The Scaling Laws of Skills in LLM Agent Systems

Qiguang Chen[1,*], **Qiming Yu**[1,*], **Yuhang Gu**[1], **Zhuoye Huang**[1], **Hanjing Li**[1], **Hongyu Liu**[1], **Simin Liu**[1], **Jinhao Liu**[1], **Dengyun Peng**[1], **Jiangyi Wang**[1], **Zheng Yan**[1], **Fanqing Meng**[1,2], **Ethan Qin**[1], **Carl Che**[1,†], **Mengkang Hu**[1,3,†]

[1]Evolvent AI Team [2]National University of Singapore [3]University of Hong Kong

[*]Equal contribution. [†]Corresponding authors.

Abstract

As agent systems scale, skills accumulate into large reusable libraries, yet their scaling laws remain poorly understood. Across 15 frontier LLMs, 1,141 real-world skills, and over 3M routing or execution decisions, we identify two coupled laws. **Routing law**: single-step routing accuracy decays logarithmically with library size (R^2>0.96 for all models), with errors progressing from local skill competition to cross-family drift and capture by overly general "black-hole skills". **Execution law**: before state realization, joint routing is approximately multiplicative, whereas correct execution can improve difficult downstream decisions by about $4\times$. A single parameter, the routing logarithmic decay slope b, couples the two laws: routing-side fits predict execution-side rescue across models, showing that the same library property controls both pre-execution collapse and downstream recoverability. The laws are actionable: law-guided optimization raises held-out routing accuracy from 71.3% to 91.7%, reduces hijack from 22.4% to 4.1%, and transfers directionally to real downstream tasks, improving mean pass rate from 49.3% to 61.6% on ClawBench and from 28.4% to 34.5% on ClawMark. These results show that agent performance depends not only on model capability, but also on the structure, granularity, and exposure policy of the skill library.

1 Introduction

As agent systems develop, skills increasingly accumulate into reusable libraries [1–4]. Early action-selection systems [5–7] and benchmarks [8, 9] showed that language models can select from large sets of callable interfaces [10]. More recent skill-centric work [11, 12] further treats reusable skills as a central abstraction for building capable agents [13–17]. Together, these systems suggest a natural scaling premise: expanding a skill library should enlarge the space of behaviors available to an agent, while remaining failures can be attributed to limited model capability or imperfect routing.

This shifts the central question from "can the model use a skill?" to "how does a growing skill library reshape the agent's routing and execution dynamics?" Existing benchmarks typically vary tasks, models, or action formats, but keep the library as background infrastructure. We instead make library size and structure the object of study. The resulting failure mode is not ordinary hallucination. In our plain-text routing protocol, vague prompts can yield 0% hallucinated skill names while task-level routing accuracy falls to 18.3% and in-library hijack exceeds 80%. The model stays inside the library, but lands in the wrong part of it.

To address this problem, we characterize this behavior through empirical laws of skill libraries. Across 15 frontier LLMs, 1,141 real-world skills, and over 3M routing or execution decisions, we find two coupled empirical laws. The *routing law* governs the pre-execution regime: single-step accuracy follows $Acc(N) = a - b \ln N$ with R^2>0.96 for every model, while errors progress from local competition among neighboring skills to cross-family drift and finally capture by overly general "black-hole" skills. The *execution law* starts after state is realized. Before any artifact exists, multi-skill routing composes approximately multiplicatively; after correct execution, concrete state can

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

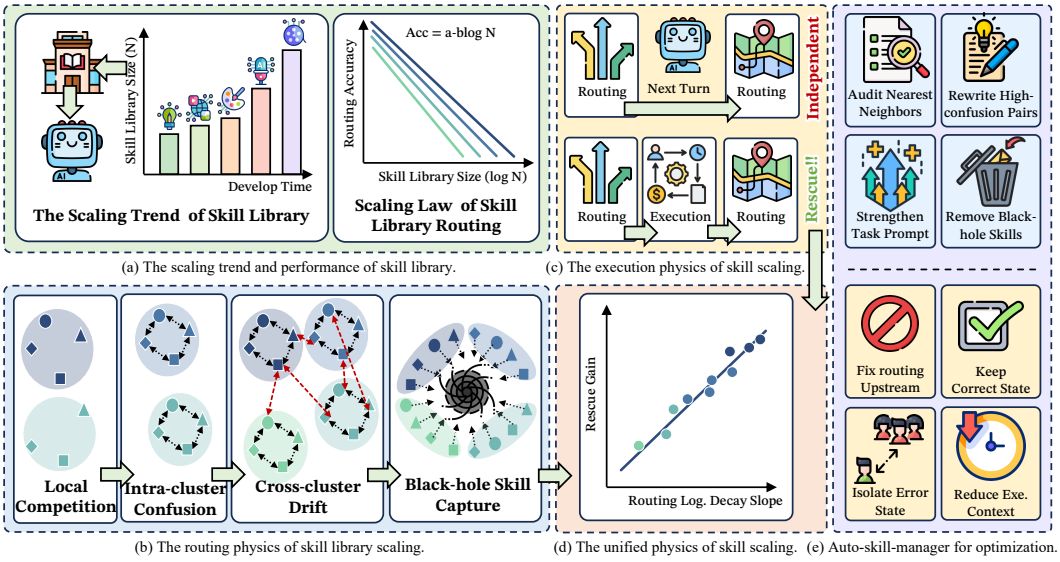


Figure 1: **Skill-library scaling laws and optimization implications. (a)** Reusable skills accumulate into larger exposed libraries. Routing accuracy decays approximately linearly with $\log N$, with model-dependent slope b. **(b)** Errors progress from local competition to cross-family drift and finally capture by overly general "black-hole" skills. **(c)** Execution separates pre-state compounding from post-state rescue. **(d)** The same library property links routing collapse and execution rescue: b predicts rescue across models without refitting. **(e)** The laws suggest concrete optimization actions: audit neighbors, rewrite confused pairs, strengthen prompts, and apply additional manager actions.

rescue difficult downstream decisions, producing about a $4\times$ gain on hard pairs. Thus execution does not simply add another chance to fail. It can either propagate bad state or provide evidence that disambiguates the next route. The two laws are connected by a single interpretable parameter: the routing decay slope b. A larger b means that accuracy collapses sooner as the library grows, reducing the probability that correct upstream state is available for downstream rescue. Conversely, when routing-side fits are combined with an independently measured rescue coefficient, b predicts execution-side rescue across models without refitting (ρ=0.74, p<0.005). This coupling is the central result of the paper: the same library property that controls pre-execution routing collapse also controls whether execution can provide useful downstream state. It turns the laws from retrospective descriptions into design constraints: reduce local competition, sharpen skill boundaries, remove abstract attractors, and expose execution state when it is likely to help.

Guided by these laws, we instantiate a law-guided auto skill manager at the law-predicted transition point for median-b agents. The manager applies four library-side actions: nearest-neighbor auditing, description-boundary rewriting, abstract-skill removal, and prompt anchoring. On 1,600 held-out tasks, it improves routing accuracy from 71.3% to 91.7% and reduces the hijack rate from 22.4% to 4.1%. Factorial ablations identify boundary rewriting (+12.8%) and abstract-skill removal (+4.9%) as the main contributors, consistent with the predicted failure stages. The same design transfers directionally to downstream ClawBench/ClawMark execution settings: mean pass rate improves from 49.3% to 61.6% on ClawBench, and from 28.4% to 34.5% on ClawMark. Code is available at `https://github.com/evolvent-ai/skill-laws`.

Our contributions are:

- **Two-stage laws of skill libraries.** We establish coupled routing and execution laws that characterize how large skill libraries fail before state is realized and recover after correct execution.
- **A unified law structure for skill libraries.** We show that the routing decay slope b links library-scale routing collapse to execution-side rescue across models without refitting.
- **Guidance for optimizing skill-based agent systems.** We turn the laws into library optimization methods for exposure control, boundary rewriting, attractor removal, and execution-state use, with held-out and downstream validation.

2 Related Work

Our work sits at the intersection of skill routing, execution-aware agent systems, and scaling laws. Existing work studies how agents select or use external capabilities, but not the coupled routing and execution laws that appear as the skill library itself scales.

Routing over external capabilities has been studied through callable-interface systems, agent benchmarks, and skill-centric agents [11, 12, 17–19]. Toolformer [5], ToolLLM [6], ToolAlpaca [20], Gorilla [7], API-Bank [8], TaskBench [9], OpenAgents [21], AgentBench [22], and recent surveys and skill-focused work [1, 2, 23] establish large external action spaces as a core agent interface. Closest to our routing setting, ComplexFuncBench [24] studies multi-step function selection, BiasBusters [25] studies systematic selection bias, and xRouter, EvoRoute, AutoTool, and MetaToolAgent optimize routing or tool choice [26–29]. These works usually vary task difficulty, router policy, or benchmark domain rather than treating skill-library size and geometry as the scaling axis.

Execution-aware agent systems study how intermediate state, feedback, and multi-step interaction change downstream behavior. ReAct [30], Plan-and-Solve [31], Reflexion [32], HuggingGPT [33], MetaGPT [34], and Voyager [35] show that traces, artifacts, feedback, and orchestration can improve end-to-end completion [23, 36–39]. Our execution law isolates a narrower mechanism: before concrete state appears, multi-skill routing is close to multiplicative composition; after correct execution, realized artifacts can rescue difficult downstream routing.

Our contribution is to connect these strands through empirical laws for skill-library scaling. Prior scaling laws study compute, data, model size, or training allocation [40–44]; retrieval and representation work explains dense candidate neighborhoods [45–48]; and the Hick-Hyman law gives a logarithmic analogue for capacity-limited choice [49, 50]. We instead scale the inference-time skill library and show that routing decay, error cascades, black-hole capture, and execution-side rescue are coupled by a unified parameter.

3 Problem Formulation & Experimental Setup

Problem formulation. Let $\mathcal{S}_N = \{s_1, \ldots, s_N\}$ be the exposed skill library, where each skill has a selectable name, a natural-language description, and an execution interface. Each task q has a gold skill $s^\star(q) \in \mathcal{S}_N$; that is, the required skill is present in the exposed library by construction. We therefore study availability-preserving failures: the model has access to the necessary skill but may select a wrong in-library alternative. We study two regimes. In a *routing trial*, the model sees only q and the skill descriptions, and the router $\mathcal{R}_\theta$ (LLM θ) proposes

$$\hat{s} = \mathcal{R}_\theta(q, \mathcal{S}_N), \tag{1}$$

where $\hat{s}$ may be correct, wrong but in-library, out-of-library, or an abstention. In an *execution trial*, routing and execution are interleaved across K steps or sub-tasks within a pipeline. At step k, the model routes with context c_k and then executes the selected skill: $\hat{s}_k = \mathcal{R}_\theta(q_k, \mathcal{S}_N, c_k), \quad y_k = \mathcal{E}_\theta(q_k, \hat{s}_k, c_k)$, where y_k is the realized artifact and c_k contains the task, skill descriptions, and, after execution begins, upstream artifacts $y_{<k}$. Thus routing trials isolate selection before state realization, while execution trials test how realized state changes downstream routing.

The primary routing metric is

$$Acc(N) = \Pr[\mathcal{R}_\theta(q, \mathcal{S}_N) = s^\star(q)]. \tag{2}$$

Errors are partitioned into *in-library hijack* ($\hat{s} \in \mathcal{S}_N, \hat{s} \neq s^\star$), *hallucination* ($\hat{s} \notin \mathcal{S}_N$), and abstention. For route-only pipelines, with $s_k^\star = s^\star(q_k)$,

$$Acc(N, K) = \Pr[\hat{s}_k = s_k^\star \text{ for all } k = 1, \ldots, K]. \tag{3}$$

We compare joint success to the multiplicative baseline implied by independent routing. Execution trials additionally score artifact correctness $Q_k \in \{0, 1\}$; in two-step analyzes, A denotes the upstream step and B the downstream step, and we measure how correct or wrong state from A changes routing or quality on B. See Append. A for more mathematical details.

Experimental setup. The scaled library contains 1,141 real-world software-agent skills from public Agent Skills, Claude Code, MCP, and community GitHub repositories after near-duplicate removal [51–58]. Near-duplicates are removed with BAAI/bge-m3 embeddings [59] (cosine > 0.95),

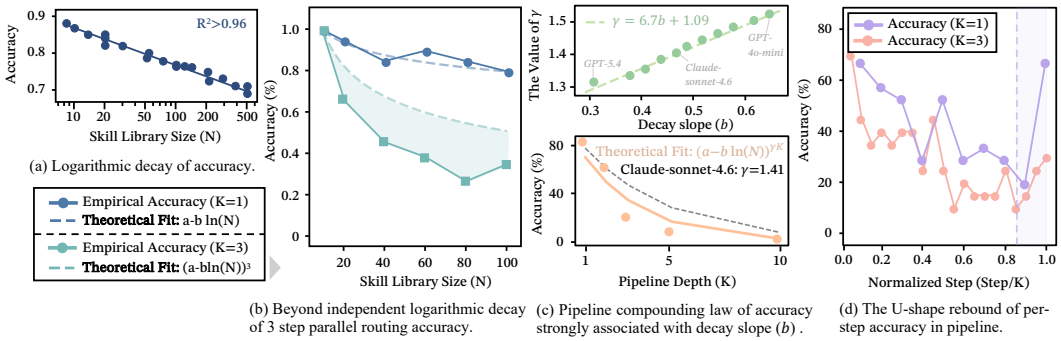


Figure 2: Library scale induces logarithmic single-step decay, super-independent pipeline loss, and a mid-chain routing trough.

and the resulting skills are organized into 14 software-automation domains. Human annotators constructed 4,075 tasks from predefined skill combinations, with GPT-5.4 assistance only for task synthesis; we retain only tasks with unanimous gold-skill annotations (κ=0.91, $\sim$9% excluded). In addition, we integrate queries from Terminal-Bench [60], τ-Bench [61], τ^2-Bench [62], FlowBench [63], PinchBench [57], and related sources, further expanding them and adding tool annotations.

For a size-N routing condition, we include the gold skill for each task and fill the remaining $N-1$ slots with domain-stratified distractors sampled without replacement; this keeps the target distribution fixed while changing library exposure. We use deterministic decoding where provider APIs permit it, parse the first skill-name span returned by the model, and score unmatched names as hallucinations rather than forcing a nearest match. Unless otherwise specified, each case is evaluated an average of four times. Main routing curves sweep $N \in \{10, 20, 50, 100, 200, 500\}$ and $K \in \{1, 2, 3, 5, 10\}$ with n=500 domain-stratified tasks per condition. We fit $Acc(N) = a - b \ln N$ by ordinary least squares over the swept sizes and report Wilson intervals for binomial rates; cross-model correlations use Spearman ρ with percentile bootstrap confidence intervals. Appendix C.1 reports per-model fits, and Appendix C.3 gives retrieval / router baselines and task-difficulty controls.

We evaluate 15 LLMs: GPT-4o-mini [64], GPT-5-mini [65], GPT-5.4-mini [66], GPT-5.4 [67], Claude Sonnet 4.6 [68], Claude Opus 4.6 [69], Gemini 3.1 Pro, Gemini 3.1 Flash Lite [70], GLM-5 [71], GLM-4.7 [72], Kimi K2.5 [73], Kimi K2.6 [74], Doubao Seed 2.0 Pro [75], DeepSeek-V4 Pro [76], and Qwen3-235B [77]. Routing uses plain-text single-skill selection unless stated otherwise, so hallucinations remain measurable. Constrained routing is used only for execution-side isolation and conditions where valid in-library selection must be enforced; it is not pooled into the plain-text hallucination or hijack analyzes. Prompts are in Appendix B. Optimization experiments use the same held-out routing protocol as the diagnostic studies.

4 Routing Law: Library-Scale Selection

This section analyzes the routing law before execution: the model observes a task and skill descriptions, chooses $\hat{s}$, and has no execution artifact y_k in context. The central effect is not just that larger libraries contain more candidates; they also create denser neighborhoods of plausible alternatives, so scale changes both the rate and the geometry of routing errors.

4.1 Library Scale: Logarithmic Decay and Compounding

We first ask whether increasing N only adds independent choices or changes the structure of the routing problem. For each N, every trial exposes the gold skill plus sampled distractors from the same domain mixture; for route-only pipelines, the K gold skills are fixed and only the exposed distractor set changes. First, single-step routing accuracy decreases almost linearly with $\ln N$ (Fig. 2 (a)). Second, three-step routing plans fall below the accuracy predicted by independently repeating the single-step selector (Fig. 2 (b)). Library scale therefore weakens individual decisions and also makes planned sequences more error-prone.

Together, these results give the first part of the routing law. The logarithmic decay suggests a finite discrimination budget: as N grows, the target skill faces more nearby competitors (Sec. 6). The

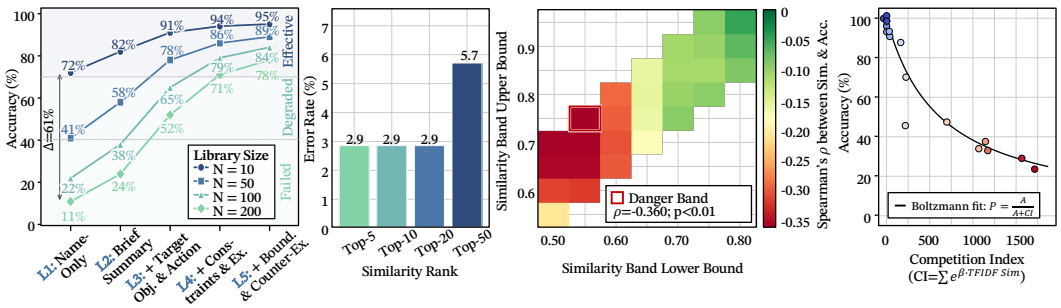


Figure 3: **Local Competition Organizes Error.** **(a)** Accuracy decays with N in all description regimes, but better descriptions reduce the decay rate. **(b)** Top-50 introduces more errors than Top-20/10. **(c)** Danger band $[0.55, 0.75)$ shows strongest negative ρ with accuracy. **(d)** The Boltzmann fit between the competition index and accuracy within the danger zone, where R^2_{CI}=0.55, outperforms the fit with N alone, where R^2_N=0.26.

compounding result explains why this single-step loss becomes worse in planned sequences (Fig. 2 (c)). A wrong early route moves the plan into the wrong semantic neighborhood, making later routes more likely to fail. The U-shape in Fig. 2 (d) then locates the fragile point: middle steps inherit most plausible continuations, while terminal steps partially recover when the final goal narrows the choice set. Append. A & B.8 report mathematical analysis and supplemental cascade diagnostics.

Routing Scaling law. Single-step routing follows $Acc(N) = a - b \ln N$ with R^2>0.96; each doubling costs $\approx$ 3 percentage points. In planned route-only pipelines, the loss compounds as $Acc(N, K) \approx (a - b \ln N)^{\gamma K}$ with empirical exponent γ=6.7b + 1.09. Per-step accuracy is U-shaped: middle steps are more error-prone than the first or last.

4.2 Local Competition: Description Boundaries

Having established the decay law, we next ask what drives its slope b. Our hypothesis is simple: before execution state is available, routing depends on how clearly the target skill is separated from similar but distinct skills. We test this by varying description quality while holding the task, gold skill, distractor set, and model fixed. More distinguishable descriptions improve accuracy and reduce decay, showing that skill boundaries matter (Fig. 3 (a)).

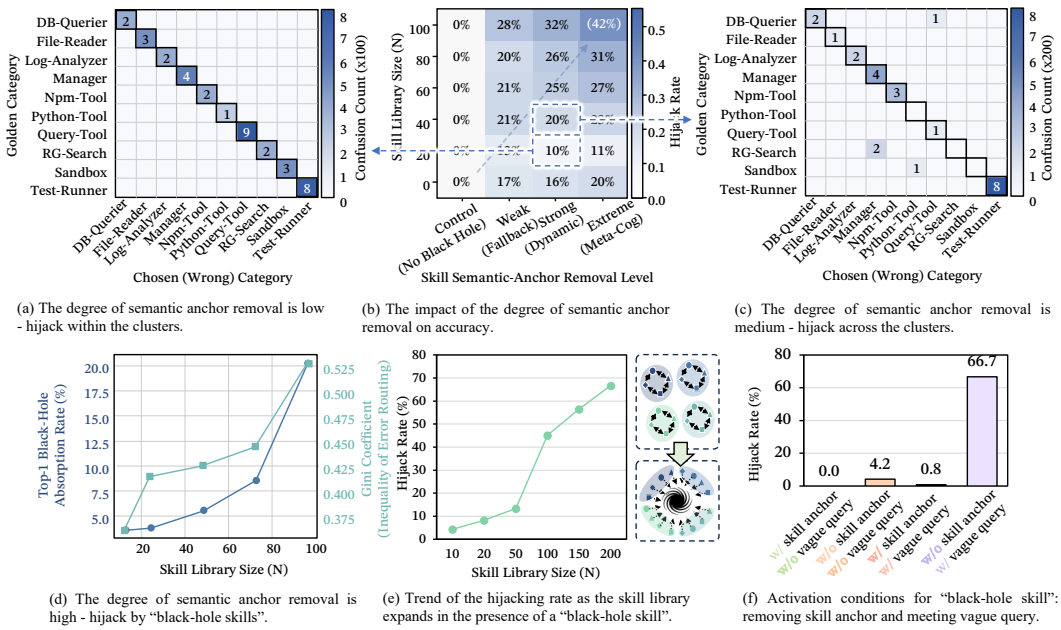
Boundary quality, however, does not explain all errors. We therefore analyze where the remaining mistakes occur. Fig. 3 (b, c) shows that errors concentrate around intermediate-similarity skills: alternatives that are close enough to be plausible, but not close enough to trigger obvious disambiguation. Exposing more candidates can therefore hurt when it adds these near-miss alternatives.

Together, these results isolate two sources of routing difficulty: description quality defines the target skill, while local competition reflects nearby alternatives. Hence, global library size is only a coarse proxy: crowded neighborhoods remain difficult despite better descriptions, and CI predicts per-skill failures better than N alone (Fig. 3 (d)). The L4→L5 paradox follows: in small libraries, counterexamples sharpen boundaries; at scale, they may instead foreground plausible confusers and redirect attention to the alternatives they aim to exclude (Append. A & C.2). Appendix C.5 further tests the mechanism by directly manipulating skill-boundary distances.

Semantic competition mechanism. Description quality changes the effective slope: $P(N, \ell) = a_\ell - b_\ell \ln N$, with L1 name-only descriptions at 11–72% and L4 constraint/example descriptions at 71–90% (b_{L1}=0.32 vs. b_{L4}=0.08). Remaining errors are local: 94% stay inside the correct cluster, and the danger zone is intermediate similarity $[0.55, 0.75)$. In this band, $Acc = A/(A + CI)$, and CI predicts errors better than N alone (R^2_{CI}=0.55 vs. R^2_N=0.26; see Append. C.5 for the definitions of CI, A, and β).

4.3 Anchor Loss: Drift and Capture

After identifying what controls the slope, we examine where lost probability mass is redistributed. Terminology for the two manipulations is fixed in Appendix B.7: *concrete anchor* and *vague query* describe the query side, while *specific* and *abstract skill* describe the skill side, so "abstract-skill removal" and "skill-anchor ablation" name one intervention. We construct paired conditions by



(a) The degree of semantic anchor removal is low - hijack within the clusters.

(b) The impact of the degree of semantic anchor removal on accuracy.

(c) The degree of semantic anchor removal is medium - hijack across the clusters.

(d) The degree of semantic anchor removal is high - hijack by "black-hole skills".

(e) Trend of the hijacking rate as the skill library expands in the presence of a "black-hole skill".

(f) Activation conditions for "black-hole skill": removing skill anchor and meeting vague query.

Figure 4: Weak task anchors and vague skill descriptions convert local skill-cluster drift into black-hole skill capture.

preserving the underlying task intent while removing concrete nouns, file names, tool-specific verbs, or schema anchors from the query; skill-anchor ablations similarly remove boundary clauses from candidate descriptions while preserving names and interfaces. The pattern is ordered: strong task-query anchors induce local substitution, weak anchors induce in-library drift, and weak anchors combined with vague skills induce "black-hole skill" capture.

In-library drift emerges as task anchors weaken (Fig. 4 (a)). Concrete prompts with strong anchors remain highly accurate at $N{=}100$ (89.7%–98.3%). The drift is structured: errors concentrate within the target skill's functional family and increase with library size as anchors weaken (Fig. 4 (b, c)). This suggests that the model preserves coarse task information but fails to reliably select the correct skill within the same functional cluster. The final stage is black-hole capture, where a few skills become attractors with much higher Gini coefficients (Fig. 4 (d)). As the library grows, broad high-abstraction skills absorb disproportionate routing mass (Fig. 4 (e)). To further characterize this pattern, we analyze routing-mass distributions across skills. Capture occurs only when both skill-anchor removal and vague queries are present (Fig. 4 (f)).

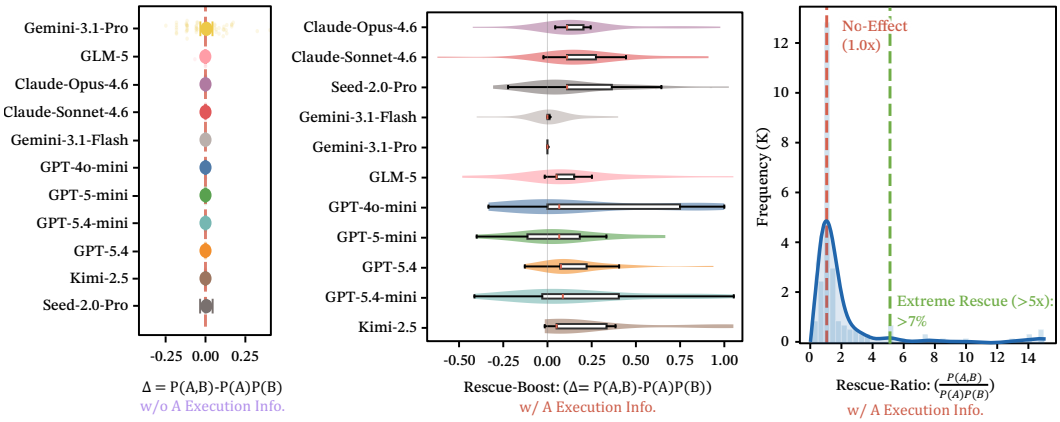
Drift and attractor mechanism. With strong anchors, errors are geometrically local substitutions within the same functional family. Weak anchors break this locality: accuracy collapses, but the model still prefers existing skills over hallucinated names. Black-hole capture is conjunctive, not additive: neither weak task anchors nor vague skills are sufficient alone, but together they induce attractor routing.

5 Execution Law: State Realization

The execution law begins where the routing law ends: a selected skill may produce state. We distinguish three cases. First, without state, multi-step skill use reduces to composed routing decisions. Second, after correct execution, the artifact can inform the next decision. Third, if the artifact is incorrect or mismatched, the same state channel may propagate errors or congest context. Thus, the key question is not whether execution helps, but when state alters downstream routing.

5.1 No State Realization: Multiplicative Baseline

We first establish a baseline before any execution artifact is produced. The question is whether routing steps within a pipeline interact merely because they are planned step-by-step. We form ordered two-skill pairs from annotated pipelines, expose the same library to the single-step and paired prompts, and suppress execution artifacts in the paired prompt. If the two routes do not interact,



(a) The pipeline routing independence without execution information.

(b) The pipeline routing with rescue-boost with real execution information.

(c) The distribution of rescue-ratio with real execution information.

Figure 5: Before state realization, paired routing is approximately multiplicative; after correct execution, realized state rescues difficult downstream routes.

pipeline success should factor as $P(A \cap B) = P(A)P(B)$ (for convenience, we use P to denote the probability of success Acc); any systematic deviation Δ would indicate interaction before state realization. Across 1,690 skill pairs, Δ is near zero (mean < 0.01) across all models (Fig. 5 (a)).

This shows that pipeline prompting alone creates neither reliable positive interaction nor systematic degradation. We attribute this to the fact that the router is modeled as a conditional distribution over skills given the prompt, and the no-state isolation setup removes dependence between the two steps before any execution artifact is observed (Appendix A).

No-state multiplicative baseline. Before execution state exists, two-step routing is approximately multiplicative: $P(A \cap B) \approx P(A)P(B), \qquad \Delta \equiv P(A \cap B) - P(A)P(B) \approx 0.$

5.2 Correct State Realization: Downstream Rescue

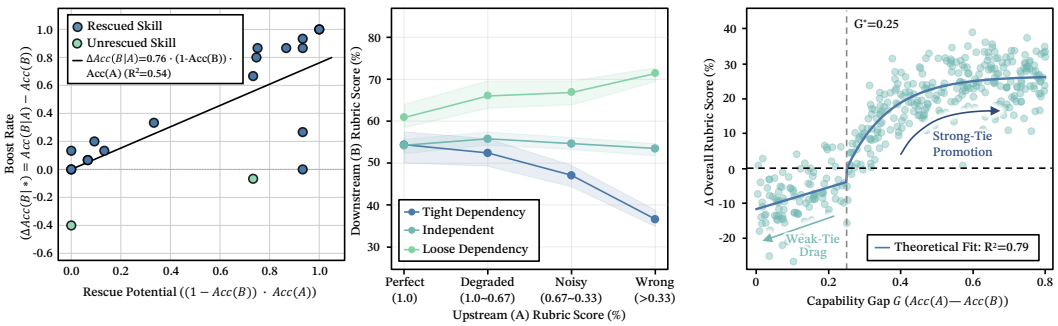
Correct execution changes downstream routing by turning the upstream artifact into concrete evidence for the next skill. In the state-realization condition, we execute the upstream skill, insert its artifact verbatim into the downstream routing context, and keep the downstream gold label fixed relative to the no-state condition (Fig. 5 (b, c)). Rescue should therefore concentrate where upstream execution is correct and pre-state downstream confidence is low.

As shown in Fig. 6 (a), regressing $\Delta P(B|A)$ on $(1 - P(B))P(A)$ yields a slope of $2\alpha \approx 0.76$ across 1,690 pairs. The fit is strongest in the hardest quartile (R^2=0.81 vs. 0.54 overall), and the effect size follows the same pattern: hard downstream decisions improve about four times more than median-difficulty ones. This rescue disappears in the no-state condition, showing that the gain is not due to merely pairing two skills. Instead, correct execution turns an underspecified downstream route into a grounded selection problem. We attribute this effect to increased downstream context concreteness, bounded by upstream success and downstream headroom (Appendices A and D.5).

Correct-state rescue law. Correct upstream execution rescues downstream routing in proportion to upstream success and downstream headroom: $\Delta P(B \mid A) = 2\alpha(1 - P(B))P(A), \qquad 2\alpha \approx 0.76.$

5.3 Incorrect State Realization: Propagation and Tie-Dependent Synergy

Wrong upstream output has two different effects. We label pair dependency strength κ from the task graph and verify it by whether the downstream checker requires the upstream artifact. If the downstream skill needs that output, the error is carried forward (Fig. 6 (b)): tight pairs lose over 15% because the required input is missing or incorrect. If the downstream skill does not really need it, the model can ignore the bad output and solve from the downstream task itself: loose pairs gain 2.8%. The no-dependency baseline stays near zero, and ignored-upstream behavior rises by 10–15%, so the gain comes from ignoring the wrong upstream result, not from fixing it.



(a) The validation of rescue law: $\Delta Acc(B|A)$=0.76 · (1-Acc(B)) · Acc(A) (R2=0.54)

(b) Rubric scores for two-step task completion under varying degrees of semantic coupling between skills.

(c) Strong-Tie Promotion and Weak-Tie Drag under Crowding Risk.

Figure 6: Incorrect state propagates through tight dependencies, while joint execution synergy changes sign with dependency structure and capability gap G.

The same logic explains the capability-gap result. The organizing variable is the capability gap G, not ordering alone. When $G < G^* \approx 0.25$, the pair is a weak tie: neither step provides a strong enough anchor for the other, and joint execution shows a shallow drag that increases toward zero as the gap grows. When $G \geq G^*$, the pair becomes a strong tie: the stronger step provides a usable scaffold for the weaker one, producing a positive gain that rises quickly and then saturates (Fig. 6 (c)). Thus the effect is not a simple strong-first vs. weak-first ordering result; it is a thresholded capability-gap law, with weak-tie drag below G^* and strong-tie promotion above G^*.

Wrong-state propagation and tie-dependent synergy. Wrong upstream state propagates in proportion to dependency strength: tight dependencies carry wrong state forward, while loose dependencies can ignore it and even show a small gain. Joint-execution synergy then changes sign with capability gap G:

$$S(G) \approx \begin{cases} -0.0775 + 0.31\,G & G < G^* \approx 0.25 \quad \text{(Weak-Tie)} \\ +0.265\,(1 - e^{-(G-G^*)/0.12}) & G \geq G^* \quad \text{(Strong-Tie)} \end{cases} \tag{4}$$

6 A Unified Law Structure for Skill Libraries

The previous two sections establish the routing and execution laws separately. The remaining question is whether the two laws share a state variable. They do: the routing slope b is not only the rate at which a model loses accuracy as the library grows; it also determines how often correct upstream state is available for execution-side rescue. Concretely, take a model whose routing accuracy at exposed library size N is $Acc(A|N) = a - b \ln N$. This probability also gates whether the downstream step receives useful realized state. Substituting $Acc(A|N)$ into the independently measured rescue equation from Sec. 5 gives $Acc_{\text{res}}(B|N) = Acc(B|N) + 2\alpha\,(1 - Acc(B|N))\,(a - b \ln N)$, where a, b come from routing experiments, and $2\alpha \approx 0.76$ is the pooled slope across all 1,690 rescue pairs. Yet the prediction tracks independently measured rescue gains across 15 models (ρ=0.74, p<0.005; Fig. 7). This is the unifying law-level result: b acts as an empirical state variable for skill-library scaling. Reducing it improves immediate routing accuracy and preserves the correct state needed for downstream rescue.

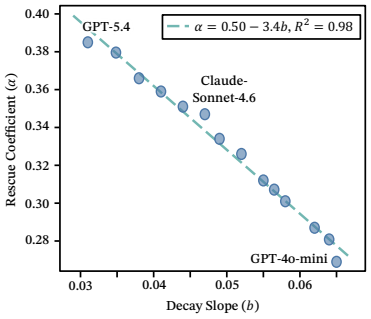
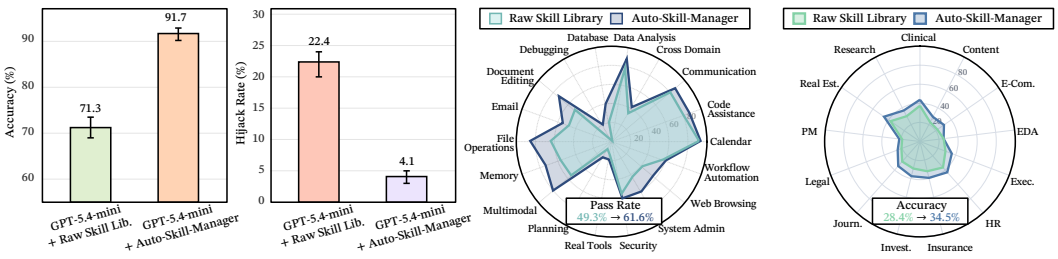


Figure 7: Cross-law coupling: routing slope predicts rescue gain.

This link also explains why the design optimization in Sec. 7 targets library structure rather than only prompt formatting. Boundary rewriting, nearest-neighbor auditing, and abstract-skill removal all reduce probability mass lost to local competition or attractor capture. In this unified law view, those changes are not separate heuristics: they lower b, delaying the routing cascade and increasing the probability that execution state can help later steps.



(a) The accuracy comparison between raw skill library and skills optimized by auto-skill-manager on held-out test set.

(b) The hijack rate comparison between raw skill library and skills optimized by auto-skill-manager on held-out test set.

(c) The pass rate comparison between raw skill library and skills optimized by auto-skill-manager on ClawBench.

(d) The pass rate comparison between raw skill library and skills optimized by auto-skill-manager on ClawMark.

Figure 8: The performance of the law-guided automatic skill manager.

7 Law-Guided Auto Skill Manager

The laws make a falsifiable design prediction: if failures are governed by library geometry rather than only by model capability, then editing that geometry should improve a fixed router on held-out tasks. Our law-guided auto skill manager implements this prediction by auditing the exposed library, estimating where routing probability mass is being lost, and editing the library while holding the router fixed. Local competition triggers nearest-neighbor audit and boundary rewrite; cross-cluster drift triggers prompt anchoring; black-hole capture triggers abstract-skill removal or narrowing; and the execution law constrains edits to preserve correct upstream state for downstream rescue. We evaluate prospectively at $N{=}150$, near the law-predicted transition point for median-b agents, with held-out tasks, gold labels, distractors, model, prompt template, and parser fixed before optimization.

The prediction holds on both the held-out benchmark and downstream agent benchmarks. On 1,600 held-out tasks, the manager raises accuracy from 71.3% to 91.7% and reduces hijack from 22.4% to 4.1% (Fig. 8 (a, b)). The optimized library transfers to downstream tasks: under the downstream setting, mean pass rate improves from 28.4% to 34.5% on CLAWMARK [78] and from 49.3% to 61.6% on CLAWBENCH [79] (Fig. 8 (c, d); Append. F). Thus the manager is not just a diagnostic cleanup pass, which can also improve routing and positive downstream transfer. Append. E.1–E.5 provide more analysis and discussion.

8 Discussion

These are empirical laws for flat natural-language skill libraries under availability-preserving evaluation: the required skill is present, so the central error is in-library misselection rather than missing capability. Within this setting, the laws identify concrete levers: report in-library hijack separately from hallucination, sweep library size N, include weak-anchor prompts, and restructure libraries before deployment. For deployed agents the benefit is safer, more auditable skill exposure, and the risk is that the same diagnostics make harmful automation more reliable, so deployments need access controls and audit logs. The exact slopes may change with learned routers, fine-tuning, or hierarchical libraries, although the decay replicates beyond the curated library (Appendix C.7). Two caveats bound the laws: coefficients are not geometry-free, since a library whose additions crowd the target decays faster than one whose additions are unrelated (Appendix C.4), and the downstream transfer uses a single run per task.

9 Conclusion

This paper identifies empirical scaling laws of skill libraries in settings where the needed skill is available but may be misselected. As reusable skills scale, behavior is shaped not only by model capability but also by library structure. Before execution, routing accuracy follows a logarithmic law; after correct execution, realized state can rescue difficult downstream decisions. A single parameter, the routing decay slope b, links these regimes by predicting both routing collapse and execution-side rescue. We instantiate this coupling in a Law-Guided Auto Skill Manager, showing that the laws can drive practical library optimization with held-out and downstream validation. Large skill libraries should therefore be designed as structured systems, where naming, granularity, exposure, and execution state determine whether scale expands capability or amplifies failure.

References

[1] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024.

[2] Weikai Xu, Chengrui Huang, Shen Gao, and Shuo Shang. Llm-based agents for tool learning: A survey. *Data Science and Engineering*, 2025.

[3] Mengkang Hu, Yao Mu, Xinmiao Yu, Mingyu Ding, Shiguang Wu, Wenqi Shao, Qiguang Chen, Bin Wang, Yu Qiao, and Ping Luo. Tree-planner: Efficient close-loop task planning with large language models. In *International Conference on Learning Representations*, volume 2024, pages 48669–48696, 2024.

[4] Mengkang Hu, Yuhang Zhou, Wendong Fan, Yuzhou Nie, Bowei Xia, Tao Sun, Ziyu Ye, Zhaoxuan Jin, Yingru Li, Qiguang Chen, et al. Owl: Optimized workforce learning for general multi-agent assistance in real-world task automation. *arXiv preprint arXiv:2505.23885*, 2025.

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

[6] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world APIs. *arXiv preprint arXiv:2307.16789*, 2023.

[7] 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.

[8] Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-Bank: A comprehensive benchmark for tool-augmented LLMs. *arXiv preprint arXiv:2304.08244*, 2023.

[9] Yongliang Shen, Kaitao Song, Xu Tan, Wenqi Zhang, Kan Ren, Siyu Yuan, Weiming Lu, and Yueting Zhuang. TaskBench: Benchmarking large language models for task automation. *arXiv preprint arXiv:2311.18760*, 2023.

[10] Bytedance Seed. Seed2. 0 model card: Towards intelligence frontier for real-world complexity. *arXiv preprint arXiv:2607.00248*, 2026.

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

[12] 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.

[13] Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*, 2025.

[14] Qiguang Chen, Mingda Yang, Libo Qin, Jinhao Liu, Zheng Yan, Jiannan Guan, Dengyun Peng, Yiyan Ji, Hanjing Li, Mengkang Hu, et al. Ai4research: A survey of artificial intelligence for scientific research. *arXiv preprint arXiv:2507.01903*, 2025.

[15] Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, et al. A survey of self-evolving agents: What, when, how, and where to evolve on the path to artificial super intelligence. *arXiv preprint arXiv:2507.21046*, 2025.

[16] Dengyun Peng, Yuhang Zhou, Qiguang Chen, JinHao Liu, Jingjing Chen, Libo Qin, and Wanxiang Che. DLPO: Towards a robust, efficient, and generalizable prompt optimization framework from a deep-learning perspective. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 8311–8334, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.441. URL https://aclanthology.org/2025.findings-emnlp.441/.

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

[18] Libo Qin, Qiguang Chen, Xiachong Feng, Yang Wu, Yongheng Zhang, Yinghui Li, Min Li, Wanxiang Che, and Philip S Yu. Large language models meet nlp: A survey. *Frontiers of Computer Science*, 20(11):2011361, 2026.

[19] Yanna Jiang, Delong Li, Haiyu Deng, Baihe Ma, Xu Wang, Qin Wang, and Guangsheng Yu. SoK: Agentic skills – beyond tool use in LLM agents. *arXiv preprint arXiv:2602.20867*, 2026.

[20] Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. *arXiv preprint arXiv:2306.05301*, 2023.

[21] Tianbao Xie, Fan Zhou, Zhoujun Cheng, Peng Shi, Luoxuan Weng, Yitao Liu, Toh Jing Hua, Junning Zhao, Qian Liu, Che Liu, et al. OpenAgents: An open platform for language agents in the wild. *arXiv preprint arXiv:2310.10634*, 2023.

[22] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. AgentBench: Evaluating LLMs as agents. *arXiv preprint arXiv:2308.03688*, 2023.

[23] Asaf Yehudai, Lilach Eden, Alan Li, Guy Uziel, Yilun Zhao, Roy Bar-Haim, Arman Cohan, and Michal Shmueli-Scheuer. Survey on evaluation of LLM-based agents. *arXiv preprint arXiv:2503.16416*, 2025.

[24] Lucen Zhong, Zhengxiao Du, Xiaohan Zhang, Haiyi Hu, and Jie Tang. ComplexFuncBench: Exploring multi-step and constrained function calling under long-context scenario. *arXiv preprint arXiv:2501.10132*, 2025.

[25] Thierry Blankenstein, Jialin Yu, Zixuan Li, Vassilis Plachouras, Sunando Sengupta, Philip Torr, Yarin Gal, Alasdair Paren, and Adel Bibi. BiasBusters: Uncovering and mitigating tool selection bias in large language models. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=DEg4vvElYu.

[26] Cheng Qian, Zuxin Liu, Shirley Kokane, Akshara Prabhakar, Jielin Qiu, Haolin Chen, Zhiwei Liu, Heng Ji, Weiran Yao, Shelby Heinecke, Silvio Savarese, Caiming Xiong, and Huan Wang. xRouter: Training cost-aware LLMs orchestration system via reinforcement learning. *arXiv preprint arXiv:2510.08439*, 2025.

[27] Guibin Zhang, Haiyang Yu, Kaiming Yang, Bingli Wu, Fei Huang, Yongbin Li, and Shuicheng Yan. EvoRoute: Experience-driven self-routing LLM agent systems. *arXiv preprint arXiv:2601.02695*, 2026.

[28] Jingyi Jia and Qinbin Li. Autotool: Efficient tool selection for large language model agents. *arXiv preprint arXiv:2511.14650*, 2025.

[29] Zheng Fang, Wolfgang Mayer, Zeyu Zhang, Jian Wang, Hong-Yu Zhang, Wanli Li, and Zaiwen Feng. Metatoolagent: Towards generalizable tool usage in llms through meta-learning. *arXiv preprint arXiv:2601.12680*, 2026.

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

[31] Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2609–2634, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.147. URL https://aclanthology.org/2023.acl-long.147/.

[32] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*, 2023.

[33] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving AI tasks with ChatGPT and its friends in Hugging Face. *arXiv preprint arXiv:2303.17580*, 2023.

[34] Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. MetaGPT: Meta programming for a multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*, 2023.

[35] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

[36] Bo Qiao, Liqun Li, Xu Zhang, Shilin He, Yu Kang, Chaoyun Zhang, Fangkai Yang, Hang Dong, Jue Zhang, Lu Wang, et al. TaskWeaver: A code-first agent framework. *arXiv preprint arXiv:2311.17541*, 2023.

[37] Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. MINT: Evaluating LLMs in multi-turn interaction with tools and language feedback. *arXiv preprint arXiv:2309.10691*, 2023.

[38] Yu Du, Fangyun Wei, and Hongyang Zhang. AnyTool: Self-reflective, hierarchical agents for large-scale API calls. *arXiv preprint arXiv:2402.04253*, 2024.

[39] Mahmoud Mohammadi, Yipeng Li, Jane Lo, and Wendy Yip. Evaluation and benchmarking of llm agents: A survey. *arXiv preprint arXiv:2507.21504*, 2025. URL https://arxiv.org/abs/2507.21504.

[40] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[41] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[42] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022.

[43] Nicholas Roberts, Niladri S. Chatterji, Sharan Narang, Mike Lewis, and Dieuwke Hupkes. Compute optimal scaling of skills: Knowledge vs reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 13295–13316, 2025. doi: 10.18653/v1/2025.findings-acl.688. URL https://aclanthology.org/2025.findings-acl.688/.

[44] Mengsong Wu, Hao Hao, Shuzhen Bi, Keqian Li, Wentao Liu, Siyu Song, Hongbo Zhao, and Aimin Zhou. Scaling laws for educational AI agents. *arXiv preprint arXiv:2603.11709*, 2026. URL https://arxiv.org/abs/2603.11709.

[45] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 6769–6781, 2020.

[46] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474, 2020.

[47] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 2019.

[48] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. MTEB: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, 2023.

[49] William E. Hick. On the rate of gain of information. *Quarterly Journal of Experimental Psychology*, 4(1):11–26, 1952.

[50] Ray Hyman. Stimulus information as a determinant of reaction time. *Journal of Experimental Psychology*, 45(3):188–196, 1953.

[51] Anthropic. Github - anthropics/skills: Public repository for agent skills. https://github.com/anthropics/skills, 2025. Accessed: 2026-04-17.

[52] Anthropic. Claude code. https://www.anthropic.com/product/claude-code, 2025. Accessed: 2026-04-16.

[53] Anthropic. Introducing the model context protocol. https://www.anthropic.com/news/model-context-protocol, 2024. Accessed: 2026-04-16.

[54] Trail of Bits. Github - trailofbits/skills: Trail of bits claude code skills for security research, vulnerability detection, and audit workflows. https://github.com/trailofbits/skills, 2025. Accessed: 2026-04-16.

[55] daymade. Claude code skills marketplace. https://github.com/daymade/claude-code-skills, 2025. Accessed: 2026-04-16.

[56] Aaronontheweb. .NET skills for claude code. https://github.com/Aaronontheweb/dotnet-skills, 2025. Accessed: 2026-04-16.

[57] PinchBench Team. PinchBench github organization. https://github.com/pinchbench, 2026. Accessed: 2026-04-16.

[58] Robhunter. Agentdeals. https://github.com/robhunter/agentdeals, 2026. Accessed: 2026-04-16.

[59] Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. M3-Embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2318–2335, 2024. doi: 10.18653/v1/2024.findings-acl.137. URL https://aclanthology.org/2024.findings-acl.137/.

[60] Mike A Merrill, Alexander G Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E Kelly Buchanan, et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. *arXiv preprint arXiv:2601.11868*, 2026.

[61] Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

[62] Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*, 2025.

[63] Ruixuan Xiao, Wentao Ma, Ke Wang, Yuchuan Wu, Junbo Zhao, Haobo Wang, Fei Huang, and Yongbin Li. FlowBench: Revisiting and benchmarking workflow-guided planning for LLM-based agents. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10883–10900, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.638. URL https://aclanthology.org/2024.findings-emnlp.638/.

[64] OpenAI. Gpt-4o mini: advancing cost-efficient intelligence. https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/, 2024. Accessed: 2026-04-17.

[65] OpenAI. Gpt-5 system card. OpenAI, August 2025. URL https://openai.com/index/gpt-5-system-card/. Accessed: 2026-05-04.

[66] OpenAI. Introducing gpt-5.4 mini and nano. https://openai.com/index/introducing-gpt-5-4-mini-and-nano/, 2026. Accessed: 2026-04-17.

[67] OpenAI. Introducing gpt-5.4. https://openai.com/index/introducing-gpt-5-4/, 2026. Accessed: 2026-04-17.

[68] Anthropic. Introducing claude sonnet 4.6. https://www.anthropic.com/news/claude-sonnet-4-6, 2026.

[69] Anthropic. Introducing claude opus 4.6. https://www.anthropic.com/news/claude-opus-4-6, 2026.

[70] Google DeepMind. Gemini 3. *Google DeepMind*, 2025. URL https://deepmind.google/models/gemini/.

[71] Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering, 2026.

[72] Zhipu AI. Glm-4.7: Advancing the coding capability. https://z.ai/blog/glm-4.7, 2025.

[73] Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, SH Cai, Yuan Cao, Y Charles, HS Che, Cheng Chen, Guanduo Chen, et al. Kimi k2.5: Visual agentic intelligence, 2026.

[74] Moonshot AI. Kimi k2.6: From code to creation, from one to many. https://www.kimi.com/ai-models/kimi-k2-6, 2026.

[75] ByteDance Seed Team. Seed 2.0 official launch. https://seed.bytedance.com/en/blog/seed2-0-%25E6%25AD%25A3%25E5%25BC%258F%25E5%258F%2591%25E5%25B8%2583, 2026. Accessed: 2026-04-16.

[76] DeepSeek-AI. Deepseek-v4:towards highly efficient million-token context intelligence. https://huggingface.co/deepseek-ai/DeepSeek-V4-Pro/blob/main/DeepSeek_V4.pdf, 2026.

[77] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

[78] Fanqing Meng, Lingxiao Du, Zijian Wu, Guanzheng Chen, Xiangyan Liu, Jiaqi Liao, Chonghe Jiang, Zhenglin Wan, Jiawei Gu, Pengfei Zhou, et al. Clawmark: A living-world benchmark for multi-turn, multi-day, multimodal coworker agents. *arXiv preprint arXiv:2604.23781*, 2026.

[79] Yuxuan Zhang, Yubo Wang, Yipeng Zhu, Penghui Du, Junwen Miao, Xuan Lu, Zhuofeng Li, Xingwei Qu, Zhengkang Guo, Yuanzhe Shen, et al. Clawbench: Can ai agents complete everyday online tasks? In *Findings of the Association for Computational Linguistics: EMNLP 2026*, pages 9386–9408, 2026.

[80] Qiguang Chen, Libo Qin, Jiaqi Wang, Jinxuan Zhou, and Wanxiang Che. Unlocking the capabilities of thought: A reasoning boundary framework to quantify and optimize chain-of-thought. *Advances in Neural Information Processing Systems*, 37:54872–54904, 2024.

[81] Qiguang Chen, Libo Qin, Jinhao Liu, Yue Liao, Jiaqi Wang, Jinxuan Zhou, and Wanxiang Che. Rbf++: Quantifying and optimizing reasoning boundaries across measurable and unmeasurable capabilities for chain-of-thought reasoning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2026.

Appendix

This appendix is organized as a reader's map from theory to implementation evidence. It first states the conditional derivations behind the routing and execution laws (Appendix A), then gives the experimental protocol needed to interpret the measurements (Appendix B). The routing-law section reports the fitted decay parameters, the fine-grained law summary, mechanism diagnostics, retriever/router baselines, task-difficulty controls, boundary manipulations, and external validity checks (Appendices C.1–C.7). The execution-law section reports cross-model rescue robustness (Appendix D.5). The final two blocks turn these measurements into system evidence: Appendix E explains the automatic skill manager and its component checks, while Appendix F tests whether the law-guided library changes transfer to downstream skill-agent benchmarks.

A Mathematical Analysis for the Scaling Laws of Skill Libraries

This section gives the formal backbone behind the empirical laws. The results are conditional theorems: under explicit assumptions about finite-capacity selection, library-competition growth, and state realization, the observed laws follow. The empirical sections then test whether those assumptions are a good description of the measured skill libraries. This separates two claims: the proofs show what the theory implies; the experiments show that the theory's assumptions hold well enough in the studied regime.

We use N for library size, K for pipeline length, $Acc(N)$ for routing accuracy, $s^\star$ for the target skill, and $\hat{s}$ for the selected skill. The margin $\Delta\mu$ measures how clearly the target skill is separated from its nearest competitors; larger $\Delta\mu$ means easier routing. The competition index CI measures the total pressure from nearby distractor skills. For execution, A is the upstream step, B is the downstream step, and $\Delta Acc(B \mid A)$ is the downstream gain after correct upstream execution. The common mechanism is simple: larger candidate sets and smaller margins lower routing confidence, while correct execution state can restore useful context. The shared empirical state variable is the routing decay slope b.

Assumptions. We use the following assumptions throughout the formal statements.

1. **Finite-capacity margin model.** For task q, the router assigns each skill a latent score $Z_i = m_i + \varepsilon_i$, where $m_\star - m_j = \Delta_j$ is the semantic margin of the gold skill against distractor j, and ε_i is zero-mean sub-Gaussian noise with scale σ.
2. **Rank-regular clustered library.** Within a functional cluster, distractor skills can be ordered by semantic rank $r = 1, \ldots, N-1$ around the target, and their effective overtake weights satisfy $w_r = \kappa/r + O(r^{-1-\xi})$ for some $\kappa, \xi > 0$ over the measured range. This is the scale-free local-crowding condition tested by the CI and semantic-gap diagnostics.
3. **Logarithmic effective competition.** The total effective pressure from plausible distractors is $C(N) = \sum_{j \neq \star} w_j(N) = C_0 + C_1 \ln N + o(\ln N)$ over the exposed-library range. Irrelevant distractors have negligible w_j; local near-misses dominate $C(N)$.
4. **Small-error linearization regime.** Over the measured operating range, the router is not saturated at 0 or 1, so first-order Taylor expansions of the error odds are valid.
5. **State-gated rescue.** Correct upstream execution produces a concrete artifact with rescue coefficient $\alpha \in [0, 1]$; rescue can only occur when upstream routing/execution is correct and only to the extent that the downstream route has remaining headroom.
6. **No pre-state leakage.** In the no-state condition, paired routing prompts do not contain an execution artifact, so any interaction before execution is bounded by a protocol term ϵ_{proto}.

Lemma 1 (Clustered libraries imply logarithmic effective competition). Under Assumption 2, Assumption 3 follows with $C_1 = \kappa$.

Proof. By Assumption 2,

$$C(N) = \sum_{r=1}^{N-1} w_r = \kappa \sum_{r=1}^{N-1} \frac{1}{r} + \sum_{r=1}^{N-1} O(r^{-1-\xi}). \tag{5}$$

The harmonic sum satisfies $\sum_{r=1}^{N-1} 1/r = \ln N + \gamma_E + o(1)$, where γ_E is the Euler constant. The residual series $\sum_r O(r^{-1-\xi})$ converges because $\xi > 0$, so it contributes a constant plus $o(1)$.

Table 1: Fine-grained appendix laws underlying the grouped routing-law and execution-law summaries in the main text.

Regime	Law	Variable	Observed Law & Pattern
Routing	Law 1: Logarithmic Decay	Capacity-limited selection	$a - b \ln N$
Routing	Law 2: Pipeline Compounding	Local errors corrupt downstream context	$(a - b \ln N)^{\gamma K}$
Routing	Law 3: Pipeline Rebound	Information compresses at mid-chain	$\delta(N) = \delta_0 + c_\delta \ln N$
Routing	Law 4: Description Quality	Quality changes effective semantic margin	$a_\ell - b_\ell \ln N$
Routing	Law 5: Local Competition	Confusion peaks at intermediate similarity	Peak at $[0.55, 0.75)$
Routing	Law 6: Failure Geometry	Description scope induces directional confusion	Local family substitution
Routing	Law 7: Anchor Removal	Vague prompts reduce semantic separation	Cross-cluster drift
Routing	Law 8: Dual-Trigger Black Hole	Abstract skills become competitive near weak anchors	Conjunctive attractor capture
Execution	Law 9: Routing Independence	No state $\Rightarrow$ multiplicative routing	$Acc(A, B) \approx Acc(A)Acc(B)$
Execution	Law 10: Execution Rescue	Realized artifacts increase concreteness	$\Delta Acc(B\|A) \propto (1 - Acc(B))Acc(A)$
Execution	Law 11: Context-Conditioned Recovery	Coupling type determines harm vs. help	$-0.072\kappa + 0.028(1 - \kappa)$
Execution	Law 12: Strong-Tie Promotion / Weak-Tie Drag	Capability gap governs sign and saturation of joint synergy	Piecewise $S(G)$ with $G^* \approx 0.25$

Therefore

$$C(N) = C_0 + \kappa \ln N + o(\ln N), \tag{6}$$

which is Assumption 3. □

Proposition 1 (Routing law from finite-capacity competition). Under Assumptions 1–4, there exist constants a and $b > 0$ such that

$$Acc(N) = a - b \ln N + o(\ln N) \tag{7}$$

over the measured library range. This is a finite-range local approximation, not a universal asymptotic claim. Moreover, increasing semantic margins decreases b.

Proof. The gold skill is selected when $Z_\star > Z_j$ for all distractors. For each distractor, define its pairwise overtake probability

$$r_j = \Pr[Z_j \geq Z_\star] = \Pr[\varepsilon_j - \varepsilon_\star \geq \Delta_j]. \tag{8}$$

Sub-Gaussian concentration gives $r_j \leq \exp(-\Delta_j^2/(4\sigma^2))$ up to a constant factor. In the measured operating range, we assume the effective competition weights are comparable to these pairwise overtake probabilities, so that the same local crowding index controls both the upper bound and the observed error pressure. Write this effective pressure as w_j. By the union bound and the usual inclusion–exclusion expansion,

$$\Pr[\exists j : Z_j \geq Z_\star] = \sum_{j \neq \star} r_j + O\left((\sum_{j \neq \star} r_j)^2 \right). \tag{9}$$

Assumption 3 and the comparability condition give $\sum_j r_j = \lambda_0 + \lambda_1 \ln N + o(\ln N)$ after absorbing constant factors into λ_0, λ_1. Assumption 4 makes the second-order term bounded and locally linearizable in the measured range, so

$$Acc(N) = 1 - \Pr[\exists j : Z_j \geq Z_\star] = a - b \ln N + o(\ln N). \tag{10}$$

If descriptions or skill boundaries increase margins Δ_j, the effective overtake pressures decrease under the same comparability condition, so the log-growth coefficient λ_1 and hence b decrease. This proves the local logarithmic form and the design prediction that sharpening boundaries reduces the slope in the measured regime. □

Lemma 2 (Boundary interventions identify semantic competition). Consider two candidate mechanisms for the decay slope:

$$b = b_{\mathrm{ctx}}(L_N) + b_{\mathrm{sem}}(\{\Delta_j\}), \tag{11}$$

where b_{ctx} depends only on prompt length or number of exposed tokens L_N, and b_{sem} depends on semantic margins. A boundary rewrite that preserves token length and candidate set but increases margins Δ_j can reduce b only through b_{sem}. Therefore, a statistically reliable decrease in b under length-matched boundary rewrites rules out a pure context-length explanation $b = b_{\mathrm{ctx}}(L_N)$.

Proof. In the intervention, L_N and the exposed candidate set are fixed by construction, so $b_{\mathrm{ctx}}(L_N)$ is invariant. The only changed argument in the decomposition is the margin set $\{\Delta_j\}$. By Proposition A, increasing margins decreases the effective overtake pressures under the finite-range comparability condition, and hence decreases b_{sem}. Thus any observed slope change under a length-matched boundary rewrite is identified as semantic-competition movement within this additive model. If b were purely a function of context length, the treatment effect would be exactly zero. □

Proposition 2 (Pipeline compounding from context compression). Let $p_N = a - b \ln N$ be the isolated single-step routing probability. In a route-only pipeline, suppose each downstream step after the first pays a compression penalty $0 \leq \eta_N < p_N$ because the prompt contains an intermediate plan state rather than the full original task anchor. Then strict all-correct pipeline success is

$$Acc(N, K) = p_N (p_N - \eta_N)^{K-1}. \tag{12}$$

If $\eta_N > 0$, then $Acc(N, K) < p_N^K$, so pipeline success falls below independent repetition. If $\eta_N / p_N = O(b)$, then

$$Acc(N, K) = p_N^K \exp\left[-(K-1)\eta_N / p_N + O(K(\eta_N / p_N)^2)\right], \tag{13}$$

which can be reparameterized as the empirical exponent form $Acc(N, K) \approx p_N^{\gamma K}$ with $\gamma > 1$. The fitted γ increases with routing fragility when the relative compression penalty η_N / p_N increases sufficiently with b.

Proof. Let $\delta_k = 1$ denote a correct route. The first step sees the original task anchor, so $\Pr(\delta_1 = 1) = p_N$. Each later step is routed from a compressed intermediate context, so

$$\Pr(\delta_k = 1 \mid \delta_1 = \cdots = \delta_{k-1} = 1) = p_N - \eta_N, \qquad k \geq 2. \tag{14}$$

Multiplying conditional probabilities gives $p_N (p_N - \eta_N)^{K-1}$. Since $p_N - \eta_N < p_N$ when $\eta_N > 0$, the product is strictly below p_N^K. For the exponential form,

$$p_N (p_N - \eta_N)^{K-1} = p_N^K (1 - \eta_N / p_N)^{K-1}. \tag{15}$$

Using $\ln(1-x) = -x + O(x^2)$ gives the displayed expansion. Finally, because $p_N^{\gamma K} = p_N^K \exp((\gamma - 1) K \ln p_N)$ and $\ln p_N < 0$, the additional exponential loss corresponds to some $\gamma > 1$. Cross-model monotonicity of γ in b is therefore an empirical regularity unless one additionally assumes that the growth of η_N / p_N dominates the change in $\ln p_N$. □

Proposition 3 (Execution rescue). Under Assumption 5, the downstream improvement from correct upstream state has the form

$$\Delta Acc(B \mid A) = 2\alpha \, Acc(A) \, (1 - Acc(B)), \qquad \alpha := \beta / 2. \tag{16}$$

Here α is the rescue coefficient plotted in Fig. 7 (range 0.28–0.40), and the measured slope in Fig. 6(a) is $2\alpha \approx 0.76$; the factor of two is therefore a rescaling between the plotted coefficient and the regressed slope.

Proof. Rescue requires two events: correct upstream state exists, and downstream routing has remaining headroom. The first event has probability $Acc(A)$. Conditional on useful state, the largest possible downstream improvement is bounded by $1 - Acc(B)$, because routes already correct before state cannot improve. Let α be the fraction of this available headroom converted into correct downstream decisions by the artifact. Multiplying the gate $Acc(A)$, the available headroom $1 - Acc(B)$, and the conversion coefficient α yields

$$\Delta Acc(B \mid A) = \alpha\, Acc(A)\,(1 - Acc(B)). \tag{17}$$

The factor 2 that appears in the empirical fit, in the statement of this proposition, and in Theorem A is a notational rescaling rather than an extra mechanism: the regressed slope is $2\alpha \approx 0.76$, so the per-unit-conversion coefficient is $\alpha \approx 0.38$, matching the rescue coefficient plotted in Fig. 7. We write the fitted slope as 2α throughout to keep the figure axis and the regression table on one scale.□

Theorem 1 (Cross-law coupling through the routing slope). Under Assumptions 1–5 and Propositions A–A, the execution-side rescued downstream accuracy satisfies

$$Acc_{\text{res}}(B) = Acc(B) + 2\alpha(1 - Acc(B))(a - b\ln N) + o(\ln N). \tag{18}$$

Thus, holding a, N, α, and $Acc(B)$ fixed, larger routing decay slope b strictly decreases the expected execution rescue.

Coupling diagnostic details. The coupling check deliberately uses two measurements that are estimated from different experimental regimes. For each model, the routing-side quantity is the slope b from the single-step library-size sweep in Appendix C.1. The execution-side quantity is the rescue coefficient α from the correct-state execution probes, estimated by regressing downstream rescue on upstream correctness and downstream headroom. The same ordered skill pairs used for rescue estimation are not used to refit the routing slope.

After these two quantities are fixed for all 15 models, we run a cross-model diagnostic regression of α on b and obtain $\alpha = 0.50 - 3.4b$. The purpose is not to add a new free parameter to the main law, but to test whether the independently measured routing fragility of a model predicts how much correct execution state can rescue later decisions. We therefore interpret the relation as a no-refit consistency check linking the routing and execution experiments. The derivative in the theorem is a partial derivative with α held fixed; if one instead treats the empirical diagnostic $\alpha = \alpha(b)$ as a structural relation, the total derivative contains the additional term $\alpha'(b)(a - b\ln N)$.

Proof. By Proposition A, the probability that the upstream route supplies the correct state is $Acc(A) = a - b\ln N + o(\ln N)$. Substitute this into Proposition A:

$$\Delta Acc(B \mid A) = 2\alpha(1 - Acc(B))(a - b\ln N) + o(\ln N). \tag{19}$$

Adding this improvement to the pre-state downstream accuracy $Acc(B)$ gives the stated equation. Differentiating with respect to b gives

$$\frac{\partial Acc_{\text{res}}(B)}{\partial b} = -2\alpha(1 - Acc(B))\ln N < 0 \tag{20}$$

for $\alpha > 0$, $Acc(B) < 1$, and $N > 1$. Hence the same slope that governs routing collapse also controls the expected gate through which correct upstream state can rescue downstream routing. □

L1: logarithmic decay from capacity-limited selection. L1 is the direct consequence of Proposition A. A complementary finite-range view comes from an extreme-value approximation. Let the target skill score be $z^\star = \langle \mathbf{u}, \tilde{\mathbf{v}}^\star \rangle$ and each competing score be $z_j = \langle \mathbf{u}, \tilde{\mathbf{v}}_j \rangle$, where $\tilde{\mathbf{v}}_i = \mathbf{v}_i + \boldsymbol{\epsilon}_i$ is the noisy representation available to a finite-capacity router. If the nearest semantic margin is $\Delta\mu$ and the effective noise variance is $\sigma_0^2 f(N)$, then

$$Acc(N) = \Pr[z^\star > \max_{j \neq \star} z_j] \approx \Phi\left(\frac{\Delta\mu - \sigma_0\sqrt{f(N)}\sqrt{2\ln N}}{\sigma_0\sqrt{f(N)}}\right). \tag{21}$$

This expression is not used as a global asymptotic equality. In the finite range $N \in \{10, \ldots, 500\}$, define $x = \ln N$ and write the right-hand side as $F(x)$. Taylor expansion around the center of the sweep x_0 gives

$$F(x) = F(x_0) + F'(x_0)(x - x_0) + O((x - x_0)^2) = a - b\ln N + O((\ln N - \ln N_0)^2), \tag{22}$$

where $b = -F'(x_0) > 0$ in the measured decreasing regime. The derivative $F'(x_0)$ still depends on the local value of $f(N)$ and on the expansion point; we use this approximation only to justify a local slope. The monotone margin prediction comes from Proposition A: larger semantic separation lowers effective overtake pressure and therefore lowers the fitted log-slope in the measured regime. Thus the proof claim is local: finite-capacity extreme competition plus observed log-scale crowding yields the measured $a - b \ln N$ form.

Assumption notes. The scaling $f(N) \propto \ln N$ is an empirical approximation, motivated by the sub-linear growth of CI(N) in our 14-cluster library and supported by the AIC comparison (ΔAIC>8 across all 15 models). The Gaussian approximation treats noise terms ϵ_i as independent; this is imperfect for semantically similar skills, but the resulting prediction is supported by the per-skill CI regression (R^2=0.55, Sec. 4) and by the cross-model log fits ($R^2>0.96$ for all models). If added skills were mostly irrelevant, CI would not grow with N and the log slope would collapse toward zero; if the effect were pure context length, description rewrites would not selectively reduce b for crowded neighborhoods. Both alternatives are contradicted by the semantic-gap regression and the boundary manipulation in Appendix C.5.

L2: pipeline compounding and cascade amplification. L2 follows from Proposition A. Let $\delta_k \in \{0, 1\}$ denote whether step k is routed correctly, and let $p_N = Acc(N)$. The independence baseline is

$$P_{\text{ind}}(N, K) = \prod_{k=1}^{K} \Pr(\delta_k = 1) = p_N^K. \tag{23}$$

In a route-only plan, later prompts are conditioned on compressed planner state. Model that loss by

$$\Pr(\delta_k = 1 \mid \delta_1 = \cdots = \delta_{k-1} = 1) = p_N - \eta_N, \qquad k \geq 2, \tag{24}$$

with $0 \leq \eta_N < p_N$. Multiplying the chain-rule factors gives

$$Acc(N, K) = p_N (p_N - \eta_N)^{K-1}, \qquad p_N = Acc(N), \quad 0 \leq \eta_N < p_N, \tag{25}$$

and hence

$$\frac{Acc(N, K)}{P_{\text{ind}}(N, K)} = \left(1 - \frac{\eta_N}{p_N}\right)^{K-1} < 1 \tag{26}$$

whenever $\eta_N > 0$. Taking logs,

$$\ln Acc(N, K) = K \ln p_N + (K-1) \ln(1 - \eta_N/p_N) = K \ln p_N - (K-1)\eta_N/p_N + O(K\eta_N^2/p_N^2). \tag{27}$$

Equivalently, define

$$\gamma = 1 + \frac{K-1}{K} \frac{\ln(1 - \eta_N/p_N)}{\ln p_N}, \tag{28}$$

which is larger than one because both logarithms are negative. Then $Acc(N, K) = p_N^{\gamma K}$ exactly for fixed N, K, η_N. The empirical scaling law uses this as a low-dimensional summary across the measured N range. The derivation is what establishes compounding below the independence baseline.

L3: mid-chain fragility from information compression. L3 follows by adding terminal recovery to the same compression model. Let I_k be the mutual information between the original user intent and the context at step k. Use

$$I_k = I_0 - c_0(k-1) \ln N + r_k, \tag{29}$$

where $c_0 \ln N$ is per-step identification loss and r_k is a terminal-constraint recovery term. Assume r_k is negligible before the final segment and increases near the end. For the middle step $m = \lfloor K/2 \rfloor$,

$$I_1 - I_m = c_0(m-1) \ln N - (r_m - r_1) \approx c_0(m-1) \ln N. \tag{30}$$

For a late step K, the terminal constraint contributes $r_K > r_m$, so

$$I_K - I_m = c_0(m-K) \ln N + (r_K - r_m). \tag{31}$$

A rebound occurs when $r_K - r_m > c_0(K-m) \ln N$; its depth is the recovered accuracy gap between the middle and late steps. In the finite range, write the relative terminal recovery as $r_K - r_m = r_0 + r_1 \ln N$, where r_1 captures the fact that final-step constraints become more valuable when the intermediate library ambiguity is larger. If local routing accuracy satisfies $Acc_k \simeq \beta_0 + \beta_1 I_k$, then the rebound depth is

$$\delta(N) = Acc_K - Acc_m \simeq \beta_1 r_0 + \beta_1 \{r_1 - c_0(K-m)\} \ln N. \tag{32}$$

Thus $\delta(N) = \delta_0 + c_\delta \ln N$ with $c_\delta > 0$ whenever $r_1 > c_0(K-m)$. The proof is conditional on the explicit recovery term; monotonicity alone would not imply the rebound.

L4: description quality as effective semantic margin. L4 follows from the margin dependence in Proposition A. Model the description-quality effect as a saturating margin gain:

$$\Delta\mu(\ell) = \Delta\mu_0 \frac{\ell}{\ell + c}, \tag{33}$$

where ℓ is description quality and the denominator says that each extra detail helps less after the main boundary is already clear. For a distractor with margin $\Delta_j(\ell)$, the sub-Gaussian bound gives

$$r_j(\ell) \le \exp\{-\Delta_j(\ell)^2/(4\sigma^2)\}. \tag{34}$$

An upper bound alone does not identify the derivative of r_j. We therefore use the same finite-range comparability condition as Proposition A: in the measured regime, the effective overtake pressure is

$$r_j(\ell) = \theta_j(\ell) \exp\{-\Delta_j(\ell)^2/(4\sigma^2)\}, \tag{35}$$

where $\theta_j(\ell)$ is bounded and does not increase fast enough to offset margin gain. Then

$$\frac{\partial \log r_j}{\partial \ell} = \frac{\partial \log \theta_j}{\partial \ell} - \frac{\Delta_j(\ell)\Delta_j'(\ell)}{2\sigma^2} < 0 \tag{36}$$

whenever the margin term dominates. Since the routing slope $b(\ell)$ is the local log-growth coefficient of $\sum_j r_j(\ell)$ in Proposition A, this gives $b'(\ell) < 0$ in that regime. The induced routing curve is

$$Acc(N, \ell) \approx a(\ell) - b(\ell) \ln N, \qquad b'(\ell) < 0, \tag{37}$$

with $a(\ell)$ allowed to move slightly because clearer descriptions can also improve small-library routing. Thus the improvement is derived through lower overtake pressure, not by multiplying the whole accuracy curve by a description-quality factor.

L5: danger zone and non-monotonic competition. L5 can be derived by separating semantic similarity from explicit disambiguation. Let $d = \Delta\mu_{ij}$ be the distance between two candidate descriptions. Write confusion pressure as

$$\Pr(i \to j) \propto g(d) \exp\left(-d^2/2\sigma^2\right), \tag{38}$$

where the Gaussian term is semantic substitutability and $g(d)$ is the probability that the candidate survives relevance and duplicate checks. Assume $g(0) = 0$ because exact or near duplicates trigger explicit boundary checking, $g'(d) > 0$ for small d, and $g(d)$ is positive and differentiable on $(0, \infty)$, bounded, and not exponentially increasing. We also assume the duplicate-check release is steep enough near zero that $g'(d)/g(d) > d/\sigma^2$ for sufficiently small positive d. The derivative of log confusion pressure is

$$\frac{d}{dd} \log \Pr(i \to j) = \frac{g'(d)}{g(d)} - \frac{d}{\sigma^2}. \tag{39}$$

For small $d > 0$, the release condition makes confusion rise. For large d, the boundedness condition makes the Gaussian tail dominate, so confusion tends to zero. By continuity, the maximum is attained at an interior point $d^\star$; any differentiable interior maximizer satisfies the first-order condition $g'(d^\star)/g(d^\star) = d^\star/\sigma^2$. This gives the non-monotonic danger zone under the duplicate-check model: plausible near-misses are worse than both irrelevant skills and obvious duplicates.

L6: descriptor geometry and asymmetric confusion. L6 follows from a set-overlap model of descriptor scope. Treat description d_i as defining a task region C_i. The overlap term is

$$J(C_i, C_j) = \frac{|C_i \cap C_j|}{|C_i \cup C_j|}, \tag{40}$$

and the directional asymmetry term is

$$A(d_i, d_j) = \frac{|C_i \setminus C_j| - |C_j \setminus C_i|}{|C_i \cup C_j|} \tag{41}$$

with $|A| \le 1$. This is a signed coverage-imbalance term, not an overlap measure. Define mutual-confusion risk as

$$Risk(i, j) \propto J(C_i, C_j)\big(1 - |A(d_i, d_j)|\big). \tag{42}$$

Then $\partial Risk/\partial J > 0$ when $|A| < 1$, and $\partial Risk/\partial |A| = -J < 0$. Thus high overlap increases confusion, while directional ownership decreases it. Boundary rewrites reduce risk either by lowering $J(C_i, C_j)$ or by increasing $|A|$ through clearer scope boundaries.

L7–L8: anchor removal and black-hole capture. L7–L8 explain why vague prompts and broad skills are dangerous mainly together. A concrete prompt points toward a specific skill, so the correct skill keeps a large margin. A vague prompt points toward the center of a functional cluster instead. If the library contains a broad abstract skill near that center, the broad skill can become the easiest explanation of the prompt:

$$\mathrm{score}(s_{\mathrm{abs}}) \approx \frac{1}{M}\sum_{i=1}^{M}\langle \mathbf{u}_{\mathrm{vague}}, \mathbf{v}_i\rangle + \epsilon, \tag{43}$$

where M is the number of concrete skills whose intent the abstract skill subsumes. Vague prompts alone need not cause capture, and abstract skills alone need not cause capture. The failure appears when both push probability mass toward the same broad skill.

Anchor-loss and black-hole experiment details. The anchor-loss experiment starts from matched task intents and progressively removes information that normally identifies the target skill. Query-anchor removal deletes or generalizes concrete nouns, file names, API names, schema fields, and domain-specific verbs while preserving the intended operation. Skill-anchor removal weakens the candidate descriptions by removing boundary clauses or replacing specific operational constraints with broader functional language. Because the task intent and gold skill are retained, a drop in accuracy measures loss of routing evidence rather than a change in what the task requires.

For every in-library error, we record both the gold functional category and the chosen wrong category. This lets us distinguish three regimes: local substitution within the same cluster, cross-cluster drift, and capture by a broad abstract skill. The semantic-anchor removal sweep crosses library size with removal level and measures hijack rate, keeping hallucinated skill names separate from in-library errors. The black-hole analysis then retains a broad catch-all skill in the exposed pool and measures top-1 absorption, routing-mass Gini, and hijack rate as the library grows. Finally, the activation test crosses two factors, skill-anchor removal and vague query construction. We treat capture as a dual-trigger effect only if the combined condition exceeds both single-factor controls; this is the empirical counterpart of Proposition 4.

Proposition 4 (Dual-trigger capture). Let the gold-skill score margin over an abstract skill be

$$M = M_0 - \Delta_q - \Delta_s, \tag{44}$$

where $M_0 > 0$ is the concrete-query/concrete-skill margin, $\Delta_q \geq 0$ is the margin loss from query-anchor removal, and $\Delta_s \geq 0$ is the margin loss from broad abstract skill descriptions. If capture occurs when $M < 0$, then there exist regimes where neither trigger alone causes capture but both together do:

$$M_0 - \Delta_q > 0, \qquad M_0 - \Delta_s > 0, \qquad M_0 - \Delta_q - \Delta_s < 0. \tag{45}$$

Proof. The three inequalities are jointly feasible whenever $\max(\Delta_q, \Delta_s) < M_0 < \Delta_q + \Delta_s$. In that interval, each individual manipulation leaves the gold skill ahead of the abstract skill, but their combined margin loss reverses the ordering. Thus capture is an interaction term in margin space rather than a necessary consequence of either vague prompts or abstract skills alone. □

L9: pre-state routing independence. L9 is the clean baseline for execution: before anything runs, there is no artifact from step A that can help step B. Let A and B be the events that the upstream and downstream routes are correct. Once we condition on the two task prompts and the exposed library, the downstream route has no extra state from the upstream route. The no-state baseline is therefore

$$\Pr(A \cap B \mid q_A, q_B, \mathcal{S}_N) = \Pr(A \mid q_A, \mathcal{S}_N)\Pr(B \mid q_B, \mathcal{S}_N) + \epsilon_{\mathrm{proto}}. \tag{46}$$

Here $\epsilon_{\mathrm{proto}}$ captures residual prompt-format correlations, shared task difficulty, or cluster-level biases that are not real execution effects. The no-state isolation setup estimates this term and finds it small in the measured regime, so multiplicativity is an empirical baseline rather than a theorem from the mere absence of an artifact.

L10: execution rescue from concrete state. L10 says that execution helps only after it produces useful concrete state. Let o_A be the artifact from the upstream step. If o_A tells the router something about the correct downstream skill $s_B^\star$, then the downstream prompt becomes less ambiguous:

$$C(q_B, o_A) = C(q_B) + I(s_B^\star; o_A \mid q_B), \tag{47}$$

where $C(\cdot)$ is concreteness and the mutual-information term is the new evidence supplied by the artifact. Closing this evidence model with the state-gated headroom model in Proposition A gives $\Delta Acc(B \mid A) = 2\alpha Acc(A)(1 - Acc(B))$. $Acc(A)$ appears because rescue requires correct upstream state; $(1 - Acc(B))$ appears because easy downstream routes have little headroom.

L11: dependency-weighted propagation. L11 describes what happens when the upstream state is wrong. The key quantity is dependency strength: if downstream step truly needs the upstream artifact, wrong state should hurt; if dependency is loose, LLM may ignore the bad artifact and use the original task signal. Let $\kappa \in [0, 1]$ measure how much downstream evidence must come from the upstream artifact. Then the quality change decomposes into propagated loss and task-context recovery:

$$\Delta Q_B(\kappa) = -\lambda\,\kappa + r(1 - \kappa). \tag{48}$$

Estimating λ and r from tight and loose dependency groups yields the reported propagation relation. The no-dependency baseline constrains r: recovery requires the downstream task to remain understandable without the upstream artifact, not arbitrary unrelated state. Wrong state hurts when downstream step needs it, but can be ignored when downstream task remains identifiable without it.

Proposition 5 (Dependency-weighted wrong-state propagation). Suppose a wrong upstream artifact is used by the downstream step with dependency weight $\kappa \in [0, 1]$. Let $\lambda > 0$ be the loss when required wrong state is used, and let $r \geq 0$ be the recovery gain from ignoring irrelevant wrong state and relying on the downstream task context. Then

$$\Delta Q_B(\kappa) = -\lambda\kappa + r(1 - \kappa). \tag{49}$$

The effect is negative for $\kappa > r/(\lambda + r)$ and positive for $\kappa < r/(\lambda + r)$.

Proof. Decompose the downstream decision into two mutually exclusive channels: with weight κ, the checker-relevant evidence must come from the upstream artifact; with weight $1 - \kappa$, the downstream task remains identifiable without that artifact. Wrong required state contributes expected change $-\lambda$ on the first channel. Ignoring irrelevant wrong state contributes expected recovery r on the second channel. Linearity of expectation gives the stated affine form. Solving $-\lambda\kappa + r(1 - \kappa) = 0$ gives the sign threshold $\kappa = r/(\lambda + r)$. □

L12: strong-tie promotion and weak-tie drag. L12 explains why putting two steps together can either help or hurt depending on the capability gap $G = |Acc(A) - Acc(B)|$. When $G < G^*$, the pair is effectively weak-tie: neither step gives the other a strong enough anchor, so joint execution carries a small drag that shrinks as the gap grows. When $G \geq G^*$, the pair becomes strong-tie: the stronger step can promote the weaker one, and the gain saturates once the useful scaffold has been provided. The empirical fit is

$$S(G) \approx \begin{cases} -0.0775 + 0.31\,G & G < G^*, \\ +0.265\,(1 - e^{-(G-G^*)/0.12}) & G \geq G^*, \end{cases} \qquad G^* \approx 0.25. \tag{50}$$

Thus Law 12 is not a symmetric ordering effect. The negative weak-tie regime is shallow, while the positive strong-tie regime rises quickly and saturates, matching the main-text observation that promotion by a sufficiently stronger step dominates weak-tie drag.

Proposition 6 (Strong-tie promotion threshold). Let joint execution have crowding cost $c(G)$ that decreases approximately linearly with capability gap in the weak-tie regime, and scaffold benefit $h(G)$ that is zero below a threshold G^* and saturates above it:

$$c(G) = (c_0 - c_1 G)_+, \qquad h(G) = h_{\max}(1 - e^{-(G-G^*)/\tau})\mathbf{1}\{G \geq G^*\}. \tag{51}$$

Then the net synergy $S(G) = h(G) - c(G)$ is negative-but-increasing below G^* and becomes positive-saturating once the scaffold benefit exceeds the residual cost, matching the empirical form in Law 12 in main text.

Proof. For $G < G^*$, $h(G) = 0$, so $S(G) = -c_0 + c_1 G$ as long as the positive part is active, a shallow negative line whenever $0 < c_1 G < c_0$. For $G \geq G^*$, the scaffold term becomes $h_{\max}(1 - e^{-(G-G^*)/\tau})$, which is increasing and concave with asymptote $h_{\max}$. Once $c(G)$ has been driven to zero, the net effect is positive and saturating; before that point, subtracting the residual cost only delays the crossing. This yields the observed asymmetric piecewise structure: weak-tie drag is limited, while strong-tie promotion rises quickly and then saturates. □

Theorem 2 (Intervention closure). Under Lemma A and Theorem A, any intervention that length-matches prompts, preserves the task distribution, and increases semantic margins is predicted by the local model to weakly improve both single-step routing and expected downstream rescue, provided it does not reduce a, α, or $Acc(B)$ and higher-order terms do not dominate:

$$\Delta Acc(N) \geq 0, \qquad \Delta Acc_{\text{res}}(B) \geq 0, \tag{52}$$

with strict improvement whenever the intervention strictly reduces b and $\alpha(1 - Acc(B)) \ln N > 0$.

Proof. By Lemma A, a length-matched boundary intervention that increases semantic margins can only move the semantic component of the slope, and Proposition A gives $\Delta b \leq 0$. Since $Acc(N) = a - b \ln N + o(\ln N)$, decreasing b weakly increases single-step routing accuracy for $N > 1$. By Theorem A,

$$\frac{\partial Acc_{\text{res}}(B)}{\partial b} = -\alpha(1 - Acc(B)) \ln N. \tag{53}$$

When $\alpha(1 - Acc(B)) \ln N > 0$, a strict decrease in b strictly increases the local model's expected rescue term. Therefore, under the fixed-headroom and negligible higher-order conditions stated above, the same library-side intervention that improves routing is predicted to improve the expected availability of correct state for downstream execution. □

B Experimental Details

B.1 Library construction and sampling.

Each routing trial exposes exactly one gold skill and $N-1$ distractors. Distractors are sampled without replacement from the same 14-domain library mixture, with the target domain kept in proportion to the full library unless a local-competition experiment intentionally changes the neighbor set. For multi-step route-only pipelines, all gold skills are fixed by the annotated task graph; only the non-gold exposed skills are resampled. We use fixed sampler seeds for paired comparisons so that baseline and intervention conditions see the same task and distractor instances.

B.2 Task construction and annotation.

Most evaluation tasks are benchmark-derived or adapted from real software-agent task settings. They keep the original task intent and receive additional gold-skill annotations for our library. We use synthetic drafting only for controlled routing cases where existing benchmarks do not require enough distinct skills to support the planned library-size and multi-skill sweeps. These synthetic cases are written from predefined skill combinations, then independently checked against the skill library. A task is retained only when annotators agree on the gold skill or gold skill sequence; disagreements are adjudicated by removing the case rather than relabeling it into the majority class. Synthetic drafting assistance by LLMs (GPT-5.4) is used only before annotation, not for scoring, rewriting, manager optimization, or downstream transfer evaluation.

Human annotation guideline for skills and tasks. Annotators were given the following written guideline.

High-Quality Task Annotation Guideline

Decide whether the task is a valid evaluation item. A valid task must be realistic, executable, and specific enough that another annotator can identify the operation, object, expected artifact, and success condition without seeing model outputs.

Inputs. You will receive a task card with:

- user-visible task instruction;
- benchmark source or synthetic-control flag, when available;
- required output artifact, file, answer, patch, report, or state change;
- pipeline position or candidate composition, when this is a multi-step case;
- any provided files, schema, command context, or environment notes.

You may also inspect the frozen skill-library snapshot to judge whether the task is in scope, but do not choose labels during this pass.

Acceptance criteria. Keep a task only if all criteria below are satisfied.

1. **Clear intent**: the instruction states a concrete user goal, not merely a topic.
2. **Concrete object**: the task identifies or clearly implies the file, repository, API, dataset, table, document, command, state, or other object to operate on.
3. **Explicit operation**: the requested action is identifiable, e.g., analyze, edit, extract, validate, transform, run, compare, summarize, repair, or generate.
4. **Observable output**: the final answer, file, patch, report, state change, or execution artifact can be checked after completion.
5. **Sufficient context**: a competent software agent would not need to invent missing requirements, hidden files, private credentials, unstated schemas, or unprovided external facts.

6. **Library relevance**: at least one available skill is plausibly needed, even if the exact gold skill set is decided later.
7. **No label leakage**: the task should not trivially reveal the answer by copying a skill name or internal label, unless that wording is naturally present in the source benchmark or real user request.
8. **Single coherent request**: the task should not bundle unrelated goals that would naturally be separate tasks.

Permitted light edits. You may make only minimal edits that preserve task intent:

- normalize spelling or formatting;
- expand an abbreviation already clear from context;
- add an explicit expected artifact when it is directly implied by the source;
- split a bundled task only if the source explicitly contains separable subtasks and each subtask can stand alone.

Rejection labels. Use `AMBIGUOUS` when multiple incompatible interpretations are equally reasonable. Use `UNDERSPECIFIED` when key inputs, objects, constraints, or success conditions are missing. Use `OUT_OF_SCOPE` when the task is not an executable software-agent task or no skill-library capability is plausibly relevant.

Reject the task if:

- the requested operation is unclear;
- the expected output cannot be checked;
- key inputs are missing;
- the task requires private credentials, hidden files, or unstated external state;
- it can only be solved by assuming capabilities not present in the task card or library;
- it is primarily open-ended conversation, preference elicitation, or general advice rather than an executable task.

Quality-control rule. Do not inspect model outputs. If annotators disagree on whether a task is `ACCEPTED`, `AMBIGUOUS`, `UNDERSPECIFIED`, or `OUT_OF_SCOPE` after applying the criteria above, exclude the task rather than forcing agreement by majority vote.

Skill and Pipeline Label Annotation Guideline

Label the minimal sufficient gold skill set or ordered gold-skill sequence using the frozen skill library. The label should contain the skills required to complete the task, not every skill that might be useful.

Inputs. You will receive:

- the accepted task card and expected artifact;
- the frozen skill library, including each skill's name, description, declared inputs, declared outputs, domain tag, examples, and boundary notes when available;
- pipeline metadata when the task is part of a multi-step case.

Core principle: minimal sufficient gold set. A gold skill set is valid only if:

1. **Sufficiency**: together, the selected skills can complete the task under their stated interfaces.
2. **Necessity**: removing any selected skill would leave a required operation, artifact, or dependency uncovered.
3. **Interface faithfulness**: the label must rely on declared skill behavior, not on unstated behavior inferred from a broad name.

Single-step tasks. Assign a gold skill set G.

- Use one skill if a single skill fully covers the task.
- Use multiple skills only when the task genuinely requires multiple capabilities, artifacts, or interfaces and no single available skill covers them.
- Do not add formatting, checking, retrieval, or helper skills unless they are required by the task's success condition.
- Mark `AMBIGUOUS` if two or more distinct gold sets are equally valid.
- Mark `UNDERSPECIFIED` if the task lacks information needed to decide the gold set.

Choosing between similar skills. When multiple skills look plausible:

1. Prefer the narrower skill whose description names the concrete object, operation, domain, input, or output in the task.

2. Prefer a skill with matching declared inputs and outputs over a skill that only matches by general wording.
3. Reject broad catch-all skills unless no narrower exposed skill can perform the task.
4. Do not select a skill when the task asks for an excluded behavior, a different artifact type, or a different execution environment.
5. If the difference remains unresolved after checking descriptions, examples, and boundary notes, mark `AMBIGUOUS`.

Pipeline tasks. Assign an ordered sequence of steps:

$$\text{Step 1: } G_1, \quad \text{Step 2: } G_2, \quad \ldots, \quad \text{Step K: } G_K.$$

Each step-level gold set must either:

- produce the final requested artifact;
- produce an intermediate artifact required by a later step; or
- perform a required validation, transformation, or state update named by the task.

Do not add optional helper steps. Do not reorder steps unless the task logic requires that order. If two skills must be used jointly at the same stage, place them in the same step-level gold set rather than creating an artificial sequence.

Dependency labels for adjacent pipeline steps. For each adjacent pair (G_i, G_{i+1}), assign one label:

`tight` the downstream step cannot be completed without the upstream artifact or state.

`loose` the upstream artifact may help, but the downstream step can still be completed from the original task or other provided context.

`none` the steps are independent controls; the upstream artifact is not needed by the downstream step.

Closure check. For every downstream step, list the required input artifact or state and verify that it is produced by an earlier step or present in the original task card. If any required input has no source, mark the case `UNDERSPECIFIED`.

Submission fields. For each accepted case, submit:

- task id;
- `ACCEPTED` task-quality status;
- gold skill set for single-step tasks, or ordered step-level gold sets for pipeline tasks;
- dependency labels for adjacent pipeline steps;
- one-sentence rationale citing the decisive interface or boundary evidence;
- `AMBIGUOUS` or `UNDERSPECIFIED` flag if no reliable label can be assigned.

Agreement policy. Annotate independently and do not inspect model outputs. A case is kept only when annotators agree on the gold skill set or full ordered sequence and dependency labels. If disagreement remains after reviewing interface match, boundary match, negative evidence, minimality, and pipeline closure, exclude the case rather than resolving by majority vote.

B.3 Routing protocol and parsing.

Most routing experiments use the following plain-text prompt:

```
You are a skill router. Given a task description and a list of
available skills, select the single most appropriate skill.

Task: \{task_description\}

Available skills:
\{skill_list\}

Respond with only the skill name. Do not explain your choice.
```

We parse the first exact skill-name span after normalizing whitespace and case. If no exposed skill name is returned, the response is counted as hallucination or abstention, not mapped to the nearest valid skill. This is why the paper can distinguish ordinary out-of-library hallucination from in-library hijack. Unless a provider disables temperature control, routing uses deterministic decoding; repeated

Table 2: Terminology for query-side and skill-side manipulations.

Side	Specific case	Degraded case
Query	concrete anchor (name, file, tool verb, schema field)	vague query
Skill	specific skill (narrow operation, explicit boundaries)	abstract skill (broad purpose, no boundaries)

runs come from independent task/distractor samples or provider-side nondeterminism, not from best-of-k selection.

B.4 Description-quality and anchor manipulations.

Description-quality sweeps hold the task, gold skill, candidate set, and model fixed while changing only the skill-description field. L1 uses names only; higher levels add purpose, inputs/outputs, boundary conditions, and examples. Anchor-loss experiments hold the task intent fixed while removing concrete query anchors such as file names, APIs, schema fields, or domain-specific verbs. Black-hole experiments cross this query manipulation with skill-side abstraction, so capture is measured as an interaction rather than inferred from one degraded condition.

B.5 Execution-state protocol.

No-state execution probes present the two steps jointly but withhold any upstream artifact, thereby estimating interaction before state realization. Correct-state probes execute the upstream skill first and insert the resulting artifact into the downstream routing context. Wrong-state probes condition on incorrect upstream artifacts and stratify pairs by dependency strength κ, where tight dependencies require the upstream artifact for the downstream checker and loose dependencies do not. Artifact correctness is scored by task-specific checkers or human verification when a checker is unavailable.

B.6 Fitting and uncertainty.

Logarithmic routing laws are fit by least squares over $N \in \{10, ..., 500\}$. Binomial rates use Wilson 95% intervals. Cross-model associations use Spearman correlation with percentile bootstrap confidence intervals. Optimization comparisons use paired task sets whenever possible; the headline prospective optimization reports the pre-specified held-out set rather than selecting the best run.

B.7 Terminology for the two manipulations.

The paper manipulates two sides of the routing problem, and the terms are kept separate here so that a result can always be attributed to the side it comes from.

The query-side manipulation removes concrete anchors from the task while preserving intent. The skill-side manipulation removes boundary clauses from candidate descriptions while preserving names and interfaces; "abstract-skill removal" and "skill-anchor ablation" refer to this single intervention, and neither is equivalent to weakening the query.

B.8 Routing Cascade Checks

This diagnostic uses route-only multi-step tasks to measure whether pipeline failure is only the product of single-step routing errors or whether later routes become harder after earlier routing decisions have compressed the task state.

Reading the per-step U-shape. The U-shape in Fig. 2 (d) describes per-step routing accuracy and does not mean error accumulation reverses. Three quantities must be kept apart: per-step accuracy, which may rise again near the final step as the goal narrows the candidate set; the all-correct chain probability, which can only stay flat or fall as steps are appended; and recovery-enabled success, which rises only when a retry, execution artifact, or state correction is available. A late-step rebound in the first quantity therefore does not undo an error made earlier in the chain.

Route-only cascade protocol. Each case is an annotated skill sequence $(s_1^\star, \ldots, s_K^\star)$ with the same underlying task graph used across library sizes. At step k, the router sees the current subtask description, the exposed library of size N, and the previous routed skill names, but it does not receive execution artifacts or checker feedback. This isolates planning state from execution state: any

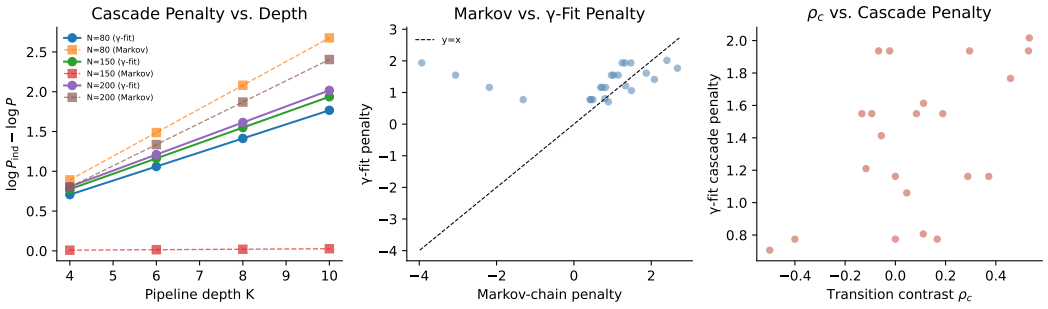


Figure 9: **Cascade Penalty Decomposition.** Pipeline depth amplifies the gap between observed end-to-end accuracy and the independence baseline.

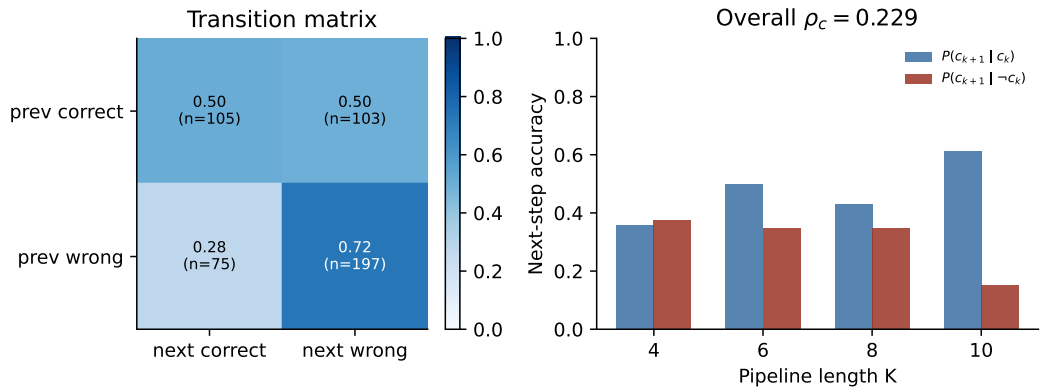


Figure 10: **State-Transition Audit.** Downstream routing accuracy depends on whether the preceding step remains correct, linking transition state to cascade behavior (annotated by human annotators).

downstream degradation comes from routing over a compressed intermediate plan, not from a wrong file, table, API result, or other realized artifact. For every N, the K gold skills are fixed and only the non-gold distractors are resampled under the same domain as the single-step routing sweep.

Matched product baseline. For each pipeline position k, we run the same subtask as an isolated single-step routing query with the same gold skill and matched distractor sampling. Let $Acc_k(N) = \Pr[\hat{s}_k = s_k^\star]$ be this isolated accuracy. The no-cascade product baseline is

$$Acc_{\text{ind}}(N, K) = \prod_{k=1}^{K} Acc_k(N). \tag{54}$$

We then run the full route-only pipeline on the same task set and record strict success $Acc(N, K) = \Pr[\hat{s}_1 = s_1^\star, \ldots, \hat{s}_K = s_K^\star]$. The extra depth loss is

$$D(N, K) = \log Acc_{\text{ind}}(N, K) - \log Acc(N, K). \tag{55}$$

Using log probabilities makes the excess penalty additive across depth and comparable between models with different absolute single-step accuracies. Fig. 9 plots $D(N, K)$ against depth for each library size (left) and compares the Markov prediction against the measured penalty (right).

Transition audit. To check whether the excess loss is consistent with local cascade behavior, we estimate adjacent correctness transitions from the same route-only runs. For every adjacent pair $(k, k+1)$, define $c_k = \mathbf{1}[\hat{s}_k = s_k^\star]$ and estimate

$$\rho_c = Acc(c_{k+1} \mid c_k) - Acc(c_{k+1} \mid \neg c_k) \tag{56}$$

after pooling adjacent pairs within the same model, library size, and depth condition. A positive ρ_c means the next route is more likely to be correct when the previous route was correct. We compare this transition estimate with the fitted depth exponent γ from the scaling law as a diagnostic, not as an additional fitted result: if both move together, the observed pipeline penalty is attributable to local routing-state compression rather than an unmatched prompt or task-difficulty artifact (Fig. 10).

Table 3: Fitted parameters for $Acc(N) = a - b \ln N$ across all 15 models. The intercept a can slightly exceed 1 for the strongest models; the law is used over $N \geq 10$.

Model	a	b	R^2	N^* (80%)
GPT-5.4	1.021	0.031	0.981	1246
GPT-5.4-mini	0.998	0.038	0.974	183
Claude Opus 4.6	0.987	0.041	0.979	96
Claude Sonnet 4.6	0.972	0.044	0.976	50
DeepSeek-V4 Pro	0.965	0.046	0.978	36
Gemini 3.1 Pro	0.961	0.047	0.971	31
Qwen3-235B	0.957	0.049	0.973	25
GPT-5-mini	0.953	0.049	0.968	23
Kimi K2.5	0.941	0.052	0.973	15
Kimi K2.6	0.939	0.053	0.971	14
Doubao Seed 2.0 Pro	0.934	0.055	0.969	11
Gemini 3.1 Flash Lite	0.921	0.058	0.972	8
GLM-4.7	0.914	0.060	0.974	7
GLM-5	0.908	0.062	0.975	6
GPT-4o-mini	0.897	0.065	0.971	4

C Routing Law

C.1 Empirical Decay Parameters

Table 3 reports the fitted intercept a, decay slope b, and 80% threshold N^* for each evaluated model. Stronger models exhibit smaller b, pushing the operational threshold to larger libraries, but none remove the logarithmic form itself.

Routing-scale experiment details. Each routing condition fixes the evaluated task set and changes only the number of exposed candidate skills. A trial contains the k gold skill and $N-k$ randomly selected distractors sampled without replacement from the domain-stratified library pool. The target distribution is therefore held fixed as N changes: increasing N adds distractors around the same gold tasks rather than introducing harder targets. For paired or repeated settings, the task identifier and candidate-set seed define the unit of aggregation. Repeated calls sharing that unit are averaged first, and task-level means are then averaged across the condition, so tasks with more repeats do not receive extra weight.

Single-step routing accuracy is estimated for each model and each $N \in \{10, 20, 50, 100, 200, 500\}$ and fit by ordinary least squares as $a - b \ln N$. Pipeline experiments use route-only annotated skill sequences: the gold sequence is known, every step is routed, but no execution artifact from an earlier step is inserted into the next step. This isolates planning and routing from state realization. The independent baseline for a length-K pipeline is computed from the corresponding single-step fit as $(a - b \ln N)^K$. The observed route-only pipeline accuracy is then compared with this baseline to estimate a model-specific compounding exponent γ. Per-step analyzes keep the pipeline length fixed while computing accuracy at each step, then use the normalized position k/K only when pooling curves across different lengths.

The fitted intercept a is not used as a probability estimate. It is the intercept at $\ln N = 0$, outside the evaluated range, which is why it can slightly exceed one for the strongest models in Table 3. All claims use the measured sweep $N \in \{10, 20, 50, 100, 200, 500\}$ and the local slope b over that range; the linear-log law is not extrapolated beyond the observed support.

C.2 Decay Slope vs. Semantic Gap Mechanism

This diagnostic connects the fitted routing law to the paper's main mechanism. If the logarithmic slope b were only a generic model-capacity or prompt-length effect, then it would not depend strongly on where a skill sits inside the library. The local-competition account makes a sharper prediction: skills whose descriptions sit close to a neighbor should lose accuracy faster as more candidates are exposed, because additional skills add plausible near-misses around the same intent. We therefore test whether per-skill decay is largest exactly where the library geometry is most crowded. The visual evidence is Fig. 3: panel (a) shows that sharper descriptions flatten the decay slope, panels (b,c)

show where local near-miss errors concentrate, and panel (d) summarizes the same effect through the competition-index fit. This appendix paragraph gives the per-skill regression behind that visual mechanism.

How the diagnostic is run. For each skill, we embed the same description text that appears in the routing prompt and compute the cosine distance to its nearest neighboring skill, denoted $\Delta\mu$. This step uses only the library text and is completed before looking at model outputs. We then estimate a per-skill decay slope using only trials where that skill is the gold target across the same library-size sweep as the main routing experiment. Appearances of the skill as a distractor for other targets are used only to define local competition, not to estimate that skill's own slope.

We correlate the per-skill slope with $1/\Delta\mu$, because a smaller nearest-neighbor gap corresponds to a larger inverse-gap value and hence a predicted larger decay slope. This keeps the diagnostic aligned with the design claim: boundary rewrites should help most when they increase separation between locally confusable skills. In the figure, this is the move from the crowded danger-band regime toward wider local margins; in the regression, it appears as a positive association between inverse gap and decay slope. The relationship is strong: Spearman ρ=0.71 (p<0.001, n=452 skills). A linear regression $b = \beta_0 + \beta_1/\Delta\mu$ gives β_0=0.008 (95% CI: $[0.003, 0.013]$), β_1=0.031 ($[0.026, 0.036]$), and R^2=0.49. The positive intercept means that even isolated skills retain a small baseline decay from finite context and selection noise, but nearly half of the per-skill variation in slope is explained by local semantic crowding. This is why the later optimization targets skill boundaries and nearest-neighbor audits rather than only changing the global prompt. The 2×2 component attribution for the two highest-leverage library-side actions is reported once, in Table 17.

C.3 Routing Law Controls: Baselines and Task Difficulty

The routing law should not depend on a single implementation of the router, nor should it be an artifact of harder tasks appearing at larger N. This section therefore collects two controls: retrieval baselines under the same scaling sweep, and task-difficulty checks that hold task hardness separate from library size.

C.3.1 Retriever and Router Baselines

Table 4 positions zero-shot LLM routing against sparse and dense retrieval baselines under the same scaling sweep. The purpose is to show that decline with library scale is not unique to one routing implementation.

Note on sample sizes. The baseline table uses n=50 tasks per condition, which is intentionally smaller than the n=500 used in the main scaling experiments. The baselines are not the primary contribution of the paper; the goal is to check whether sparse retrieval, dense retrieval, and LLM routing all show degradation with larger candidate sets. We report means across runs where repeated runs are available rather than best-of-run values. The low R^2 values for some rows reflect sampling variance at this scale, not absence of the decay trend: at n=500, the GPT-5 series achieves R^2>0.96 in the main scaling experiment.

BM25 is a lexical control, not the main skill-router regime. The main analysis concerns LLM semantic routing: the model reads natural-language skill descriptions, infers task intent, and can therefore be pulled toward broad semantic attractors. BM25 is included only as a sparse lexical retrieval control. It matches surface terms between the query and skill descriptions, but it does not perform the same semantic abstraction, state-aware skill selection, or execution-conditioned reasoning [80, 81] studied in the paper. This difference is why the large-N reversal is informative. At N=200, BM25 (78.0%) outperforms LLM routing (GPT-5.4-mini, 60.7%): BM25 is less exposed to black-hole capture precisely because it lacks the semantic generalization that makes an abstract catch-all skill attractive to an LLM router. The result is therefore not an endorsement of BM25 as a deployment choice; LLM routing is substantially better at small N, handles paraphrase better, and is the relevant regime for skill-based agents. Rather, the reversal is a negative-control signal: the large-library degradation we analyze is not just generic task difficulty or prompt length, but a semantic in-library hijack mechanism specific to LLM-style skill routing.

C.3.2 Task-Difficulty Confounds

The difficulty concern is that larger libraries might accidentally contain harder tasks, so the observed decay could reflect task selection rather than library size. Our routing sweep is constructed to avoid this: the gold task pool is held fixed. Concretely, every size-N condition draws from the same

Table 4: Routing accuracy across methods and library sizes (n=50 tasks per condition; means across runs where available). All methods show the same qualitative decline as N grows, but this table is an auxiliary implementation check rather than a headline quantitative fit.

Method	N=20	N=50	N=100	N=200	b	R^2
BM25	85.3%	90.0%	79.3%	78.0%	0.041	0.52
Dense Retrieval (BGE)	90.0%	90.7%	88.0%	80.0%	0.041	0.68
LLM (GPT-5.4-mini)	93.3%	92.7%	80.7%	60.7%	0.140	0.82

domain-stratified task pool (n=500 per condition in the main sweep); each trial exposes the gold skill plus $N-1$ distractors sampled from the same library mixture. Thus larger N changes the number of alternatives around the same targets, not the target distribution itself.

The control is visible in the fitted scaling curves. Under this fixed-target, distractor-only sweep, all 15 models still follow $Acc(N) = a - b \ln N$ over $N \in \{10, 20, 50, 100, 200, 500\}$, with R^2=0.968–0.981 and positive slopes b=0.031–0.065 (Table 3). The strongest model has the shallowest decay (b=0.031), while GPT-4o-mini has the steepest (b=0.065); the sign and log-linear form are unchanged across the model range. Because the task pool does not get harder as N grows, these slopes measure sensitivity to additional distractors rather than a between-condition shift in intrinsic task hardness.

We use the same logic for route-only pipeline controls: the annotated gold skill sequence is fixed by the task graph, and only non-gold exposed skills are resampled. The observed pipeline degradation therefore cannot be explained by longer or harder target tasks being introduced at larger N; it occurs when the same required skills are routed among more candidate distractors. Finally, per-domain stratification prevents a domain-composition artifact: each size condition preserves the same domain mixture, so the decay is not driven by difficulty becoming overrepresented at large library sizes.

Token-budget boundary. The library-size sweep increases both distractor count and prompt length. Our controls do not make a pure token-budget estimate. They show that the decay is not only generic long-context degradation: the task pool is fixed, length-matched boundary rewrites still move accuracy, and the semantic competition index predicts failures better than N alone. The measured slope therefore applies to the flat exposed-library prompt format studied here, not to pagination, hierarchy, or retrieval-gated top-M exposure.

We also ran a small GPT-5.4-mini context-growth check that keeps the required skill and task pool fixed while increasing only the exposed skill context. The added context contains non-gold skills drawn by the same domain-stratified sampler, so the required skill remains available in every condition and the target tasks do not change. The check is intended as a token-growth stress test, not a replacement for the full 15-model scaling sweep: it asks whether the GPT-5.4-mini decline appears when task hardness is fixed and only the exposed candidate context grows. The observed pattern matches the main sweep direction, supporting the interpretation that scale-induced context growth is part of the flat-library routing problem rather than a separate task-difficulty artifact.

We additionally use a context-matched skill-count control to separate exposed skill count from approximate prompt length. The control uses the same tasks and GPT-5.4-mini router in all conditions. For each task, we first sample the N=150 candidate set and use its description-only candidate list to define a fixed token budget. We then vary the exposed skill count from N=10 to 150, truncating GPT-5.4-expanded descriptions evenly across skills to match that same candidate-token budget (Table 5). Thus the number of selectable in-library alternatives changes while the exposed candidate-list token count remains approximately fixed.

Table 6 makes the context-length boundary explicit. The routing law is estimated on the 1,141-skill diagnostic library after near-duplicate removal, where increasing N also increases the candidate text shown to the model. The downstream manager starts from the broader 1,412-skill raw transfer library and does not claim to hold prompt length fixed; instead, it changes the exposed surface by selecting a smaller domain- and task-conditioned context from that raw library. Therefore the downstream gains should be read as evidence that law-guided gating and editing help under a matched transfer protocol, not as an isolated estimate of the causal effect of tokens.

Table 5: Context-matched skill-count control. The same task set and GPT-5.4-mini router are evaluated under approximately matched candidate-list token budgets while changing the number of exposed skills. Candidate budgets are set by the description-only $N{=}150$ condition, while each evaluated condition renders truncated GPT-5.4-expanded descriptions to match that budget.

Skills	Candidate tokens	Accuracy	Trials
10	5176	60.5%	580
25	5176	56.2%	580
50	5176	51.0%	580
75	5176	54.1%	580
100	5176	53.1%	580
125	5176	53.1%	580
150	5176	49.5%	580

Table 6: Context-scale accounting for the downstream library construction. This is not a pure token-only ablation; it records the scale difference between full exposed context and the gated manager context used in transfer checks.

Context construction	Raw skills	Global cap	Per-task cap	Per-domain cap
Raw collected library	1412	–	–	–
Deduplicated diagnostic library	1141	–	–	–
Manager-gated downstream library	1412	250	80	24

What each control rules out. The controls separate several possible explanations but do not collapse them into a single causal estimate. The fixed-target sweep rules out task-set hardening across N. Domain stratification rules out a domain-composition shift. Length-matched boundary rewrites rule out a purely token-count account for the boundary intervention, because the candidate list length is matched while local accuracy changes. The per-skill inverse-gap regression and CI fit rule out a pure candidate-count account, because skills with more crowded neighborhoods lose accuracy faster than isolated skills under the same global library size. Finally, the sparse-retrieval control is a negative control for semantic abstraction: BM25 degrades with candidate growth but is less vulnerable to abstract black-hole capture than LLM routing. Together these checks support semantic competition as the dominant measured mechanism in the flat exposed-library protocol, while leaving open how much a hierarchical or retrieval-gated interface would reduce the slope.

C.4 Geometry-Conditioned Routing Law

Definition (Query-conditioned competition mass). For a query q, target skill s^*, exposed library $\mathcal{S}_N$, and routing utility $u(q,s)$, define

$$Z_q(N,\mathcal{G}) = \sum_{s\in\mathcal{S}_N\setminus\{s^*\}} \exp(\beta[u(q,s)-u(q,s^*)])\,, \tag{57}$$

where $\mathcal{G}$ denotes the query-conditioned library geometry. Under the corresponding finite-capacity choice approximation,

$$P(s^* \mid q,\mathcal{S}_N) = \frac{1}{1+Z_q(N,\mathcal{G})}. \tag{58}$$

Proposition 7 (Geometry-conditioned routing scaling). Suppose that, over a geometry regime $\mathcal{G}$ and the measured library range, the effective competition mass satisfies

$$Z_q(N,\mathcal{G}) = c_q(\mathcal{G})N^{\kappa_q(\mathcal{G})}(1+o(1)), \qquad c_q(\mathcal{G})>0, \quad \kappa_q(\mathcal{G})\geq 0, \tag{59}$$

so that

$$\operatorname{logit} P(s^* \mid q,\mathcal{S}_N) = -\log c_q(\mathcal{G}) - \kappa_q(\mathcal{G})\log N + o(1). \tag{60}$$

Then, around an interior operating point N_0, routing accuracy has the finite-range expansion

$$Acc(N,\mathcal{G}) = a_{\mathcal{G}} - b_{\mathcal{G}}\log N + O\big((\log N-\log N_0)^2\big)\,, \tag{61}$$

Table 7: Geometry-conditioned reanalysis of 1,344 routing decisions.

Geometry statistic	Spearman ρ with accuracy	p-value
Maximum similarity	-0.594	1.32×10^{-4}
Top-5 mean competition	-0.589	1.58×10^{-4}
Dangerous-competitor count	-0.603	9.89×10^{-5}
Log-sum-exp competition mass	-0.548	5.42×10^{-4}

Table 8: Cluster-robust logistic analysis of the same routing decisions. All specifications include model fixed effects with standard errors clustered by model–task; lower AIC indicates better fit. Note that once geometry is included, $\log N$ loses significance, which is the empirical form of Proposition C.4.

Specification	AIC	Geometry p	$\log N$ p
$\log N$	577.5	—	.0011
$\log N$ + dangerous count	542.4	4.16×10^{-6}	.0105
$\log N$ + Top-5 competition	518.5	3.08×10^{-17}	.201
$\log N$ + dangerous count + Top-5	516.9	1.52×10^{-11}	.199

with

$$b_{\mathcal{G}} = \kappa_{\mathcal{G}}\, p_0(1-p_0), \qquad p_0 = P(s^* \mid q, \mathcal{S}_{N_0}). \tag{62}$$

Geometry therefore changes baseline difficulty through $c_{\mathcal{G}}$ and scale sensitivity through $\kappa_{\mathcal{G}}$, and an accuracy floor can compress the observable accuracy-space slope through the factor $p_0(1-p_0)$.

Proof. Equation (58) gives $\operatorname{logit} P(s^* \mid q, \mathcal{S}_N) = -\log Z_q(N, \mathcal{G})$; substituting Eq. (59) yields Eq. (60). Writing $x = \log N$ and $P(x) = \sigma(A_{\mathcal{G}} - \kappa_{\mathcal{G}} x)$ with $A_{\mathcal{G}} = -\log c_{\mathcal{G}}$, a Taylor expansion around $x_0 = \log N_0$ gives $P(x) = P(x_0) - \kappa_{\mathcal{G}} P(x_0)(1 - P(x_0))(x - x_0) + O((x - x_0)^2)$, and absorbing the constant terms into $a_{\mathcal{G}}$ proves Eqs. (61)–(62). □

Lemma 3 (Exact dangerous-competitor exposure). Let the complete non-target pool contain $L = M - 1$ skills, of which $m_q > 0$ lie in a query-specific dangerous set. If $d = N - 1$ non-target skills are sampled without replacement,

$$H_q(N) = 1 - \frac{\binom{L-m_q}{d}}{\binom{L}{d}} \tag{63}$$

is strictly increasing in N before saturation.

Proof. With $Q_d = 1 - H_q(N)$ the probability of drawing only safe skills, $Q_{d+1}/Q_d = (L - m_q - d)/(L - d) < 1$ whenever $m_q > 0$, so Q_d strictly decreases and $H_q(N)$ strictly increases with exposure size. □

Corollary (Law-derived manager controls). The geometry-conditioned law yields three control rules used by the manager. Compressing exposure from N to $N' < N$ predicts $Acc(N') - Acc(N) \approx b \log(N/N')$. For any competitor s_j the choice model gives $\partial P(s^*)/\partial u(q, s_j) = -\beta P(s^*) P(s_j) < 0$, so the highest-probability local competitors carry the largest marginal harm and should be audited first. Finally, under $\Delta P(B \mid A) = 2\alpha(1 - P(B))P(A)$ we have $\partial\Delta/\partial P(A) = 2\alpha(1 - P(B)) > 0$ and $\partial\Delta/\partial P(B) = -2\alpha P(A) < 0$, which prioritises reliable upstream state and difficult downstream decisions with high headroom. These derivatives motivate exposure compression, targeted boundary repair, and verified-state writeback respectively.

C.5 Mechanism Probe: Manipulating Skill Boundaries

The routing law says that larger libraries hurt because they create denser neighborhoods of plausible alternatives. To probe this mechanism directly, we run a *description intervention*: we manipulate the semantic similarity between pairs of skills by rewriting their descriptions to be more or less similar, then test whether routing accuracy changes as predicted by the capacity model.

Table 9: Fixed-N intervention. Library size is held at $N = 120$ while only the number of query-local competitors changes. The Cochran–Armitage trend test gives $z = -4.62$, $p = 3.89 \times 10^{-6}$; the paired $k = 0$ versus $k = 7$ drop is 33.3 points (bootstrap 95% CI 21.7–45.0; exact McNemar $p = 1.91 \times 10^{-6}$). Holding N fixed isolates geometry as the cause: the declared library size is identical in every row.

Local competitors k	Correct / Total	Accuracy
0	60 / 60	100.0%
1	50 / 60	83.3%
3	53 / 60	88.3%
7	40 / 60	66.7%

Local-competition experiment details. The description-quality sweep is paired at the trial level. We hold fixed the task, gold skill, exposed distractor set, model, prompt template, decoding setting, parser, and scoring rule. Only the description field changes. The levels progress from name-only text to brief summaries, target objects and actions, constraints and examples, and finally bounded counterexamples. This design tests whether sharpening the skill boundary changes routing while keeping the underlying task and candidate set unchanged.

Local competition is measured before model inference. For each exposed library, we compute similarity between the gold skill and each exposed non-gold skill, rank the nearest distractors, and count exposure to predefined similarity bands. Error-rate summaries are then grouped by these precomputed ranks and bands. The danger band is selected as the band with the most negative Spearman association between exposure and accuracy: it is the region where adding candidates most reliably hurts. Finally, we compute a competition index

$$CI(s^\star, \mathcal{S}_N) = \sum_{s_j \in \mathcal{S}_N \setminus \{s^\star\}} \exp(\beta \, \mathrm{TFIDFSim}(s^\star, s_j)) \,, \tag{64}$$

over exposed non-gold skills and fit the Boltzmann form

$$Acc(\hat{s} = s^\star) = \frac{A}{A + CI(s^\star, \mathcal{S}_N)}. \tag{65}$$

Here CI is a scalar summary of distractor pressure around the gold skill. The coefficient β is a fitted inverse-temperature parameter: when β is larger, small increases in description similarity create much larger competitive weight. If $\beta = 0$, every exposed distractor contributes equally and CI collapses to a count; positive β makes near neighbors dominate the sum. The parameter $A > 0$ is the fitted gold-side baseline strength, absorbing the target skill's average salience, description quality, and any model-level preference for the correct class that is not explained by exposed distractors. Thus the fitted probability compares one target mass A against the total distractor mass CI. The gold skill is excluded from CI so that the index measures distractor pressure, not target descriptiveness.

Boundary intervention. We select 60 skill pairs from the danger zone ($\cos \in [0.55, 0.75)$) and 60 pairs from the safe zone ($\cos < 0.55$). For each pair, we produce three description variants: *original*, *pushed apart* (targeting $\cos < 0.45$), and *pulled together* (targeting $\cos > 0.80$). We verify cosine shifts using BGE embeddings before running routing experiments (n=100 tasks per pair per variant). All variants are length-matched to within ± 15 tokens and written at L4 quality; functional equivalence is verified by two annotators (κ=0.88). The manipulation therefore changes the local semantic geometry while holding the skill function fixed.

Mechanism check.

- **Pushed apart**: accuracy increases by $+18.4\%$ for danger-zone pairs (p<0.001, n=60) and $+9.2\%$ for safe-zone pairs (p=0.003).
- **Pulled together (danger zone)**: accuracy decreases by -11.3% (p<0.001).
- **Pulled together (safe zone)**: accuracy *increases* by $+3.1\%$ (p=0.041), consistent with the near-duplicate disambiguation effect.

All four directional predictions match the capacity model without refitting. This supports the mechanism behind the routing law: description similarity is not merely correlated with routing error, but can causally move probability mass among local competitors.

Table 10: Flat versus family-first routing. Family-first routing cuts final-stage exposure by roughly 90% but does not improve end-to-end accuracy, because family-selection error and within-family confusion offset the gain.

Protocol	Total size	Family accuracy	Final-stage exposure	Final accuracy
Flat	60	—	60	6.25%
Family-first	60	87.5%	6.67	6.25%
Flat	120	—	120	3.13%
Family-first	120	87.5%	12.27	1.56%

C.6 Hierarchical and Progressively Revealed Libraries

The routing law still applies under hierarchy, but N must be read as the number of skills exposed at the current decision, not the total library size. Flat routing asks the model to choose from all N skills at once. Family-first routing splits this into two steps: choose a family, then choose within that family. For a 120-skill library, family-first routing exposes 12.27 skills on average at the second stage, an 89.8% reduction in single-step exposure.

Hierarchy also adds a failure mode: if the family choice is wrong, the correct skill never appears downstream. We compared flat and family-first routing on all 32 tasks with 64 decisions per condition:

The compression does not translate into higher accuracy. At N=60 both protocols reach 6.25%; at N=120 family-first routing reaches 1.56%, below the 3.13% of flat routing, because the 12.5% family-selection error and within-family confusion cancel the exposure reduction. The more accurate statement is therefore that hierarchy converts one global routing problem into several smaller routing decisions that must each succeed.

For a two-level library, the end-to-end rate is

$$P_{\text{hier}} = P(\text{correct family}) \times P(\text{correct skill} \mid \text{correct family}) \approx G\,(a - b \ln N_{\text{leaf}}), \tag{66}$$

where G is family-selection accuracy and N_{leaf} the skills exposed after the family is chosen, against flat routing $P_{\text{flat}} \approx a - b \ln N_{\text{total}}$. Hierarchy pays off only when $G(a - b \ln N_{\text{leaf}}) > a - b \ln N_{\text{total}}$, which is the *exposure-compression–gate-error tradeoff*: a smaller leaf candidate set helps routing, but the extra family gate adds its own error. Whether hierarchy helps therefore cannot be read off the exposure reduction alone; a report must give family-selection accuracy, per-level exposure, within-family routing accuracy, and end-to-end accuracy together. The same reasoning covers clustering and progressive disclosure: clustering that merely organizes the library while still exposing every skill leaves the flat law unchanged, whereas cluster-then-reveal systems are governed at each level by the law for that level's exposed size, multiplied by the success rates of the levels above it. For L levels,

$$P_{\text{hier}} \approx \prod_{l=1}^{L} (a_l - b_l \ln N_l)\,, \tag{67}$$

so hierarchy does not remove the routing law; it redistributes the effect of total library size across per-level exposed sizes and adds cross-level error accumulation.

C.7 External Validity

To validate the routing law beyond the main skill-routing experiments, we test its qualitative logarithmic form on ToolBench [6], not as a refit of the main paper's quantitative slopes (Fig. 11).

External routing setting. We sample 140 answer traces from the ToolBench G1 split and extract 633 unique API functions with their natural-language descriptions. For each of 30 sampled queries per library size $N \in \{5, 10, 20, 50, 100, 200\}$, we construct a library containing the gold API plus $N-1$ distractors drawn uniformly from the full API pool. Both GPT-4o-mini and Claude Sonnet 4.6 route each query to a single API name using the same zero-shot plain-text protocol as the main experiments. The random seed is fixed at 42.

ToolBench measurement details. For each model and library size, each query is scored as correct if the parsed API name matches the gold API from the original ToolBench trace. A response that does not match any exposed API name is counted as hallucination; an exposed but non-gold API

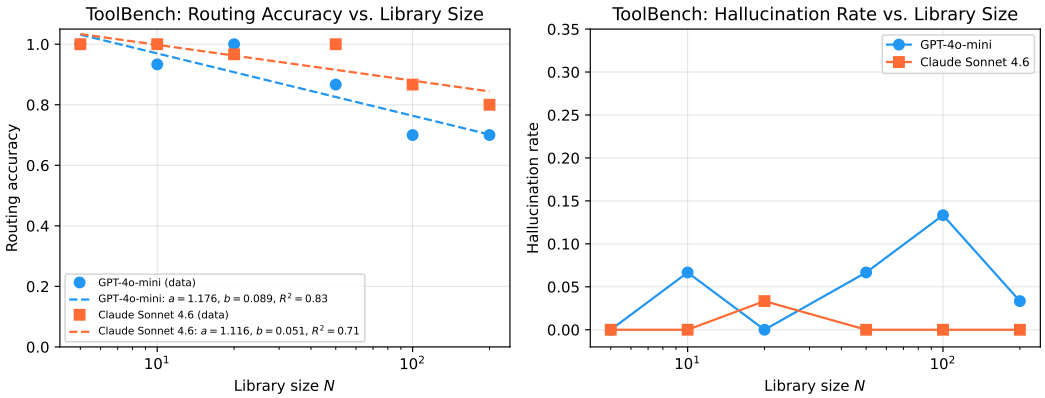


Figure 11: **ToolBench Validity.** ToolBench routing accuracy and hallucination rates follow the same qualitative library-size pattern on an external tool corpus.

Table 11: ToolBench routing accuracy and hallucination rate by library size. 30 queries per condition, gold API + uniform distractors.

	GPT-4o-mini		Claude Sonnet 4.6	
N	Accuracy	Halluc.	Accuracy	Halluc.
5	1.000	0.000	1.000	0.000
10	0.933	0.067	1.000	0.000
20	1.000	0.000	0.967	0.000
50	0.867	0.067	1.000	0.000
100	0.700	0.133	0.867	0.000
200	0.700	0.033	0.800	0.000
Log fit	a=1.176, b=0.089, R^2=0.83		a=1.116, b=0.051, R^2=0.71	

is counted as an in-library routing error. Accuracy and hallucination rate are computed over the 30 sampled queries at each N. The log fit is estimated separately for each model from those condition means, using the same $a - b \ln N$ form as the main routing experiment.

Routing-law replication. Table 11 and Fig. 11 show the accuracy and hallucination rates for both models. GPT-4o-mini accuracy declines from 1.000 at N=5 to 0.700 at N=200; Claude Sonnet 4.6 declines from 1.000 to 0.800. Log fits $Acc(N) = a - b \ln N$ yield: GPT-4o-mini: a=1.176, b=0.089, R^2=0.83; Claude Sonnet 4.6: a=1.116, b=0.051, R^2=0.71. Both slopes are positive, and the qualitative decay pattern replicates across model families. Hallucination rate stays low ($\leq$13%) across all N for both models, replicating the main routing result that in-library routing errors dominate over hallucination at scale.

D Execution Law

D.1 Coverage–Routing Law

Let q be the fraction of arriving tasks whose benchmark-annotated target skill the current library covers, let $R(N)$ be the conditional probability of selecting that skill when it is present, and let $S(N, q)$ be the overall success probability. The routing law gives $R(N) = A - B \ln N$, so

$$S(N, q) = q\, R(N) = q\,(A - B \ln N). \tag{68}$$

Coverage decides whether a task can be routed correctly at all; library size decides whether it then is. Both enter multiplicatively.

We tested this on a skill-only replication: 320 synthetic skills, 32 benchmark tasks, and library sizes $N \in \{20, 40, 60, 90, 120, 180, 240\}$ with GPT-5.4-mini. Fitting $R(N)$ on the q=1 cells alone gives $R(N) = 1.115 - 0.207 \ln N$ (R^2=0.715). Substituting this into Eq. (68) predicts the held-out q=0.25, 0.50, 0.75 cells without refitting:

Table 12: Held-out test of the Coverage–Routing Law on 42 cells not used to fit $R(N)$, and the joint fit over all 70 cells. The calibration slope is essentially 1, so the multiplicative form is not only correlated with the data but on the right scale.

Metric	Held-out ($q < 1$)	Joint (all q)
Pearson r	0.864	0.888
Pearson 95% CI	$[0.760, 0.925]$	$[0.825, 0.929]$
p-value	1.66×10^{-13}	1.40×10^{-24}
Spearman ρ	0.732	0.770
R^2	0.744	0.788
MAE (pp)	3.94	4.12
Calibration slope	1.002	1.005
Calibration intercept (pp)	-0.63	-0.12

A separability check supports writing coverage and routing as two factors: among samples where the target skill is present, conditional routing accuracy does not vary systematically with the global coverage mixture q (Spearman $\rho = 0.065$, $p = 0.636$, 56 cells).

Missing-skill blind spot. When no applicable skill is present, the agents in these probes almost never abstain; they instead force-select a superficially related skill. Removing only the benchmark-annotated target while leaving same-family substitutes yields a substitute selection rate of 93.8–95.3%, but overall task accuracy still falls. Removing the entire capability family drives applicable-skill selection to zero, yet abstention stays at 1.4% over 1,000 routing decisions.

Design implication. Adding a skill has two opposing effects: it can raise coverage q, and it can lower the conditional rate $R(N)$ by crowding the library. Whether a new skill belongs in the library therefore depends on whether its coverage gain exceeds its routing cost, so skills that fill a genuine gap and skills that duplicate existing capability should not be admitted under the same rule.

D.2 Concrete Failure Cases

Case 1: local boundary confusion. Two real skills in the library, *Anomaly Detection* and *Constraint Validation*, have description embeddings with cosine similarity 0.9321. Both mention "validate data" and "quality checks"; the operational difference is that Anomaly Detection finds previously unknown outliers and schema inconsistencies, whereas Constraint Validation checks rules that are already given, such as not-null, uniqueness, and numeric ranges.

For the request "Run data quality checks and validate the dataframe", either skill is plausible. Across 96 such routing decisions, Constraint Validation is chosen 83 times (86.5%), and all 45 errors fall inside this pair with none reaching the other skills. The model is not choosing randomly; the two descriptions form a boundary it cannot resolve, and the wording of Constraint Validation happens to overlap the query more closely.

Lengthening the descriptions does not fix this. Adding explicit operational boundaries does:

```
Anomaly Detection: use to discover previously unknown anomalies or outliers; do
     not use it when the task only asks to enforce predefined constraints.
Constraint Validation: use when validation rules such as not-null, uniqueness,
    or numeric ranges are already specified; do not use it for exploratory
    anomaly detection.
```

Pooled over six models, accuracy on these requests moves from 53.1% to 95.8%. When the request still carries no decisive information, the correct behavior is to ask which of the two operations the user wants rather than to guess between them.

Case 2: black-hole capture. The second case is not local confusion but capture by an over-broad skill. The specialized skill is *Dependency Installation*, which installs the dependencies declared in `package.json`. The request "Please install the dependencies listed in this project's package.json as

Table 13: Cumulative routing success under self-reflection retry. Retry raises cumulative success substantially, but every model is worse at N=128 than at N=64 under every budget.

Model	Skills	RT = 1	RT = 3	RT = 5
GPT-5.4-mini	64	5.0%	48.8%	57.5%
GPT-5.4-mini	128	2.5%	30.6%	33.8%
GPT-5.4	64	12.5%	31.2%	62.5%
GPT-5.4	128	8.0%	19.4%	42.5%
GPT-5.5	64	26.2%	43.8%	68.8%
GPT-5.5	128	10.0%	32.5%	58.1%
DeepSeek-V4-Pro	64	15.6%	59.4%	71.9%
DeepSeek-V4-Pro	128	6.2%	21.9%	46.9%
Gemini-3.1-Pro	64	15.6%	50.0%	81.2%
Gemini-3.1-Pro	128	3.1%	15.6%	46.9%
Qwen3.7-Max	64	28.1%	59.4%	87.5%
Qwen3.7-Max	128	6.2%	31.2%	40.6%

quickly as possible" offers three concrete anchors (install, package.json, dependencies) that point unambiguously to it.

Rewriting the same intent as "ensure that everything needed for the project to function smoothly is taken care of" removes all of them. A synthetic broad-fallback diagnostic skill, which claims to apply whenever a request is ambiguous or specialized skills look too narrow, then absorbs the task. The claim is not that the model recovers the deleted npm information, but that the system fails to ask for clarification and is intercepted instead.

GPT-5.4-mini selects the fallback at five of the six tested library sizes, and the case reproduces in 10 of 11 models at some size. Over a wider diagnostic (11 models, three attractor variants, six sizes, 20 victim tasks, 3,960 decisions), this fallback captures 265 of 1,320 decisions (20.1%), reproducing in 10 of 11 models with a per-model range of 8.3–34.2%. The two mechanisms need different remedies: local confusion calls for symmetric include/exclude boundaries on both skills, while black-hole capture calls for restricting fallback exposure and triggering clarification first.

D.3 Self-Reflection and Retry

We test self-reflection based candidate rejection. In each round the agent selects a skill, evaluates whether it fits the request, removes it from the candidate set if rejected, and reroutes over what remains. Retry time RT denotes the maximum number of selection–reflection rounds, so RT = 1 is first-choice routing and RT = 3, 5 measure cumulative success within three and five rounds. Six models, 128 tasks, $N \in \{64, 128\}$:

Retry improves cumulative success substantially, but scale dependence survives it: every model does worse at N=128 than at N=64 under every budget. In a separate multi-scale experiment, library size stays negatively correlated with cumulative routing success for every model and every RT setting (correlation magnitude ≥ 0.780). Retry therefore mitigates first-choice failure without removing the effect of library size, which is why the manager treats rejection and retry as one control among several rather than as a substitute for library-side repair.

D.4 Law-Guided Targeting Under an Equal Budget

If a manager improves accuracy simply by rewriting more descriptions, the laws add nothing. To separate the two, every non-baseline policy below edits exactly two skill descriptions. We used the discovery split of the geometry experiment to identify three consistently difficult skill families and their dominant competitors, then evaluated on 10 new library seeds per family.

Law-guided targeting fixes all nine baseline failures without introducing new errors (exact paired $p = 0.0039$) and beats both equal-budget alternatives by 16.7 pp. The gain therefore comes from allocating a fixed budget to the competition boundaries the scaling laws identify as most harmful, not from editing more text. This is what the paper means by the manager being law-derived rather than heuristic: the routing law says when to compress exposure, the geometry analysis says which

Table 14: Equal-budget comparison. All policies edit exactly two skill descriptions, so the difference is where the budget is spent, not how much of it is spent.

Policy	Accuracy	Gain per edited skill
Baseline	70.0%	—
Uniform rewriting	83.3%	+6.67 pp
Random targeting	83.3%	+6.67 pp
Law-guided targeting	100.0%	+15.0 pp

competitive structures to repair first, and the execution law says when realized state should be verified and reused.

D.5 Cross-Model Rescue Robustness

The execution law does not imply that every model receives the same rescue gain. Rescue is largest when the upstream step is likely to be correct and the downstream route has headroom. The cross-model diagnostic therefore serves a narrower role: it checks that the state-rescue pattern is systematic across model families while leaving the precise coefficient fit as diagnostic support rather than a separate headline law.

Execution-state experiment details. The execution-state experiments use ordered two-skill pairs from annotated pipelines. For each pair, we first run matched single-step routing trials for the upstream skill A and the downstream skill B under the same exposed-library regime. These trials estimate $Acc(A)$ and $Acc(B)$ without any execution artifact. The no-state paired condition then asks for the two routes jointly but still withholds upstream output; its observed joint success $Acc(A, B)$ is compared with $Acc(A)Acc(B)$ through $\Delta = Acc(A, B) - Acc(A)Acc(B)$. This is the protocol-leakage control: if pairing alone changes routing, Δ should move away from zero before any state is realized.

How to read the multiplicativity control. The no-state result is a leakage diagnostic, not a theorem of pairwise independence. Small pair-level residuals can occur because the two prompts may share wording or domain context. The key comparison is scale: these no-state residuals are small relative to the downstream changes observed only after a verified upstream artifact is inserted.
For correct-state trials, the upstream skill is actually executed. Only artifacts that pass the upstream checker, or human verification when no checker exists, enter the correct-state condition. The artifact is inserted verbatim into the downstream routing context, while the downstream gold skill and candidate library are kept matched to the no-state condition. Rescue boost is measured as the downstream success gain relative to the matched no-state downstream route, and rescue ratio is measured as the realized joint success divided by the product baseline. The rescue-law regression uses boost rate as the response and rescue potential $(1 - Acc(B))Acc(A)$ as the predictor.

Wrong-state and capability-gap robustness. The wrong-state results are also heterogeneous in a structured way. In the quality propagation audit, 11 models over 68 ordered pairs produce 23,739 downstream-scored rows. Tight dependencies lose quality when the upstream artifact is wrong rather than perfect (-7.2%), while loose dependencies show a small positive difference (+2.8%) because the downstream step can often ignore the bad upstream artifact. Independent pairs stay near zero (-0.8%). The ignored-upstream rate rises in all three groups (about +10.5% to +14.5%), so the loose-pair gain should be read as bad-state rejection or fallback, not as true repair of the wrong artifact.

The capability-gap analysis shows the same kind of structured heterogeneity. Across 11 models and 253 model-pair summaries, large-gap pairs have positive product synergy (+25.2% on average, CI half-width 11.6%), whereas small-gap pairs are near zero (+1.5%, CI half-width 3.4%). The sign is model-stable in the large-gap regime (10 positive models, none negative). Thus the thresholded G result is not used as a universal closed-form deployment rule; it summarizes where positive joint execution is most consistently observed.
The pooled rescue coefficient is reported for the same reason: it tests the state-gated form of the execution law rather than estimating a deployment constant. Different models, dependency types, and domains can change the local magnitude of α, but the predicted structure remains the same: rescue requires correct upstream state and downstream headroom, whereas wrong-state effects depend on whether the downstream checker actually needs the upstream artifact.

For wrong-state trials, we deliberately condition on upstream artifacts that fail the upstream checker or receive degraded rubric scores. Pairs are stratified by dependency type from the task graph and by whether the downstream checker requires the upstream artifact. Tight pairs require the upstream output for downstream success; loose pairs keep enough downstream task signal that the artifact can be ignored. Capability-gap trials estimate $G = Acc(A) - Acc(B)$ from matched single-step success before joint execution is evaluated. The fitted transition around $G^* \approx 0.25$ is therefore based on a pre-execution capability measure rather than on post-hoc joint-task outcomes.

E Design Principles for the Automatic Skill Manager

This section explains how the empirical laws are converted into the automatic skill manager evaluated in Sec. 7. It has three roles. First, it fixes the design vocabulary by mapping each observed failure regime to a named library-maintenance action. Second, it specifies the manager implementation: scorecards, planning, edit actions, runtime gating, and closure checks (Appendix E.1). Third, it separates the evidence for the manager into focused checks: boundary-rewrite stability (Appendix E.3), a stage-level factorial ablation (Appendix E.4), and rewrite-style selection (Appendix E.5). The central constraint is that the manager is not treated as a free-form prompt-engineering bundle: each edit is tied to a measured law variable, while the router, task set, candidate construction, prompt template, and parser are held fixed in the held-out routing evaluation.

E.1 Manager Pipeline

Table 15 gives the rulebook used by the manager. Each row starts from a failure regime measured earlier in the paper, turns it into an optimization rule, and names the mechanism that justifies the rule. The table should be read as the design contract for the manager: edits are allowed only when they target a measured source of routing or execution failure.

Table 15: Optimization rules derived from the observed routing and execution regimes. The action names are the terms used consistently in the manager description and ablations.

Failure Regime	Optimization Rule	Grounding Mechanism
Local semantic competition	**Nearest-neighbor audit**: compare new and existing skills against closest neighbors; merge or rewrite skills in the $[0.55, 0.75)$ danger band.	Local competition: confusion is driven by local crowding, not global library size.
Descriptor ambiguity	**Boundary rewrite**: add boundary conditions and counterexamples; verify with pair-level routing audits.	Boundary mechanism: boundary clauses are not generic multipliers; reciprocal exclusions reduce off-pair misses.
Anchor loss and black-hole capture	**Abstract-skill removal or narrowing**: keep broad catch-all skills out of the flat routing pool; preserve concrete operational anchors.	Drift and attractor mechanism: abstract attractors are dangerous only when prompt anchors are weak.
Pipeline fragility	**Prompt anchoring**: reintroduce user intent at mid-chain steps; order reliable upstream steps before hard downstream choices.	Routing-scale and rescue laws: mid-chain is fragile; correct realized state most strongly helps hard downstream choices.
Execution-state propagation	**Runtime context gating and closure checks**: prefer loose dependency between steps; when joint execution is needed, pair skills across a sufficient capability gap rather than as weak-tie peers.	Wrong-state and tie-dependent synergy: loose coupling limits error propagation; strong-tie pairs promote weaker steps once $G \geq G^*$, while weak-tie pairs create drag.

The automatic skill manager is an analyzer-first library maintenance loop. It does not change the base model or train a new router. It takes a structured skill library as input, computes law-aligned risks, proposes human-reviewable edits, applies approved changes, and then re-audits the before/after library. The implementation used in the experiments follows six stages: schema normalization,

scorecard construction, recommendation planning, edit application, runtime gating, and before/after audit.

Stage 1: schema normalization. Each skill record contains an identifier, name, description, examples, family label, tags, input/output metadata, and explicit anchors: `verbs`, `objects`, and `constraints`. Optional pipeline edges record upstream skill, downstream skill, dependency type (`tight`, `loose`, or `independent`), and edge weight. The manager normalizes YAML/JSON/metadata-derived skills into this schema before scoring.

Stage 2: law-aligned scorecards. The manager computes three scorecard levels. A *pair scorecard* measures lexical or embedding overlap, shared anchors, same-family membership, competition risk, merge candidate score, and weak-drag risk. A *skill scorecard* aggregates each skill's top-neighbor similarity, family conflict, anchor strength, abstraction score, black-hole risk, routing fragility, and rewrite priority. A *library scorecard* aggregates competition density, danger-zone mass, anchor-weakness mass, black-hole exposure, family-interference index, pipeline-fragility index, and predicted routing stability. These quantities correspond directly to the law variables and action names in Table 15: local competition drives nearest-neighbor audit and boundary rewrite, anchor weakness drives prompt anchoring, black-hole exposure drives abstract-skill removal or narrowing, and pipeline fragility drives runtime context gating and closure checks.

Stage 3: recommendation planning. The planner converts scorecards into pending actions rather than silently modifying the library. High local-competition risk triggers `rewrite` or `review-overlap` actions for boundary rewrite. High abstraction or black-hole risk triggers `remove`, `narrow`, or `rewrite` actions for abstract-skill removal or narrowing. Weak anchors trigger prompt-anchoring edits that add concrete verbs, objects, and constraints. Candidate-skill simulation compares a proposed skill to the existing library and returns `reject`, `add-with-review`, or `approve-candidate` based on nearest overlap, same-family hits, anchor strength, abstraction score, and predicted routing stability delta.

Stage 4: library edits. Approved actions are applied as explicit library edits. A boundary `rewrite` replaces the skill description and anchor fields with narrower verbs, objects, constraints, and reciprocal exclusions. A `merge` action keeps a target skill, absorbs examples/tags/anchors from dropped skills, and removes redundant entries. A `remove` or `narrow` action handles broad catch-all skills that create abstract-attractor risk. After applying a plan, the manager recomputes all scorecards and emits a before/after diff with metric deltas for competition density, danger-zone mass, anchor weakness, black-hole exposure, and predicted routing stability.

Stage 5: runtime context and closure checks. For downstream agent runs, the manager also supports runtime context selection. It scores skills against the task instruction using matches in names, tags, anchors, descriptions, families, and domain profiles, then exposes a domain-gated context rather than the full flat library. Local artifact tasks receive smaller context limits to reduce irrelevant tool/vendor distraction. A closure checker infers required output artifacts from the task or from explicit task IDs, then prevents completion when required files are missing or empty. This implements the execution-law constraint: correct state must be present before downstream decisions or final completion are trusted.

Stage 6: before/after audit. The complete loop is: load the library; score skill, pair, and library risks; plan rewrite/merge/remove actions; apply approved edits; diff before/after scorecards; and evaluate held-out routing plus downstream execution. In the experiments, this loop produced the optimized library evaluated in Sec. 7. The implementation is deliberately conservative: every edit is grounded in an observed law variable, and the router is held fixed so gains can be attributed to library geometry rather than model retraining.

E.2 Evidence Map for the Manager

The auto-skill-manager has more than two optimization stages: it audits neighbors, rewrites boundaries, strengthens anchors, merges or reviews overlapping skills, narrows or removes abstract skills, gates runtime context, and checks execution closure. The appendix evidence does not claim to decompose every component independently. Instead, it uses three focused checks that cover the main claims needed for Sec. 7.

Runtime context gating and closure checks are part of the downstream transfer setup; they are not treated as separate routing-law ablation factors. This separation is deliberate. The held-out

Table 16: Stability of the bilateral boundary rewrite over 15 prospectively specified pairs. Positive accuracy counts report pairs with boundary accuracy above the original condition; outside-miss and local-support counts report pairs moving in the desired direction.

Analysis unit	Accuracy gain	Outside-miss change	Local-support change
GPT-4o-mini	$+11.3\%$	-19.7%	$+19.7\%$
GPT-5.4-mini	$+9.3\%$	-9.7%	$+9.7\%$
Pooled pairs	$+10.3\%$	-14.7%	$+14.7\%$

routing ablation isolates the two largest library-surface actions under a fixed router, prompt template, parser, and task set. The downstream transfer uses the auto-skill-manager because completed benchmark tasks also require runtime context selection and closure checks. We therefore report component evidence for the main routing-law actions and treat runtime execution controls as part of the downstream system configuration, not as additional proof that each manager component has an independent causal effect.

E.3 Boundary-Rewrite Stability Details

Table 16 reports the model-level and pair-level stability checks for the 15-pair boundary-rewrite probe. Each probe pair contains two nearby skills from the danger zone: the gold skill and its most plausible local competitor. The rewrite is *bilateral*: we add reciprocal boundary clauses to both descriptions, so each skill states not only what it does, but also how it differs from the paired neighbor. For example, if two skills both mention repository analysis, the rewritten descriptions explicitly separate the trigger conditions, inputs, and outputs that should send a task to one skill rather than the other.

We track three quantities. *Accuracy gain* is the ordinary change in selecting the gold skill. *Outside-miss change* counts errors that leave the intended two-skill neighborhood entirely: the router selects some third skill, not the gold skill or its paired local competitor. *Local-support change* is the complementary mass assigned to the intended neighborhood, i.e., probability that the router selects either the gold skill or the paired competitor. A positive local-support change does not by itself mean the router is correct; it means the rewrite has moved attention from diffuse unrelated alternatives back to the relevant local decision.

The stable effect across models is therefore *localization*. Boundary text first pulls diffuse outside-pair errors back into the intended two-skill neighborhood. Accuracy then improves only when the localized mass lands on the correct member of the pair rather than the competitor. This is why the table reports both outside-miss reduction and accuracy gain: the first measures whether the rewrite restored the right neighborhood, and the second measures whether it also resolved the within-pair boundary.

E.4 Design Evidence: Factorial Ablation of Bundled Optimization

The prospective evaluation (Sec. 7) reports a bundled 20.4% gain from the optimized library. The ablation below is a stage-level mechanism probe, not a complete decomposition of every manager component. It toggles one routing-law stage and one attractor-law stage. *Boundary rewrite* is the ablated form of the nearest-neighbor-audit/boundary-rewrite action in Table 15: crowded pairs are identified first, and rewriting then adds boundary clauses, concrete anchors, and scope exclusions. *Abstract-skill removal* represents the anchor-loss/black-hole action: broad catch-all skills are removed from the flat routing pool or narrowed so that they no longer absorb vague prompts. Table 17 reports the 2×2 factorial ablation on the 1,600-task held-out set at $N=150$, keeping the router, task set, distractor construction, prompt template, and parser fixed.

Boundary rewrite alone accounts for 12.8% of the accuracy gain and 12.6% of the hijack reduction. Abstract-skill removal alone contributes 4.9% accuracy and 5.1% hijack reduction. The two effects sum to 17.7%; the remaining 2.7% is the interaction, consistent with the dual-trigger mechanism: removing abstract skills is most effective when descriptions are already concrete enough to prevent vague-prompt drift. Thus the ablation validates two representative law channels used by the manager: local pairwise confusion and abstract-attractor capture. It should be read as a mechanism probe for two stages of the optimization pipeline, not as an exhaustive list of all manager actions.

Table 17: Stage-level 2×2 ablation at $N=150$ on the 1,600-task held-out set. "Boundary Rewrite" probes the local-competition stage after nearest-neighbor audit; "Abstract-Skill Removal" probes the anchor-loss/black-hole stage. The auto-skill-manager also includes prompt anchoring, overlap review, runtime context gating, and closure checks.

Boundary Rewrite	Abstract-Skill Removal	Accuracy	Hijack
No	No	71.3%	22.4%
Yes	No	84.1%	9.8%
No	Yes	76.2%	17.3%
Yes	Yes	91.7%	4.1%

E.5 Design Evidence: Rewrite Style Selection

The boundary-rewrite stage in the ablation still leaves an implementation choice: how should rewritten skill descriptions be expressed? This check is an implementation choice inside the boundary-rewrite stage, not a separate causal claim. Table 18 compares five rewrite styles by the same library-geometry quantities used in the routing law: stability, anchor weakness, danger-zone mass, and black-hole exposure. Longer structured cards can improve a manager-native routing check while adding shared boilerplate that leaves competition density or danger-zone mass less improved. The final label-free compact card keeps the useful identity, action, and boundary information while minimizing reusable template language.

Table 18: Rewrite-style selection using library-geometry diagnostics. Stability is better when higher; anchor weakness, danger-zone mass, and black-hole exposure are better when lower. Manager-native routing accuracy is used only to select the rewrite style.

Rewrite form	Actions	ΔStab.	ΔAnchor	ΔDanger	ΔBlack-hole	ΔComp.	Mgr. Acc.	Reg.
Pair boundary rewrite	72	+0.020	-0.048	-0.001	-0.017	-0.001	0.811→1.000	0/180
Long structured card	60	+0.003	-0.039	+0.008	-0.011	+0.029	0.778→1.000	0/180
Compact labeled card	60	+0.012	-0.040	-0.003	-0.014	+0.010	0.783→0.994	0/180
Low-boilerplate labeled card	60	+0.017	-0.040	-0.008	-0.015	+0.000	0.783→0.994	0/180
Label-free compact card	60	+0.018	-0.040	-0.009	-0.017	-0.003	0.783→0.994	0/180

This analysis only selects the rewrite style used inside the boundary-rewrite stage of the held-out optimization. The causal claim remains the factorial result in Appendix E.4 and the prospective held-out evaluation in Sec. 7.

F Downstream Transfer: CLAWBENCH & CLAWMARK

This section reports the downstream skill-based agent checks referenced in Sec. 7. These runs test whether the law-guided library changes transfer beyond diagnostic routing tasks. They are not used to establish the routing or execution laws, which are measured in controlled probes.
The downstream suites are intentionally broader than the controlled routing probes: CLAWBENCH [79] covers 242 task entries across 18 tool-oriented domains, while CLAWMARK [78] covers 100 held-out tasks across 13 role-oriented domains (Table 19). This coverage table describes the transfer data only; the routing laws are fitted on the separate controlled routing sweeps where task identity and domain mixture are held fixed across library sizes.

Setup. Using `gpt-5.4-mini` as the base model, we compare raw and optimized skill-library configurations under matched downstream execution settings. The auto-skill-manager uses the same law-guided changes as the prospective routing optimization: nearest-neighbor audit, boundary rewrite, abstract-skill removal or narrowing, and prompt anchoring, etc. The downstream comparison holds the base model, execution prompt family, task split, and scoring rule fixed between raw and optimized libraries. Mean pass rate is computed as the domain-macro average from the benchmark checkers within each downstream suite and is reported separately for CLAWBENCH and CLAWMARK.

Transfer measurement details. Held-out routing transfer is measured on the pre-specified 1,600-task set at $N=150$. The raw-library and optimized-library conditions use the same model, prompt template, parser, task list, candidate-set construction, and scoring rule. The optimized condition

Table 19: Task-domain coverage of the downstream transfer data.

Suite	Tasks	Domains	Largest domains
CLAWBENCH	242	18	document editing (18), email (18), cross-domain (17), data analysis (17), workflow automation (17), plus 13 smaller domains
CLAWMARK	100	13	research assistant (15), content operation (12), HR (11), e-commerce (9), journalist (8), product management (8), plus 7 smaller domains

Table 20: Domain-level downstream transfer on CLAWMARK and ClawBench. Entries are pass@4 rates; the Avg. row reports the domain-macro average.

Setting	Tasks	Raw	Auto-Skill-Manager
Clinical Ass.	4	37.5	43.8
Content Ope.	12	22.9	29.2
E-Com.	9	22.2	30.6
EDA	1	25.0	25.0
Exe. Ass.	7	28.6	35.7
HR	11	36.4	43.2
Insurance	7	32.1	39.3
Inv Ana.	6	29.2	37.5
Jour.	8	28.1	34.4
Legal Ass.	6	20.8	25.0
Proj. Man.	8	18.8	21.9
Real Est.	6	37.5	45.8
Research Ass.	15	30.0	36.7
Avg.	–	28.4	34.5

changes only the library surface produced by the law-guided manager: boundary rewrites, nearest-neighbor audit, abstract-skill removal or narrowing, and prompt anchoring. We report two routing outcomes. Accuracy is the fraction of tasks routed to the gold skill. Hijack rate is the fraction routed to an exposed but non-gold skill, so hallucinations are not mixed into the in-library failure rate.

Downstream transfer uses matched raw-library and auto-skill-manager-library runs under the same execution setting. This separation matters because downstream success depends both on the library surface and on the execution policy that preserves or loses intermediate state. We therefore treat the downstream result as a transfer check for the optimized library, not as evidence used to fit the routing or execution laws. The comparison is therefore deliberately narrower than a full agent-system ablation: it asks whether the same library-side changes that improve controlled routing remain useful when benchmark checkers score completed tasks. It does not isolate every runtime component, and it should not be read as evidence that library editing alone solves execution-state propagation.

Result and interpretation. The optimized library improves downstream mean pass rate from 49.3% to 61.6% on CLAWBENCH and from 28.4% to 34.5% on CLAWMARK; per-domain breakdowns are in Tables 21 and 20. This pattern is useful but bounded: the library changes improve downstream transfer, while downstream success still depends on execution-state management. This is consistent with the paper's two-law picture: library structure controls routing-side probability mass, and execution-state management controls whether downstream state is usable.

Table 21: Domain-level downstream transfer on CLAWBENCH. Entries are pass@4 rates; the Avg. row reports the domain-macro average.

Setting	Tasks	Raw	Auto-Skill-Manager
Calendar	15	91.7	93.3
Code Ass.	15	80.0	86.7
Comm.	15	80.0	86.7
Cross Dom.	17	33.8	41.2
Data Ana.	17	77.9	88.2
Database	5	20.0	40.0
Debug.	5	0.0	20.0
Doc. Edit.	18	51.4	73.6
Email	18	48.6	55.6
File Ope.	15	65.0	86.7
Memory	15	58.3	75.0
Multimodal	15	56.7	81.7
Plan.	5	10.0	20.0
Real Tools	5	15.0	20.0
Security	15	56.7	61.7
System Admin	15	43.3	61.7
Web Brows.	15	41.7	56.7
Workflow Auto.	17	57.4	60.3
Avg.	–	49.3	61.6

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the two coupled empirical laws and the main experimental scope; Sec. 3 defines the evaluated models, skill library, tasks, and protocols.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Sec. 8 discusses scope, limitations, and future work.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The paper presents empirical laws rather than formal theorems; Appendix A states the modeling assumptions and derivations used to interpret them.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Sec. 3 and Appendix B provide the experimental setup, model list, library construction, and prompt templates.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code, prompts, seeds, evaluation scripts, routing traces for headline conditions, and aggregate result tables are released with the paper.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: Sec. 3 and Appendix B.2 describe the zero-shot inference setting, library construction, annotation protocol, splits, and fitted quantities.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: The main controlled findings report confidence intervals, sample sizes, and statistical tests.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: The experiments call commercial LLM APIs rather than training or serving models locally, as described in Sec. 3; no GPU or cluster allocation is involved.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The study design and annotation setting are described in Sec. 3.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Sec. 8 discusses broader impacts and scope.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Sec. 3 describes the skill-library sources and release plan.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Existing repositories, LLM APIs, benchmarks, and embedding models are cited in Sec. 3 and the reference list.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Sec. 3 and Appendix B.2 document the skill library and annotation protocol; the release artifact documents construction and license provenance.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: Sec. 3 and Appendix B.2 describe the annotation process and the written guideline used for skill and task labels.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The annotation setting is described in Sec. 3; No IRB approval is required for this skill-labeling task.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research?

    Answer: [Yes]

    Justification: LLM usage is central to the study and is described in Sec. 3; GPT-5.4 was used for task synthesis, and 15 LLMs are the core experimental subjects for routing and execution measurements.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.