jongho Kim, Seung won Hwang, Seoho Song, and Young-In Song. Relevance to utility: Process-supervised rewrite for rag. *arXiv preprint arXiv:2509.15577*, 2025. arXiv:2509.15577v2.

Maria Marina, Nikolay Ivanov, Sergey Pletenev, Mikhail Salnikov, Daria Galimzianova, Nikita Krayko, Vasily Konovalov, Alexander Panchenko, and Viktor Moskvoretskii. LLM-Independent adaptive rag: Let the question speak for itself. *arXiv preprint arXiv:2505.04253*, 2025. arXiv:2505.04253v1.

Renwei Meng. Policy-Conditioned counterfactual credit for verifiable reinforcement learning of long-horizon language agents. *arXiv preprint arXiv:2606.05263*, 2026. arXiv:2606.05263v1.

Viktor Moskvoretskii, Maria Lysyuk, Mikhail Salnikov, Nikolay Ivanov, Sergey Pletenev, Daria Galimzianova, Nikita Krayko, Vasily Konovalov, Irina Nikishina, and Alexander Panchenko. Adaptive retrieval without self-knowledge? bringing uncertainty back home. *arXiv preprint arXiv:2501.12835*, 2025. arXiv:2501.12835v2.

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. arXiv:2305.15334v1.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. ToolRL: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*, 2025. arXiv:2504.13958v1.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*, 2023. arXiv:2307.16789v2.

Rajinder Sandhu, Di Mu, Cheng Chang, Md Shahriar Tasjid, Himanshu Rai, Maksims Volkovs, and Ga Wu. Aligning dense retrievers with llm utility via distillation. *arXiv preprint arXiv:2604.22722*, 2026. arXiv:2604.22722v2.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023. arXiv:2302.04761v1.

Weihang Su, Yichen Tang, Qingyao Ai, Zhijing Wu, and Yiqun Liu. DRAGIN: Dynamic retrieval augmented generation based on the information needs of large language models. *arXiv preprint arXiv:2403.10081*, 2024. arXiv:2403.10081v3.

Weihang Su, Jianming Long, Qingyao Ai, Qiaozhi He, Yichen Tang, Changyue Wang, Yiteng Tu, Yingbo Wang, and Yiqun Liu. Skill retrieval augmentation for agentic ai. *arXiv preprint arXiv:2604.24594*, 2026. arXiv:2604.24594v3.

Yile Wang, Peng Li, Maosong Sun, and Yang Liu. Self-Knowledge guided retrieval augmentation for large language models. *arXiv preprint arXiv:2310.05002*, 2023. arXiv:2310.05002v1.

Zifei Wang, Wei Wen, Qiang Ji, Keyu Chen, Ruizhi Qiao, and Xing Sun. Skill is not document: Query-conditioned compatibility for llm agent skill routing. *arXiv preprint arXiv:2606.03565*, 2026. arXiv:2606.03565v5.

Hengran Zhang, Keping Bi, Jiafeng Guo, and Xueqi Cheng. An iterative utility judgment framework inspired by philosophical relevance via llms. *arXiv preprint arXiv:2406.11290*, 2024. arXiv:2406.11290v3.

Hengran Zhang, Minghao Tang, Keping Bi, Jiafeng Guo, Shihao Liu, Daiting Shi, Dawei Yin, and Xueqi Cheng. Utility-Focused llm annotation for retrieval and retrieval-augmented generation. *arXiv preprint arXiv:2504.05220*, 2025. arXiv:2504.05220v5.

Hengran Zhang, Minghao Tang, Keping Bi, and Jiafeng Guo. Beyond relevance: Utility-centric retrieval in the llm era. *arXiv preprint arXiv:2604.08920*, 2026. arXiv:2604.08920v1.

YanZhao Zheng, ZhenTao Zhang, Chao Ma, YuanQiang Yu, JiHuai Zhu, Yong Wu, Tianze Xu, Baohua Dong, Hangcheng Zhu, Ruohui Huang, and Gang Yu. SkillRouter: Skill routing for llm agents at scale. *arXiv preprint arXiv:2603.22455*, 2026. arXiv:2603.22455v5.

A EVERY CELL OF THE FORKED-ARM EXPERIMENT

Table 1 lists every (task, checkpoint, arm) cell of Section 5.1: planned and healthy runs, the uptake rate on the ladder of Section 4, mean verified reward with its percentile bootstrap interval, and the mean number of live model requests. Its body is written by the same script that writes the calibration report, from the run manifests released with the paper.

Table 1: Every cell of the experiment: runs, healthy runs, uptake rate, mean verified reward with its bootstrap interval, and live model requests. Failed runs count in n and in no mean. Auto-generated by `experiments/analyze_calibration.py`.

Task	Ckpt	Condition	n	healthy	uptake	mean R	sd	CI	live req.
crystallographic-wyckoff-position-analysis	13	distractor	9	9	0.56	0.608	0.182	[0.506, 0.730]	28.1
crystallographic-wyckoff-position-analysis	13	none	9	9	0.00	0.544	0.178	[0.461, 0.667]	27.4
crystallographic-wyckoff-position-analysis	13	own	9	9	1.00	0.960	0.079	[0.900, 1.000]	22.9
crystallographic-wyckoff-position-analysis	13	retrieved	9	9	0.67	0.839	0.243	[0.678, 0.950]	25.3
crystallographic-wyckoff-position-analysis	23	distractor	9	9	0.11	0.517	0.050	[0.483, 0.550]	18.7
crystallographic-wyckoff-position-analysis	23	none	9	9	0.00	0.539	0.033	[0.517, 0.550]	20.3
crystallographic-wyckoff-position-analysis	23	own	9	9	0.67	0.828	0.260	[0.656, 0.950]	19.1
crystallographic-wyckoff-position-analysis	23	retrieved	9	9	0.56	0.778	0.266	[0.606, 0.939]	22.1
dialogue-parser	11	distractor	6	6	0.33	0.695	0.125	[0.611, 0.778]	14.8
dialogue-parser	11	none	6	6	0.00	0.750	0.091	[0.695, 0.805]	19.0
dialogue-parser	11	own	6	6	1.00	0.833	0.000	[0.833, 0.833]	22.5
dialogue-parser	19	distractor	6	6	0.00	0.667	0.000	[0.667, 0.667]	13.5
dialogue-parser	19	none	6	6	0.00	0.667	0.000	[0.667, 0.667]	21.8
dialogue-parser	19	own	6	6	0.67	0.778	0.086	[0.722, 0.833]	23.2
econ-detrending-correlation	16	distractor	6	6	0.00	1.000	0.000	[1.000, 1.000]	11.5
econ-detrending-correlation	16	none	6	6	0.00	1.000	0.000	[1.000, 1.000]	9.5
econ-detrending-correlation	16	own	6	6	0.00	1.000	0.000	[1.000, 1.000]	10.0
econ-detrending-correlation	8	distractor	6	6	0.00	0.833	0.408	[0.500, 1.000]	7.3
econ-detrending-correlation	8	none	6	6	0.00	0.833	0.408	[0.500, 1.000]	8.0
econ-detrending-correlation	8	own	6	6	1.00	1.000	0.000	[1.000, 1.000]	14.0
edit-pdf	25	distractor	6	6	0.00	1.000	0.000	[1.000, 1.000]	37.0
edit-pdf	25	none	6	6	0.00	1.000	0.000	[1.000, 1.000]	35.5
edit-pdf	25	own	6	6	0.17	1.000	0.000	[1.000, 1.000]	46.0
edit-pdf	25	retrieved	6	6	0.50	1.000	0.000	[1.000, 1.000]	38.2
edit-pdf	31	distractor	6	6	0.00	0.667	0.516	[0.333, 1.000]	34.2
edit-pdf	31	none	6	6	0.00	0.500	0.548	[0.167, 0.833]	30.3
edit-pdf	31	own	6	6	0.00	0.667	0.516	[0.333, 1.000]	37.0
edit-pdf	31	retrieved	6	6	0.00	0.667	0.516	[0.333, 1.000]	34.2
financial-modeling-qa	22	distractor	6	6	0.17	0.000	0.000	[0.000, 0.000]	20.8
financial-modeling-qa	22	none	6	6	0.00	0.000	0.000	[0.000, 0.000]	20.7
financial-modeling-qa	22	own	6	6	0.50	0.000	0.000	[0.000, 0.000]	20.2
financial-modeling-qa	22	retrieved	6	6	0.33	0.000	0.000	[0.000, 0.000]	22.8
financial-modeling-qa	9	distractor	6	6	0.00	0.167	0.408	[0.000, 0.500]	21.8
financial-modeling-qa	9	none	6	6	0.00	0.000	0.000	[0.000, 0.000]	10.8
financial-modeling-qa	9	own	6	6	0.33	0.000	0.000	[0.000, 0.000]	21.0
financial-modeling-qa	9	retrieved	6	6	0.17	0.500	0.548	[0.167, 0.833]	20.0
hvac-control	28	distractor	6	6	0.00	0.500	0.548	[0.167, 0.833]	42.0
hvac-control	28	none	6	6	0.00	0.667	0.516	[0.333, 1.000]	36.3
hvac-control	28	own	6	5	1.00	0.400	0.548	[0.000, 0.800]	41.2
hvac-control	28	retrieved	6	6	1.00	0.667	0.516	[0.333, 1.000]	23.0
hvac-control	6	distractor	6	6	0.17	1.000	0.000	[1.000, 1.000]	51.2
hvac-control	6	none	6	6	0.00	0.500	0.548	[0.167, 0.833]	58.2
hvac-control	6	own	6	6	1.00	0.667	0.516	[0.333, 1.000]	52.0
hvac-control	6	retrieved	6	6	1.00	0.167	0.408	[0.000, 0.500]	61.8
llm-prefix-cache-replay	17	distractor	6	6	0.67	0.500	0.548	[0.167, 0.833]	24.8
llm-prefix-cache-replay	17	none	6	6	0.00	0.333	0.516	[0.000, 0.667]	28.7
llm-prefix-cache-replay	17	own	6	6	1.00	1.000	0.000	[1.000, 1.000]	17.8
llm-prefix-cache-replay	17	retrieved	6	6	0.83	1.000	0.000	[1.000, 1.000]	14.0
llm-prefix-cache-replay	8	distractor	6	6	0.83	0.167	0.408	[0.000, 0.500]	43.3
llm-prefix-cache-replay	8	none	6	6	0.00	0.000	0.000	[0.000, 0.000]	35.7
llm-prefix-cache-replay	8	own	6	6	1.00	1.000	0.000	[1.000, 1.000]	17.7
llm-prefix-cache-replay	8	retrieved	6	6	1.00	1.000	0.000	[1.000, 1.000]	13.5
pptx-reference-formatting	19	distractor	6	6	0.00	1.000	0.000	[1.000, 1.000]	16.5
pptx-reference-formatting	19	none	6	6	0.00	1.000	0.000	[1.000, 1.000]	14.5
pptx-reference-formatting	19	own	6	6	0.00	1.000	0.000	[1.000, 1.000]	12.0
pptx-reference-formatting	9	distractor	6	6	0.00	0.833	0.408	[0.500, 1.000]	21.3
pptx-reference-formatting	9	none	6	6	0.00	0.833	0.408	[0.500, 1.000]	19.2
pptx-reference-formatting	9	own	6	6	0.67	0.833	0.408	[0.500, 1.000]	23.5
threejs-to-obj	20	distractor	6	6	0.00	1.000	0.000	[1.000, 1.000]	22.0
threejs-to-obj	20	none	6	6	0.00	1.000	0.000	[1.000, 1.000]	13.7
threejs-to-obj	20	own	6	6	0.00	1.000	0.000	[1.000, 1.000]	11.2
threejs-to-obj	20	retrieved	6	6	0.17	1.000	0.000	[1.000, 1.000]	14.3
threejs-to-obj	9	distractor	6	6	0.50	0.667	0.516	[0.333, 1.000]	30.8
threejs-to-obj	9	none	6	6	0.00	0.667	0.516	[0.333, 1.000]	30.5
threejs-to-obj	9	own	6	6	1.00	0.833	0.408	[0.500, 1.000]	32.8
threejs-to-obj	9	retrieved	6	6	1.00	0.833	0.408	[0.500, 1.000]	41.3
tictoc-unnecessary-abort-detection	50	distractor	6	6	0.00	0.792	0.371	[0.475, 0.975]	103.0
tictoc-unnecessary-abort-detection	50	none	6	6	0.00	0.758	0.390	[0.458, 1.000]	107.7
tictoc-unnecessary-abort-detection	50	own	6	6	0.83	0.708	0.360	[0.425, 0.950]	91.7
tictoc-unnecessary-abort-detection	50	retrieved	6	6	0.83	0.700	0.377	[0.408, 0.950]	80.2
tictoc-unnecessary-abort-detection	64	distractor	6	6	0.17	0.308	0.367	[0.100, 0.608]	61.0
tictoc-unnecessary-abort-detection	64	none	6	6	0.00	0.517	0.406	[0.217, 0.817]	85.7
tictoc-unnecessary-abort-detection	64	own	6	6	1.00	0.675	0.449	[0.375, 0.975]	112.8
tictoc-unnecessary-abort-detection	64	retrieved	6	6	0.33	0.483	0.289	[0.275, 0.725]	48.2
travel-planning	20	distractor	6	6	0.33	1.000	0.000	[1.000, 1.000]	12.2
travel-planning	20	none	6	6	0.00	0.833	0.408	[0.500, 1.000]	12.8
travel-planning	20	own	6	6	0.17	1.000	0.000	[1.000, 1.000]	12.0
travel-planning	20	retrieved	6	6	0.33	1.000	0.000	[1.000, 1.000]	16.0
travel-planning	8	distractor	6	6	1.00	1.000	0.000	[1.000, 1.000]	32.2
travel-planning	8	none	6	6	0.00	1.000	0.000	[1.000, 1.000]	27.7
travel-planning	8	own	6	6	1.00	1.000	0.000	[1.000, 1.000]	27.3
travel-planning	8	retrieved	6	6	1.00	0.833	0.408	[0.500, 1.000]	26.3

B CROSS-CELL INFERENCE

Table 2 aggregates the map without giving the nine-repeat cells more weight than the six-repeat cells. For the own-package finite map, the primary interval keeps all 22 realised states fixed and resamples continuations; the sharp-null test permutes arm labels within states. Task-level sensitivities preserve each pair of rule-labelled states and use both a task-first bootstrap and an 11-cluster t interval. The distractor and retrieved intervals remain exploratory cell-first summaries; uptake resamples and stratifies by task. Bootstrap procedures use 20,000 draws and fixed seeds. Per-cell intervals are descriptive: among 22 own-protocol cells, three exhaustive permutation tests have raw $p < 0.05$ and Holm/BH retain two; across all 60 treatment-versus-none cells, six are raw, Holm retains one and BH three.

Contrast	n	Estimate [95% CI]	Test
Own − none, fixed states	22	+0.147 [+0.082, +0.213]	$p < 0.001$
Own − none, task bootstrap	11	+0.147 [+0.010, +0.319]	–
Own − none, cluster t	11	+0.147 [-0.020, +0.315]	$p = 0.079$
Distractor − none	22	+0.044 [-0.043, +0.139]	$p = 0.243$
Top-1 − full own	16	-0.017 [-0.141, +0.109]	$p = 0.715$
Own ΔV: early − late	11	+0.098 [-0.067, +0.273]	$p = 0.508$
Own uptake: rule difference	11	+39.9 pp [+15.8, +64.8]	$p < 0.001$

Table 2: Cross-cell estimates among runs meeting the fixed health gate. The fixed-state and task-level rows expose the uncertainty target; the cluster t sensitivity crosses zero. The last two rows use an exact sign test and a task-stratified CMH test.

C FURTHER VIEWS OF THE MAP

Table 3 gives the numbers behind Figure 2; Figures 3–5 render the same cells three more ways; Figure 6 shows the uptake contrast of Section 5 task by task; and Figure 7 is the checkpoint probe that fixed the first-write rule, run on the previous model generation and not pooled with the experiment.

Table 3: Assigned-arm ΔV per task at the early and the late label, for the own-package and distractor protocols, with 95% exploratory bootstrap intervals; † marks cells with fewer than two healthy runs on a side. Checkpoints are exchange counts of the replayed prefix. Auto-generated from the manifest.

Task	ckpt (early / late)	ΔV_{own} early	ΔV_{own} late	ΔV_{dis} early	ΔV_{dis} late	early − late (own)
crystallographic-wyckoff-position-analysis	13 / 23	+0.416 [+0.282, +0.517]	+0.289 [+0.117, +0.444]	+0.063 [-0.098, +0.217]	-0.022 [-0.056, +0.011]	+0.127
dialogue-parser	11 / 19	+0.083 [+0.028, +0.138]	+0.111 [+0.055, +0.166]	-0.055 [-0.166, +0.055]	+0.000 —	-0.028
econ-detrending-correlation	8 / 16	+0.167 [+0.000, +0.500]	+0.000 —	+0.000 [-0.500, +0.333]	+0.000 —	+0.167
edit-pdf	25 / 31	+0.000 —	+0.167 [-0.333, +0.667]	+0.000 —	+0.167 [-0.333, +0.667]	-0.167
financial-modeling-qa	9 / 22	+0.000 —	+0.000 —	+0.167 [+0.000, +0.500]	+0.000 —	+0.000
hvac-control	6 / 28	+0.167 [-0.333, +0.667]	-0.267 [-0.833, +0.300]	+0.500 [+0.167, +0.833]	-0.167 [-0.667, +0.333]	+0.433
llm-prefix-cache-replay	8 / 17	+1.000 —	+0.667 [+0.333, +1.000]	+0.167 [+0.000, +0.500]	+0.167 [-0.333, +0.667]	+0.333
pptx-reference-formatting	9 / 19	+0.000 [-0.333, +0.500]	+0.000 —	+0.000 [-0.333, +0.333]	+0.000 —	+0.000
threejs-to-obj	9 / 20	+0.167 [-0.333, +0.667]	+0.000 —	+0.000 [-0.500, +0.500]	+0.000 —	+0.167
tictoc-unnecessary-abort-detection	64 / 50	+0.158 [-0.292, +0.583]	-0.050 [-0.442, +0.342]	-0.208 [-0.600, +0.208]	+0.033 [-0.367, +0.425]	+0.208
travel-planning	8 / 20	+0.000 —	+0.167 [+0.000, +0.500]	+0.000 —	+0.167 [+0.000, +0.500]	-0.167

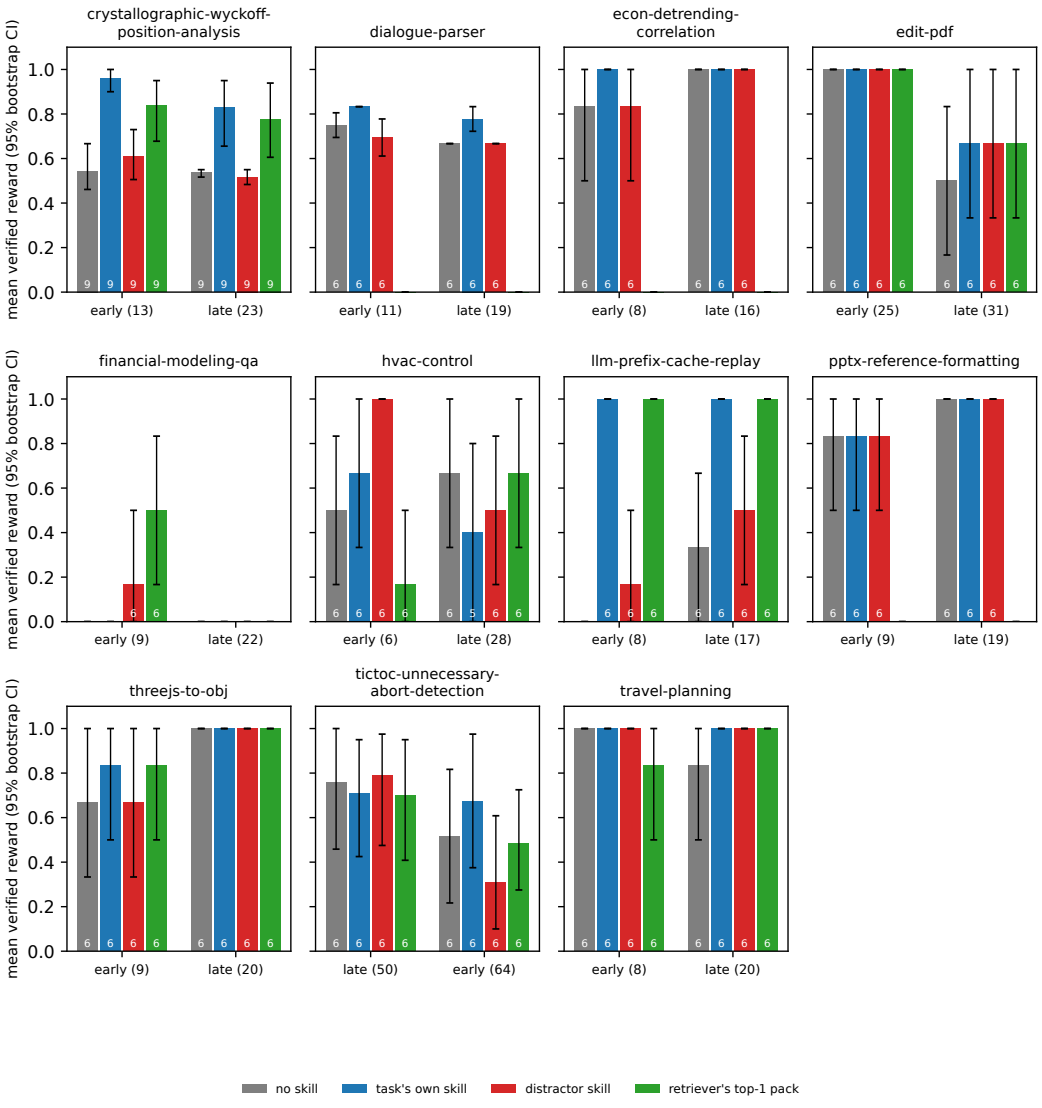


Figure 3: Mean verified reward per arm at the early and late checkpoint, one panel per task, with 95% bootstrap intervals; the number in each bar is its healthy-run count. Source: the released `calibration.json`.

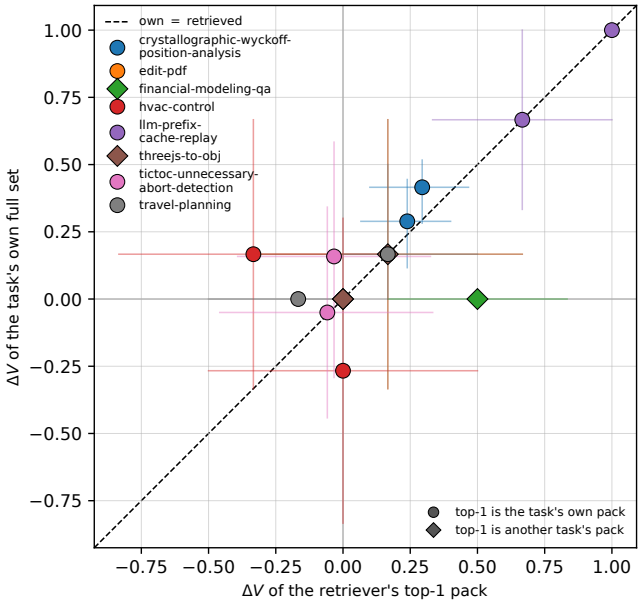


Figure 4: Ranking against utility: the ΔV of the single pack a retriever ranks first (horizontal) against the ΔV of the task's own full set (vertical), one marker per (task, checkpoint) that has the retrieved arm. On the dashed diagonal the top-ranked pack is worth exactly what the full set is; below it the full set is worth more. Diamonds mark the tasks whose top-ranked pack belongs to another task. Source: the released `calibration.json`.

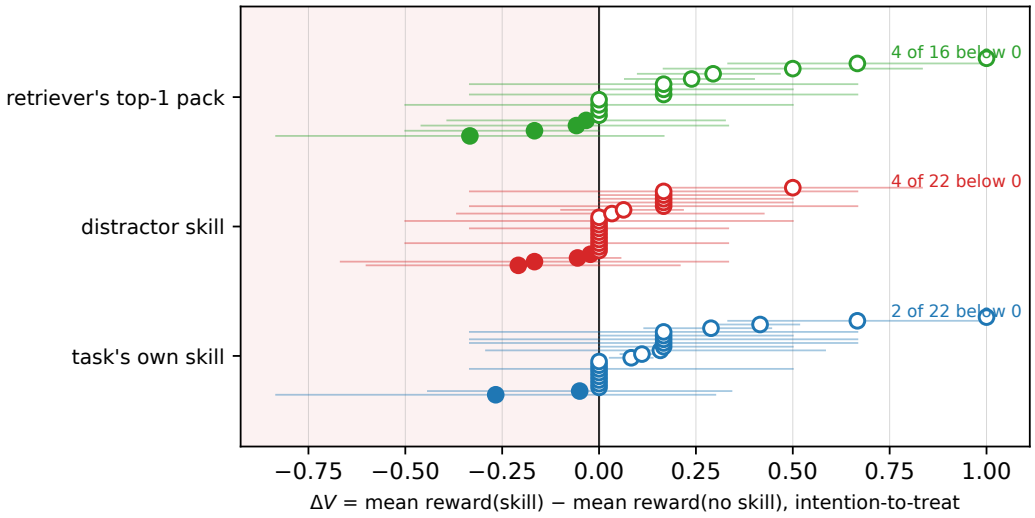


Figure 5: Harmful load: every treatment cell's assigned-arm ΔV laid out by condition, with the region below zero shaded. A filled marker is a cell whose point estimate is below zero; the thin bar is its 95% bootstrap interval where it has one. The label on each row is how many of that condition's cells fall below zero. Source: the released `calibration.json`.

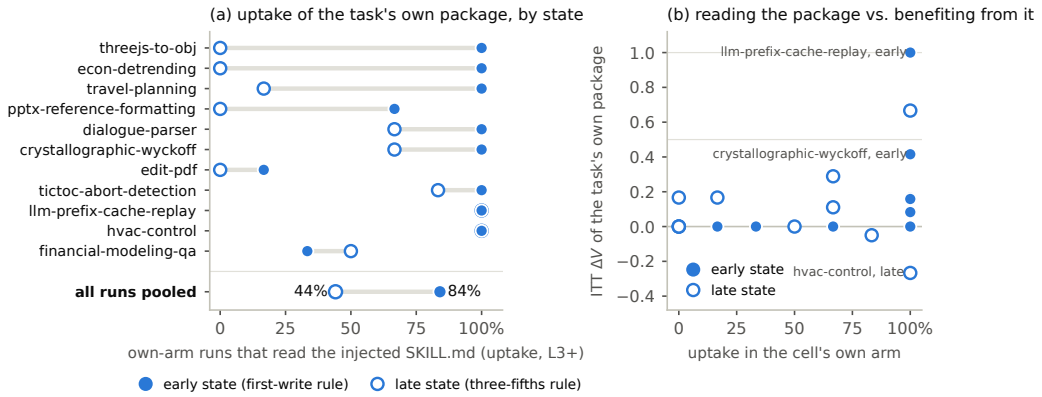


Figure 6: Uptake by state. (a) Share of own-arm runs in which the agent read the injected `SKILL.md` (ladder level L3 or above) at the first-write-rule (early) and three-fifths-rule (late) checkpoint of each task, and pooled over all 137 healthy own-arm runs (58/69 against 30/68). (b) The same per-cell uptake against the cell's assigned-arm ΔV for the own-package protocol: reading the package is close to necessary for a large gain and far from sufficient for one. Generated from the released `pooled.json` and `calibration.json`.

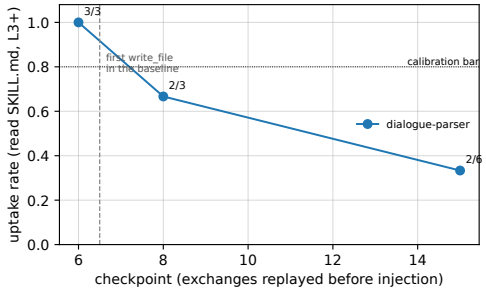


Figure 7: Uptake of the same injected skill on the same baseline of `dialogue-parser`, as a function of the checkpoint it was injected at; the dashed line is the exchange at which the baseline agent first wrote a file. Counts are healthy treated runs that reached L3.

D HARNESS AND ENVIRONMENT DETAILS

The sandbox is apple/container (CLI v1.3.1, builder image v0.13.1), the only runtime available to us that runs the benchmark; runs launch at least 60 s apart. The harness carries eleven patch hunks across four files, kept in the repository with pristine and patched checksums: the resume path is opened to the agent we use and its model proxy is bound where the sandbox can reach it; the image builder is worked around, because apple/container transfers a directory named as a `COPY` source as an empty directory and reports success; and the verifier is given its own package-index configuration, because the offline configuration our agent-install image carries otherwise reaches the task's `test.sh`, whose `uvx`-installed pytest then never runs and whose fallback writes reward 0. Each of the last two defects was found because it had silently invalidated a task of an earlier round of this experiment. The same defect is why a healthy run must show a test session in its verifier log: most `test.sh` scripts in the benchmark write reward 0 when the command that installs their test dependencies fails, so a verifier that never ran is indistinguishable from a failed task in every other field; 36 runs were read that way for two days before the log was consulted. Token counts are never consulted either: on a replay whose agent never reached the model, the harness still reports the recorded usage.

E STATIC RETRIEVAL BASELINES, RE-MEASURED

The forked-arm experiment of Section 5.1 injects the pack a retriever ranks first. This appendix gives the ranking baselines that choice has to be read against, re-measured rather than quoted, and states the two setup facts that change how their numbers should be read.

Setup. Alignment with Zheng et al. (2026) is established at the task level, after an ambiguity that cost us a false start. Its evaluation release publishes the task set and both skill pools, so nothing needs reconstructing; the released file lists 87 tasks and discarding the twelve whose ground truth is generic-only yields exactly the 75 the paper reports on. The composition initially appeared not to match — scoring against the full ground-truth field gives 16 single- and 59 multi-skill tasks against the stated 24/51 — but the paper's split is taken over the *core* ground-truth field, which reproduces 24/51 exactly. We therefore align at the task level and state which ground-truth field each number is scored against, since the two differ by enough to matter. Beyond the task set the harness differs from ours, and their setup injects only skill names and descriptions where ours makes the whole package available on disk. Their reported floor and ceiling, 14.89% without skills and 32.67% with oracle skills, are quoted here as literature values and have not been re-measured under our setup, where they are required control arms in any case. Where we quote a number from either baseline paper we pin the arXiv version. This is not pedantry: two of the three values Wang et al. (2026) reports for SkillRouter changed between v4 and v5, and v4's Comp@k column duplicates its Recall@k column throughout the R3-Skill table. A reader checking our citations against the wrong revision would find them wrong.

Results. Table 4 gives the ranking baselines our own numbers have to be read against. We re-ran them rather than quoting them, for a reason that turned out to matter: the published figures for these systems are not stable across arXiv versions, and one of them does not survive re-measurement.

The R3-Skill release reproduces. Our embedding-only numbers land within 0.20 points of the values its own script carries, and the reranked numbers within 0.39. The residual on Set-Compat deserves one clarification, because it is easy to report as a reproduction failure and it is not one: that script is calibrated against arXiv:2606.03565v4, matching it on all eleven metrics in both blocks, while v5 revised the reranker row upward. Of the 1.78-point gap between our 31.49 and the 33.27 printed in v5, 1.39 points are the paper's own revision and only 0.39 points are reproduction error.

The SkillRouter embedding row is the one that does not survive. That paper reports it at Hit@1 22.09 and attributes the weakness to benchmark mismatch; loading the released checkpoint as its own model card specifies gives 51.46, and substituting R3-Skill's retrieval instruction — the obvious confound, since Wang et al. (2026) state all systems receive the same instruction — moves it only to 48.65. The same code, data and metric reproduce that paper's SkillRouter *reranker* row to within 1.0 point, so the gap is not in our wiring. Forcing the mean-pooling fallback that `sentence-transformers` applies to a checkpoint shipping no `1_Pooling/config.json`

Table 4: Static retrieval baselines on the R3-Skill test split (5,696 queries, 2,050 candidate skills), all scored by one metric implementation copied verbatim from the R3-Skill release. Published values are from Wang et al. (2026) Table 4 (arXiv:2606.03565v5); the two rows marked † are the ones that paper reports for SkillRouter as an external baseline. Set-Compat is computed over the 505 multi-skill queries.

	Measured (ours)			Published	
Retrieval system	Hit@1	nDCG@10	Set-Compat	Hit@1	Set-Compat
R3-Embedding	72.09	81.58	27.92	72.07	28.12
+ R3-Reranker	75.23	81.73	31.49	75.39	33.27
SkillRouter-Embedding†	51.46	62.23	12.08	22.09	–
+ SkillRouter-Reranker	64.52	71.07	20.99	–	–
R3-Embedding + SkillRouter-Reranker†	68.98	78.68	26.14	67.98	24.36

Table 5: SkillRouter reproduced on its own released benchmark (`SkillRouter-Eval-Core`, 75 scored tasks). This is a calibration gate on our re-implementation, not a contribution: Zheng et al. (2026) reports the average over the two pool tiers at a top-20 rerank depth, so that is the row to compare. We fall short on both metrics, by more on recall than on precision at rank one.

	Hit@1		Recall@10	
Pool tier	Ours	Published	Ours	Published
Easy (78,361 skills)	66.67	76.0	55.31	–
Hard (79,141 skills)	64.00	72.0	53.60	–
Average (reported protocol)	65.34	74.0	54.45	70.4
shortfall	−8.66		−15.95	

— which this one does not — yields 25.61, 3.5 points from the published value. We therefore report the row as consistent with a pooling misconfiguration rather than asserting that as its cause: the same checkpoint scores well on a different external benchmark in the same paper, which a global pooling fault would not permit.

Table 5 is a calibration gate rather than a result. Reproducing Zheng et al. (2026) on its own released benchmark leaves us 8.66 points short on Hit@1 and 15.95 on Recall@10 against the protocol the paper actually reports (the mean of the two pool tiers at a top-20 rerank depth). A shortfall of this size is not the signature of a mis-integration, which costs upwards of twenty points, but it is large enough that we do not claim any ordering between these two retrievers. Two candidate causes are open and untested: our metric implementation scores the graded relevance labels as binary, and we truncate skill bodies to 2,000 tokens, which is a consequential choice for a system whose central claim is that the body carries the decisive signal.

The 8B configuration Zheng et al. (2026) report at 76.0 is omitted throughout. Those weights were never released, so the number cannot be reproduced by anyone and we quote it only as a claim.