When Does a Skill Add Value? Task-Conditional Gain Prediction for Selective Skill Use

Anonymous authors
Paper under double-blind review

Abstract

Agent skills are expected to improve task performance. Yet we find that they often provide no benefit, and can even hurt performance while incurring additional token costs. Can we predict whether a skill will help before the agent acts? We introduce SkillDelta, a framework for predicting task-conditional skill gains from paired executions of the same agent with and without the skill. A local predictor transfers these historical gains to new tasks without retraining the agent. Under explicit transfer assumptions, our analysis links support coverage, representation mismatch, and execution noise to prediction error and decision regret. Across five benchmarks and three target agents, paired history improves observed-gain ranking over skill-assisted outcomes alone in 12 of 15 settings. At matched expected skill-use rates, SkillDelta improves success over random activation in all 15 settings, with an average absolute gain of 4.3%. Most of this advantage comes from allocation across task groups. Evidence for additional within-group selection value is strongest on ToolQA and weaker elsewhere. Code and reproduction materials are available at this anonymous repository.

1 Introduction

Agent skills promise to improve task execution through reusable procedures without retraining the agent (Zhang et al., 2025). Yet relevant instructions can duplicate existing ability, reduce success, or add tokens without benefit. A tool call invokes an operation; a skill supplies a workflow that can guide several tools. Models can learn tool use during training (Schick et al., 2023; Eleti et al., 2023), whereas users can supply skills afterward, with descriptions that may overstate or misrepresent their benefits. *Can we predict whether a supplied skill will improve the agent's chance of success on its current task, before execution?*

Existing work measures skill utility (Li et al., 2026; Han et al., 2026; Liu et al., 2026), improves retrieval (Zheng et al., 2026), and optimizes skill content (Yang et al., 2026). SRA-Bench further shows that agents struggle to judge when skills are needed (Su et al., 2026). However, finding a relevant skill or succeeding with it does not establish benefit: the agent may already solve the task without it. Model routers compare answering models (Ong et al., 2025; Somerstep et al., 2025); our question holds the agent fixed and asks how its success changes when given a supplied skill. This requires predicting *incremental gain* before either outcome is observed.

Predicting benefit requires distinguishing an intervention's effect from its outcome (Shalit et al., 2017). Agent tasks let us measure this difference through *paired execution history*: running the same agent under use and skip conditions, with repetition improving precision. Deployment raises a generalization question: when can gains measured on historical tasks predict benefit on a new task? Topic similarity alone is insufficient; useful history must cover the target, and the representation must capture similarity in skill benefit.

We develop SkillDelta, a local predictor that aggregates paired gains from eligible historical tasks without retraining the agent. Skill or family information restricts the support, and fixed task embeddings identify neighbors. Building on local regression analysis (Kpotufe, 2011), we connect support coverage, representation mismatch, and execution noise to prediction error and decision loss under explicit transfer assumptions. The analysis explains why expanding a history bank and

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

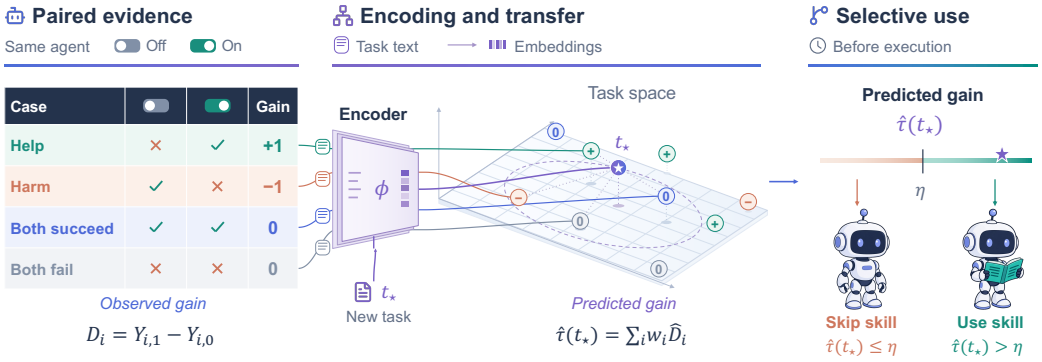

Figure 1: SKILLDELTA predicts skill gains before execution. Paired outcomes provide observed gains, which are transferred through nearby task embeddings to predict gain on a new task. Thresholding this prediction determines whether to use the skill. Only task text is encoded; the geometry is schematic.

repeating its executions address different sources of error. An accompanying DeepSeek Harness plugin gates skill injection before the first model step.

We test both the value of paired evidence and the decisions it supports across five benchmarks and three agents. Paired history improves observed-gain ranking over skill-assisted outcomes alone in 12 of 15 settings. At matched expected skill-use rates, selective use yields an average absolute success-rate gain of 4.3% over random activation (Figure 6). This advantage combines allocation across task groups with selection within groups; the clearest within-group benefits occur on ToolQA (Figure 7). Evidence-scaling diagnostics examine how support, neighborhood size, and repetitions affect prediction error.

Contributions.

1. We formulate pre-execution skill-gain prediction, distinguishing incremental benefit from skill-assisted success, and characterize how coverage, representation mismatch, and execution noise constrain transfer from paired history.
2. We introduce SKILLDELTA, a local gain estimator that guides skill use before execution without retraining the agent, and provide a DeepSeek Harness plugin for integration.
3. Paired history enables higher success than random activation at matched expected skill-use rates across five benchmarks and three agents, chiefly through better allocation across task groups.

Our code and reproduction materials, including a SKILLDELTA plugin for DeepSeek Harness, are available in the anonymous repository.

2 PROBLEM DEFINITION: TASK-CONDITIONAL SKILL GAIN

2.1 EXPECTED GAIN AND PAIRED OBSERVATIONS

Let $t \in \mathcal{T}$ be a task, $s = s(t)$ its supplied applicable skill or fixed skill bundle, and s_0 the no-skill condition. Both conditions use the same agent and execution protocol, differing only in skill injection. For a binary success outcome $Y(t)$, define

$$\begin{aligned} \mu_1(t,s) &= \mathbb{E}[Y(t) \mid t, s] = \Pr(Y(t) = 1 \mid t, s), \\ \mu_0(t,s_0) &= \mathbb{E}[Y(t) \mid t, s_0] = \Pr(Y(t) = 1 \mid t, s_0), \\ \tau(t,s) &= \mu_1(t,s) - \mu_0(t,s_0). \end{aligned} \tag{1}$$

The subscripts label execution conditions. The gain $\tau(t,s)$ is the change in the agent's success probability when supplied with the skill. When the supplied skill is fixed by context, we write $\mu_1(t) \equiv \mu_1(t,s)$, $\mu_0(t) \equiv \mu_0(t,s_0)$, $\tau(t) \equiv \tau(t,s)$, and $\widehat{\pi}(t) \equiv \widehat{\pi}(t,s)$.

For example, two tasks may each have 80% success with the skill. If their no-skill success probabilities are 80% and 40%, their gains are zero and 40%, respectively. Predicting skill-assisted success alone treats these tasks alike, although only the second benefits from the skill.

For support task t_i with skill $s_i = s(t_i)$, the success fractions $\widehat{\mu}_1(t_i, s_i)$ and $\widehat{\mu}_0(t_i, s_0)$ from r_i repetitions per condition estimate the probabilities in Equation (1). Their difference is the observed paired gain:

$$\widehat{D}_i = \widehat{\mu}_1(t_i, s_i) - \widehat{\mu}_0(t_i, s_0), \qquad \mathbb{E}[\widehat{D}_i \mid t_i, s_i] = \tau(t_i, s_i). \tag{2}$$

With one paired execution, $\widehat{D}_i \in \{-1, 0, 1\}$ records realized harm, no change, or help. Pairing matches the task across conditions; it does not require identical execution trajectories. Observed differences combine expected gains with execution noise. Independent repetitions under a fixed execution distribution reduce the noise in $\widehat{D}_i$.

2.2 Selective use as a downstream application

The policy $\widehat{\pi}_\eta$ maps each task to an action—use (1) or skip (0)—by comparing its predicted gain $\widehat{\tau}(t, s)$ with threshold η:

$$\widehat{\pi}_\eta(t, s) = \mathbf{1}\{\widehat{\tau}(t, s) > \eta\}. \tag{3}$$

At $\eta = 0$, the policy selects positive predicted gains; a larger threshold requires a larger predicted benefit. We evaluate success rate and token usage against Always-off (skip the skill) and Always-on (use it). Section 3 develops the gain estimator; Section 4 specifies the evaluation protocol.

3 Method and Theoretical Analysis

SkillDelta averages observed gains from similar historical tasks to predict whether a supplied skill will help a new task. The target provides its description and skill identity; execution outcomes come only from support tasks. We analyze when this transfer is reliable, describe the predictor, and connect its errors to decision loss. Proofs and extensions are in Appendix A.

3.1 What can transfer?

Identical historical outcomes can be consistent with either benefit or harm on a new task (Appendix A.2). Predicting its gain therefore requires a relation between support and target effects. For each eligible pair, we assume

$$|\tau(t) - \tau(t')| \le L d(\phi(t), \phi(t')) + \varepsilon_\phi. \tag{4}$$

Here ϕ represents task text and d measures distance. The residual ε_ϕ accounts for gain differences left unexplained by the representation distance.

Consider a calculator skill and two questions about the same medical quantity: one asks for a definition, the other for a unit conversion. Their topics are similar, but the calculation procedure may help only the second. Predicting gains requires a representation that captures such differences in benefit, with ε_ϕ allowing for those it misses.

3.2 What limits gain prediction?

For support task i, write $\widehat{D}_i = r_i^{-1} \sum_{\ell=1}^{r_i} D_{i\ell}$, where $D_{i\ell} \in [-1, 1]$ are independent across tasks and repetitions, with mean $\tau(t_i)$. Let nonnegative weights sum to one and let $\widehat{\tau}(t) = \sum_i w_i(t) \widehat{D}_i$. Define the weighted radius $\rho_w(t) = \sum_i w_i(t) d(\phi(t), \phi(t_i))$.

Theorem 1 (Local gain-prediction bound)**.** *Under Equation (4), for any fixed target t, weights independent of support outcomes, and $\delta \in (0, 1)$, with probability at least $1 - \delta$,*

$$|\widehat{\tau}(t) - \tau(t)| \le \underbrace{L\rho_w(t)}_{\text{coverage}} + \underbrace{\varepsilon_\phi}_{\text{representation residual}} + \underbrace{\sqrt{2 \log(2/\delta) \sum_i \frac{w_i(t)^2}{r_i}}}_{\text{execution noise}}. \tag{5}$$

The same ε_ϕ appears in both equations because averaging with weights that sum to one preserves the residual bound. The bound distinguishes three ways to change the evidence. A larger support pool can supply closer neighbors. Using more neighbors can reduce noise but increase transfer bias;

repeating executions improves gain precision at fixed coverage and representation. These distinctions guide the evidence-scaling diagnostics. The bound's numerical strength at small neighborhoods and the additional conditions for future-task guarantees are discussed in Appendices A.3 and A.5.

3.3 Local prediction from paired evidence

Figure 1 illustrates the default signed-gain predictor.

Measure paired gains. The left table separates help, harm, success in both conditions, and failure in both. Both unchanged outcomes have zero gain; the observed differences $\widehat{D}_i$ provide the support labels.

Transfer gains from nearby tasks. In the middle panel, the support rule uses skill or family metadata to select eligible tasks, for example those sharing group $g(t)$. Question-only vectors $\phi(t)$ determine similarity within that scope. These steps play different roles: the support rule specifies which historical skill interventions can inform the target, while the encoder locates similar questions among them. Sharing a skill or family makes tasks eligible; it does not assign them equal gains. Neither observed successes nor gains enter the encoder or determine the neighbor weights. We select up to k nearest tasks, shown around the star, and weight their gains by clipped cosine similarity:

$$\widehat{\tau}(t,s) = \sum_{i \in N_k(t,s)} w_i(t,s)\widehat{D}_i, \qquad w_i(t,s) = \frac{[\cos(\phi(t),\phi(t_i))]_+}{\sum_{j \in N_k(t,s)}[\cos(\phi(t),\phi(t_j))]_+}. \tag{6}$$

If all clipped similarities are zero, we use uniform weights. With shared weights, this estimator also equals $\widehat{\mu}_{1,w}(t) - \widehat{\mu}_{0,w}(t)$, where $\widehat{\mu}_{a,w}(t) = \sum_i w_i(t)\widehat{\mu}_a(t_i)$.

For illustration, neighbors with gains $+1$, 0, and -1 and weights 0.5, 0.3, and 0.2 give $\widehat{\tau} = 0.3$. Both helpful and harmful evidence affect the estimate. Zero-gain neighbors retain their weight, bringing the prediction toward zero rather than disappearing from the calculation.

Predict gain and choose an action. The right panel applies Equation (3) to the signed estimate. Let $\mathcal{I}_{-t}$ index historical support tasks excluding target t. Algorithm 1 returns the estimate and action; with no eligible support, it skips the skill and reports zero as a fallback score. All main experiments use this signed-gain rule; shared support and threshold settings appear in Section 4.

Algorithm 1 Default signed-gain prediction and selective skill use

Input: Paired outcomes $Y_{i,a,\ell}$ for support tasks $(t_i, s_i)_{i \in \mathcal{I}_{-t}}$, target (t, s), and representation ϕ
Input: Scope $\mathcal{I}(t,s)$, neighborhood size k, and threshold η
$\mathcal{I} \leftarrow \mathcal{I}(t,s) \cap \mathcal{I}_{-t}$
if $\mathcal{I} = \varnothing$ **then return** $(0, 0)$
$\widehat{D}_i \leftarrow r_i^{-1} \sum_{\ell=1}^{r_i} (Y_{i,1,\ell} - Y_{i,0,\ell})$ for $i \in \mathcal{I}$
$N_k \leftarrow$ indices of the $\min(k, |\mathcal{I}|)$ largest $\cos(\phi(t), \phi(t_i))$, $i \in \mathcal{I}$
$q_i \leftarrow [\cos(\phi(t), \phi(t_i))]_+$ for $i \in N_k$
$w_i \leftarrow q_i / \sum_{j \in N_k} q_j$ if the denominator is positive; otherwise $w_i \leftarrow 1/|N_k|$
$\widehat{\tau}(t,s) \leftarrow \sum_{i \in N_k} w_i \widehat{D}_i$; $\widehat{\pi}(t,s) \leftarrow \mathbf{1}\{\widehat{\tau}(t,s) > \eta\}$
return $(\widehat{\tau}(t,s), \widehat{\pi}(t,s))$

The historical bank stores task descriptions, skill or family identities, fixed question vectors, and paired outcomes. Each prediction can be traced to its retrieved records and weights; new records can be added without refitting model parameters. At deployment, the upstream system supplies a new task and candidate skill. SkillDelta encodes the question, retrieves eligible neighbors, and scores the skill before any target execution, without an additional generative self-judgment call. The bank can serve subsequent tasks under the same agent and skill conditions. We release a DeepSeek Harness plugin that follows this workflow with uniform neighbor averaging, injecting the supplied skill before the first model step when predicted gain exceeds the threshold.

3.4 Decision accuracy and loss

Let $\widehat{a}_\eta(t) = \widehat{\pi}_\eta(t)$ be the action chosen by Equation (3), $a_\eta(t) = \mathbf{1}\{\tau(t) > \eta\}$ its oracle, and $B(t)$ the right-hand side of Equation (5).

Corollary 2 (Threshold decision)**.** *On the event in Theorem 1, $\widehat{a}_\eta(t) = a_\eta(t)$ whenever $|\tau(t) - \eta| > B(t)$.*

The margin $|\tau(t) - \eta|$ determines how much error a decision can tolerate. To measure the cost when a decision is wrong, define threshold-relative regret as

$$R_\eta(\widehat{a}, t) = |\tau(t) - \eta| \, \mathbf{1}\{\widehat{a}_\eta(t) \neq a_\eta(t)\}. \tag{7}$$

At $\eta = 0$, this is the success probability lost relative to choosing with knowledge of the true gain.

Proposition 3 (Routing regret is controlled by gain error)**.** *For every target task and threshold η,*

$$R_\eta(\widehat{a}, t) \leq |\widehat{\tau}(t) - \tau(t)|. \tag{8}$$

Consequently, under the event in Theorem 1, $R_\eta(\widehat{a}, t) \leq B(t)$.

For example, at a zero threshold, skipping a skill that raises success from 50% to 51% loses less than skipping one that raises it from 50% to 70%. AUROC evaluates ranking by observed gain sign; regret also depends on gain magnitude. This motivates evaluating prediction quality alongside policy success and token usage. Proofs and extensions appear in Appendices A.4 and A.5.

Matched-use-rate comparison isolates the value of this task-level choice. Relative to always skipping, a policy's expected success gain on a fixed collection is the sum of the gains on selected tasks divided by the total number of tasks. A random policy with the same expected use rate assigns skill uses independently of those gains. A success advantage over that reference therefore measures the value of selecting the recipients, beyond simply using the skill more often.

4 Experiment and Results

4.1 Benchmarks and target-agent stacks

Table 1: Task and skill inventory.

Benchmark	Tasks	Skills
ToolQA	1,430	14
MedCalc-Bench	1,100	55
BigCodeBench	1,136	139
LogicBench	760	19
SpreadsheetBench	399	1‡

‡External skill source, not from the SRA-Bench skill corpus.

We evaluate four SRA-Bench domains (Su et al., 2026) and SpreadsheetBench (Ma et al., 2024) (Table 1) under three fixed target-agent stacks: `Qwen-Turbo`, `GLM-5.3-Flash`, and `DeepSeek-V4-Flash`. Each task is executed with and without its supplied skill, using temperature zero, frozen artifacts, and benchmark-provided evaluators.

The skill-enabled condition uses the supplied skill for ToolQA and LogicBench, a calculator skill for MedCalc-Bench, all gold skills for BigCodeBench, and an externally optimized spreadsheet skill with a pinned evaluation harness. The main Qwen-Turbo and GLM-5.3-Flash results use one paired execution per task, except LogicBench, which uses three-repeat means. DeepSeek-V4-Flash uses three-repeat means on all five benchmarks. RQ3 uses repeated-execution panels to study support-gain precision. Execution profiles and token ledgers are frozen separately for each stack; outcomes are never pooled across stacks. Protocol details, artifact audits, and task–skill structure are reported in Appendices B and C.

4.2 Within-inventory leave-one-task-out evaluation

We retrospectively evaluate the completed inventories using leave-one-task-out prediction: both outcomes of each target are excluded from its support. All three agents share question-only `text-embedding-3-small` vectors (1,536 dimensions), L2-normalized before cosine similarity. On every benchmark, the predictor averages signed gains from up to $k = 6$ same-family neighbors, using nonnegative cosine weights, and enables the skill when the score is strictly positive. If fewer than six neighbors exist, it uses all available neighbors; empty support yields score zero and



Figure 2: Paired history improves observed-gain ranking in most settings. Bars show AUROC for positive versus nonpositive observed gains; dashed lines indicate chance performance. Paired and with-skill-only predictors share target-excluded, same-family $k = 6$ neighborhoods and weights.

Skip skill. This common rule removes benchmark-specific thresholds and aggregation choices (Appendix B.2). Here, "family" denotes the recorded task group: supplied-skill families, BigCodeBench focal groups, or SpreadsheetBench task types (Appendix C).

Prospective panels test new targets under their historical frozen settings (Appendix D). After fixing each action, we take the recorded success and execution tokens from the skill-enabled condition if the action is use, and from the no-skill condition otherwise. This reconstructs policy outcomes without additional task executions.

Direct self-judgment gives the target agent the question and the same skill text as the skill-enabled condition, without historical outcomes or support statistics. One shared prompt requests a binary use/no-use action; the fixed parser accepts a unique decision label and maps ambiguous or empty responses to no-use.

An outcome-free relevance gate uses task–skill TF–IDF cosine similarity, with a median threshold fixed before outcomes are read. A with-skill-only predictor averages skill-enabled outcomes over the same neighbors and weights, isolating the value of paired gains. TF–IDF kNN and family-mean controls receive the same paired support as SKILLDELTA (Appendix B).

Evidence-scaling diagnostics. RQ3 varies support-pool size, neighborhood size, or retained execution repetitions and measures gain MAE against fixed empirical targets. The repetition sweep holds neighborhoods fixed; full settings are in Appendix B.5.

4.3 EMPIRICAL RESULTS

4.3.1 RQ1: SKILL-GAIN STRUCTURE AND PREDICTION

Both helpful and harmful outcomes occur under the same supplied skill (Appendix C). This variation, hidden by skill-level averages, motivates predicting task-level gains.

RQ1 measures AUROC for positive ($\widehat{D} > 0$) versus nonpositive observed gains on held-out tasks, probing whether representation and support preserve skill-effect structure (Theorem 1).

Across the 15 stack–benchmark panels in Figure 2, paired evidence improves AUROC over the no-outcome reference on 12 panels and over the with-skill-only reference on 12; it exceeds both on 10. The gains are most consistent on ToolQA and MedCalc-Bench. BigCodeBench provides the clearest contrast: on all three stacks, its outcome-free lexical score ranks gains better than the paired predictor. The with-skill-only predictor exceeds the paired predictor on Qwen-Turbo BigCodeBench and SpreadsheetBench, and on DeepSeek-V4-Flash ToolQA.

Can the agent judge for itself? Direct self-judgment enables the skill on over 91% of tasks in 14 of 15 panels; DeepSeek-V4-Flash SpreadsheetBench is the exception at 44.4%. Its policy is therefore usually close to Always-on and offers little gain-sign discrimination. SKILLDELTA's continuous

Table 2: Task success and token change under the unified protocol.

		Task success (%)				Token savings (%)	
Model	Benchmark	Off	On	Self-judge	SKILLDELTA	Self-judge	SKILLDELTA
Qwen-Turbo	ToolQA	28.6	44.7	44.5	44.5	-5.3	10.4
	MedCalc-Bench	43.2	71.7	70.5	70.6	-65.8	14.8
	BigCodeBench	43.7	51.4	51.4	47.4	-98.2	49.4
	LogicBench	75.5	87.2	87.4	85.2	-78.7	29.1
	SpreadsheetBench	15.5	19.8	18.3	17.8	-72.3	36.2
GLM-5.3-Flash	ToolQA	38.7	45.0	45.1	47.8	-2.6	23.8
	MedCalc-Bench	78.4	83.3	83.3	88.3	-27.8	41.1
	BigCodeBench	52.0	63.3	63.2	59.8	-65.4	25.4
	LogicBench	69.2	85.1	84.3	83.9	-91.0	8.2
	SpreadsheetBench	46.1	73.4	68.4	72.4	-26.0	4.8
DeepSeek-V4-Flash	ToolQA	25.5	38.3	37.9	37.9	-5.8	3.0
	MedCalc-Bench	74.3	89.4	88.8	90.1	-66.1	12.4
	BigCodeBench	49.3	61.5	61.5	57.7	-85.3	26.5
	LogicBench	74.4	85.1	84.0	83.6	-85.6	19.5
	SpreadsheetBench	46.5	61.4	49.4	59.9	-28.5	8.0

Token savings are relative to Always-on. SKILLDELTA uses target-excluded, same-family cosine-weighted $k = 6$ signed-gain scores and threshold zero. Its execution costs exclude history collection and encoding; self-judge includes its routing call.

scores exceed the binary self-judge scores in AUROC on all 15 panels. Its decision accuracy and the judge's routing overhead appear in Table 3.

4.3.2 RQ2: DECISION VALUE OF GAIN PREDICTION

At the zero-threshold operating points in Table 2, SKILLDELTA improves observed success over Always-off and saves tokens relative to Always-on on all 15 panels; 14 task-bootstrap intervals for the success gain exclude zero (Table 5).

At the same expected skill-use rate, SKILLDELTA achieves higher success than random activation in all 15 panels, with an average absolute gain of 4.3% (Figure 6). Matching each recorded task group's use rate separates this advantage into 0.61% from within-group selection and 3.72% from between-group allocation (Figure 7). The larger allocation term reflects directing the same expected number of skill uses toward groups with higher average observed gains. This answers a practical question—which kinds of tasks merit skill use—even when ranking tasks within each group is difficult.

Finer task selection is clearest on ToolQA: its three-agent mean is 4.41%, and all three conditional intervals exclude zero. Elsewhere, within-group intervals include zero, except for a negative interval on Qwen-Turbo BigCodeBench.

These decisions trade some success for lower execution cost. Across the 15 equally weighted panels, mean success is 63.12% versus 64.04% for Always-on, with 20.8% mean token savings (range: 3.0–49.4%). Success is below Always-on in 12 of the 15 panels. Direct self-judgment increases total tokens by 2.6–98.2% relative to Always-on, including its routing call.

Figure 3 varies η for ToolQA, MedCalc-Bench, and LogicBench. Higher thresholds reduce skill use, with success–cost trade-offs that depend on the benchmark and agent. The default stars reproduce Table 2.

Ridge controls use fixed grouped splits for all three agents (Table 6). Dual Ridge adds an incremental-token head to gain regression; matched-rule single Ridge uses the same validation search without a cost term. Cost prediction saves more tokens on several panels, sometimes at lower test success.

4.3.3 RQ3: WHAT MAKES TRANSFER WORK?

Figure 4 varies support-pool size n, neighborhood size k, and execution repetitions r across three benchmarks and three models. The rows probe coverage, aggregation, and execution noise in Equation (5), using held-out gain MAE.

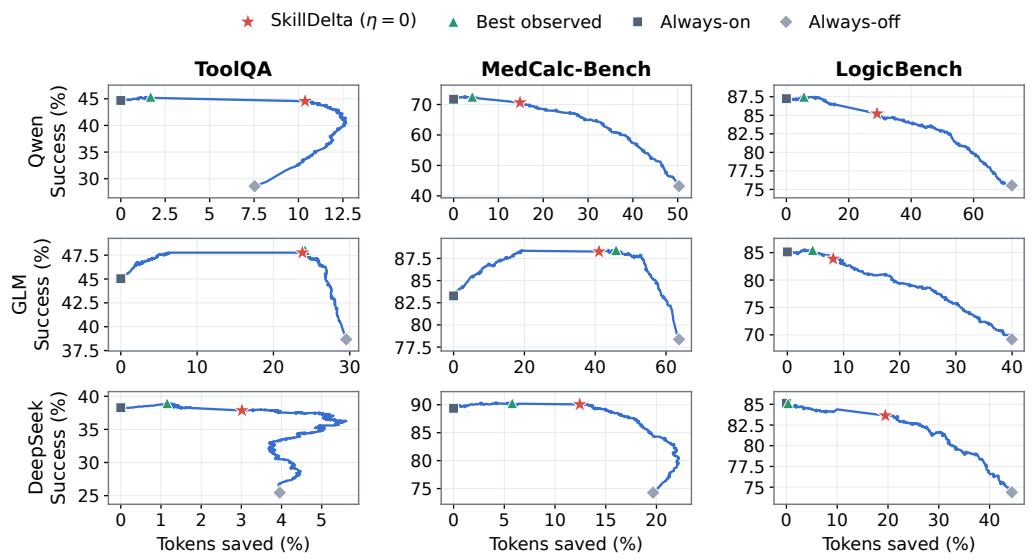

Figure 3: Success–token trade-offs as the decision threshold changes. Fixed within-family, cosine-weighted $k = 6$ scores with each target excluded from support. Stars mark the default ($\eta = 0$); triangles mark the highest success found retrospectively in each sweep.

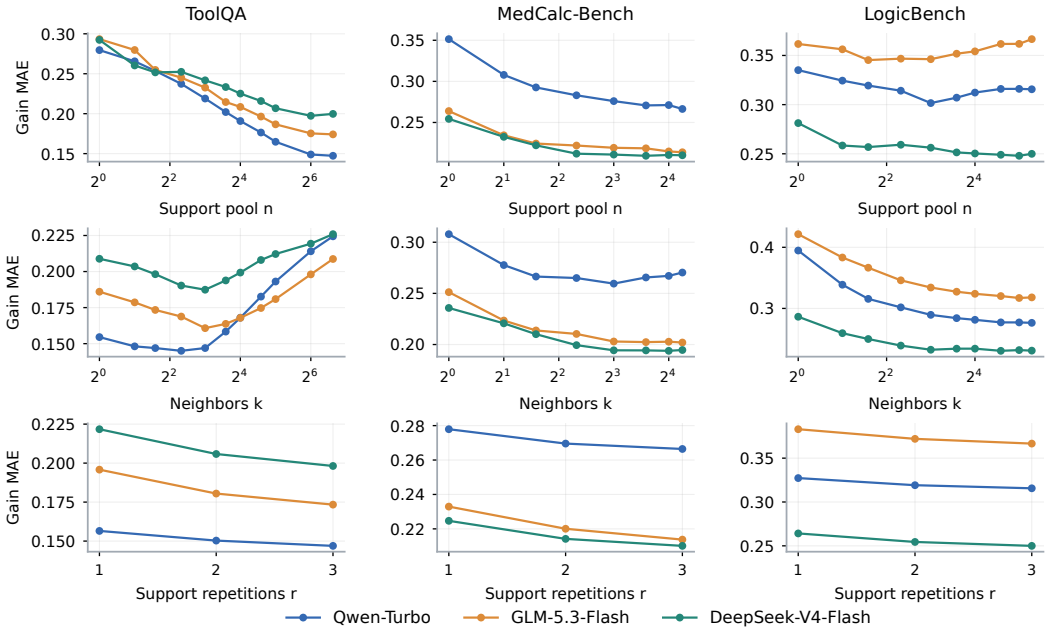

Figure 4: Support, neighborhood size, and repetition affect prediction differently. Gain MAE (lower is better) against fixed targets while varying support-pool size (top), neighbors (middle), or execution repetitions (bottom). Support helps unevenly; larger neighborhoods trade locality for pooling; repetitions consistently reduce error.

Support size: benefits depend on the benchmark. Larger within-family pools generally help ToolQA and MedCalc-Bench; LogicBench varies across agents and need not improve monotonically. Additional candidates can provide closer neighbors and reduce the coverage term $L\rho_w$. Their value depends on whether they improve the selected neighborhood and preserve relevant skill-effect structure.

Neighborhood size: pooling trades noise against locality. ToolQA favors intermediate neighborhoods; MedCalc-Bench and LogicBench show agent-dependent minima or gains from broader pooling. This matches the aggregation trade-off: adding neighbors reduces execution noise through larger effective support, but can increase ρ_w by including tasks with different effects.

Repeated executions: smaller errors with diminishing returns. With neighborhoods fixed, increasing repetitions reduces MAE in all nine panels, with smaller improvements at each step. This is consistent with the $r^{-1/2}$ noise term: repetition improves support-gain precision while leaving coverage and representation unchanged.

5 Related Work

Skills, retrieval, and adaptive augmentation. Skill research also studies testing and coverage (Wang et al., 2026; Tan et al., 2026). Tool retrieval (Shi et al., 2025) and adaptive augmentation address which external resources to retrieve and when to use them (Schick et al., 2023; Asai et al., 2024; Jeong et al., 2024).

Instance-wise performance and heterogeneous effects. Algorithm selection and meta-learning predict task-specific performance (Rice, 1976; Xu et al., 2008; Mısır & Sebag, 2017; Achille et al., 2019); performance prediction infers unobserved model–task scores (Zhao et al., 2025), while sample-efficient evaluation estimates benchmark performance from fewer examples (Maia Polo et al., 2024; Wu et al., 2026). Heterogeneous-effect estimation and policy learning connect incremental outcomes to decisions (Shalit et al., 2017; Nie & Wager, 2021; Athey et al., 2019; Diemert et al., 2021; Zhou et al., 2023). Two-response estimators predict the mean outcome under each condition and take the difference (Künzel et al., 2019); shared local weights recover our signed-gain estimator. Agent tasks can be repeated under both conditions, allowing us to study how precisely these differences can be measured and transferred to new tasks.

Transfer and selective decisions. Domain adaptation studies coverage and conditional shift (Peyré & Cuturi, 2019; Courty et al., 2017; Redko et al., 2017; Koc et al., 2025), while selective and conformal prediction formalize uncertainty (El-Yaniv & Wiener, 2010; Lei et al., 2018). We connect the coverage and precision of paired history to skill-use decisions.

6 Discussion and Limitations

What limits prediction? Representation matters: TF–IDF yields higher gain-sign AUROC on three of the five Qwen-Turbo benchmarks, with BigCodeBench rising from 0.527 to 0.582 under the same paired history (Table 4). Semantic similarity alone need not preserve similarity in skill benefit. BigCodeBench's sparse groups and heterogeneous skill bundles may also constrain local evidence (Appendix C). Its natural-shift panel further shows a success gain over Always-off while task-level ranking weakens (Appendix D).

Deployment cost. A practical use case is repeated task execution under stable agent and skill conditions, where paired evaluation records can be reused for incoming tasks. Reported token savings cover execution and exclude bank construction, target encoding, and retrieval. Collecting a bank from scratch adds an investment whose value depends on reuse; agent or skill changes require rechecking its evidence. This makes history availability and reuse central to deployment.

7 Conclusion

SkillDelta predicts whether a supplied skill improves a fixed agent's chance of success before execution. Our analysis explains when paired history supports this transfer. Across five benchmarks and three agents, the resulting selection improves success over random activation at the same expected skill-use rate. The gains arise mainly from allocation across task groups, with the clearest additional within-group value on ToolQA. Paired history can therefore guide which kinds of tasks receive a skill, even where finer task selection is weak.

AI Use Statement

The authors led the research, with AI tools providing assistance in writing, coding, and analysis. The authors revised and checked the content and take full responsibility for the final paper.

Ethics Statement

This work uses existing agent benchmarks and collects no new human-subject data. Experiments invoke hosted language-model services; model identities, token usage, failures, and costs are recorded for audit, while credentials and private service metadata are excluded from released artifacts. Applications involving safety-critical instructions require a separate safety assessment beyond the task-success and token metrics studied here. Deployment trade-offs are discussed in Section 6.

Reproducibility Statement

The anonymous reproduction package reconstructs main predictions and policy metrics for all three agents from task-level outcomes and embeddings. It includes matched-rate bootstrap code and integrity checks. The retained repeated outcomes also support direct recomputation of RQ3 curves. The supplementary code recomputes threshold sweeps, selection-value components, and the Ridge controls. Proofs appear in the appendix.

References

Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless Fowlkes, Stefano Soatto, and Pietro Perona. Task2Vec: Task embedding for meta-learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6430–6439, 2019.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *International Conference on Learning Representations*, 2024.

Susan Athey, Julie Tibshirani, and Stefan Wager. Generalized random forests. *The Annals of Statistics*, 47(2):1148–1178, 2019. doi: 10.1214/18-AOS1709.

Nicolas Courty, Rémi Flamary, Devis Tuia, and Alain Rakotomamonjy. Optimal transport for domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(9):1853–1865, 2017.

Eustache Diemert, Artem Betlei, Christophe Renaudin, Massih-Reza Amini, Théophane Gregoir, and Thibaud Rahier. A large scale benchmark for individual treatment effect prediction and uplift modeling. *arXiv preprint arXiv:2111.10106*, 2021.

Ran El-Yaniv and Yair Wiener. On the foundations of noise-free selective classification. *Journal of Machine Learning Research*, 11(53):1605–1641, 2010.

Atty Eleti, Jeff Harris, and Logan Kilpatrick. Function calling and other API updates. OpenAI, 2023.

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

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. doi: 10.1080/01621459.1963.10500830.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C. Park. Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2024.

Okan Koc, Alexander Soen, Chao-Kai Chiang, and Masashi Sugiyama. Domain adaptation and entanglement: An optimal transport perspective. In *Proceedings of the 28th International Conference on Artificial Intelligence and Statistics*, volume 258 of *Proceedings of Machine Learning Research*, pp. 3034–3042, 2025.

Samory Kpotufe. k-NN regression adapts to local intrinsic dimension. In *Advances in Neural Information Processing Systems*, volume 24, pp. 729–737, 2011.

Sören R. Künzel, Jasjeet S. Sekhon, Peter J. Bickel, and Bin Yu. Metalearners for estimating heterogeneous treatment effects using machine learning. *Proceedings of the National Academy of Sciences*, 116(10):4156–4165, 2019. doi: 10.1073/pnas.1804597116.

Jing Lei, Max G'Sell, Alessandro Rinaldo, Ryan J. Tibshirani, and Larry Wasserman. Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, 113(523): 1094–1111, 2018.

Xiangyi Li et al. SkillsBench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*, 2026.

Yujian Liu, Jiabao Ji, Li An, Tommi Jaakkola, Yang Zhang, and Shiyu Chang. How well do agentic skills work in the wild: Benchmarking LLM skill usage in realistic settings. *arXiv preprint arXiv:2604.04323*, 2026.

Zeyao Ma, Bohan Zhang, Jing Zhang, Jifan Yu, Xiaokang Zhang, Xiaohan Zhang, Sijia Luo, Xi Wang, and Jie Tang. SpreadsheetBench: Towards challenging real world spreadsheet manipulation. In *Advances in Neural Information Processing Systems*, volume 37, pp. 94871–94908, 2024. doi: 10.52202/079017-3007.

Felipe Maia Polo, Lucas Weber, Leshem Choshen, Yuekai Sun, Gongjun Xu, and Mikhail Yurochkin. tinyBenchmarks: Evaluating LLMs with fewer examples. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 34303–34326, 2024.

Mustafa Mısır and Michèle Sebag. Alors: An algorithm recommender system. *Artificial Intelligence*, 244:291–314, 2017. doi: 10.1016/j.artint.2016.12.001.

Xinkun Nie and Stefan Wager. Quasi-oracle estimation of heterogeneous treatment effects. *Biometrika*, 108(2):299–319, 2021. doi: 10.1093/biomet/asaa076.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs from preference data. In *International Conference on Learning Representations*, 2025.

Gabriel Peyré and Marco Cuturi. Computational optimal transport. *Foundations and Trends in Machine Learning*, 11(5–6):355–607, 2019.

Ievgen Redko, Amaury Habrard, and Marc Sebban. Theoretical analysis of domain adaptation with optimal transport. In *Machine Learning and Knowledge Discovery in Databases*, volume 10535 of *Lecture Notes in Computer Science*, pp. 737–753. Springer, 2017. doi: 10.1007/978-3-319-71246-8_45.

John R. Rice. The algorithm selection problem. *Advances in Computers*, 15:65–118, 1976.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, pp. 68539–68551, 2023.

Uri Shalit, Fredrik D. Johansson, and David Sontag. Estimating individual treatment effect: Generalization bounds and algorithms. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 3076–3085, 2017.

Zhengliang Shi, Yuhan Wang, Lingyong Yan, Pengjie Ren, Shuaiqiang Wang, Dawei Yin, and Zhaochun Ren. Retrieval models aren't tool-savvy: Benchmarking tool retrieval for large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 24497–24524. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.1258.

Seamus Somerstep, Felipe Maia Polo, Allysson Flavio Melo de Oliveira, Prattyush Mangal, Mírian Silva, Onkar Bhardwaj, Mikhail Yurochkin, and Subha Maity. CARROT: A cost aware rate optimal router. *arXiv preprint arXiv:2502.03261*, 2025.

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

Boyin Tan, Xiaowei Huang, and Youcheng Sun. Skill coverage: A test adequacy metric for agent skills. *arXiv preprint arXiv:2606.20659*, 2026.

Leye Wang, Zixing Wang, and Anjie Xu. SkillTester: Benchmarking utility and security of agent skills. *arXiv preprint arXiv:2603.28815*, 2026.

Skyler Wu, Yash Nair, and Emmanuel J. Candès. Efficient evaluation of LLM performance with statistical guarantees. *arXiv preprint arXiv:2601.20251*, 2026.

Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. SATzilla: Portfolio-based algorithm selection for SAT. *Journal of Artificial Intelligence Research*, 32:565–606, 2008.

Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, Kai Qiu, Yuqing Yang, Dongdong Chen, Xue Yang, and Chong Luo. SkillOpt: Executive strategy for self-evolving agent skills. *arXiv preprint arXiv:2605.23904*, 2026.

Barry Zhang, Keith Lazuka, and Mahesh Murag. Equipping agents for the real world with agent skills. Anthropic Engineering, 2025.

Qinyu Zhao, Ming Xu, Kartik Gupta, Akshay Asthana, Liang Zheng, and Stephen Gould. Can we predict performance of large models across vision-language tasks? In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 77781–77806, 2025.

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.

Zhengyuan Zhou, Susan Athey, and Stefan Wager. Offline multi-action policy learning: Generalization and optimization. *Operations Research*, 71(1):148–183, 2023. doi: 10.1287/opre.2022.2271.

A Proofs and Generalization Guarantees

A.1 Reading guide and notation

The argument proceeds from the evidence needed to identify a gain, through prediction error for one task, to the consequences for skill-use decisions. We then average these results over new tasks. Policy evaluation is a separate step: it asks what recorded executions tell us about a chosen policy's value. Figure 5 summarizes these dependencies.

We use the fixed agent, supplied-skill condition, and notation of Sections 2 and 3. Write $\tau(t)$ for $\tau(t, s)$ and abbreviate the noise factor and error radius in Equation (5) as

$$v_w(t) = \sum_i \frac{w_i(t)^2}{r_i}, \qquad B_\delta(t) = L\rho_w(t) + \varepsilon_\phi + \sqrt{2\log(2/\delta)v_w(t)}. \tag{9}$$

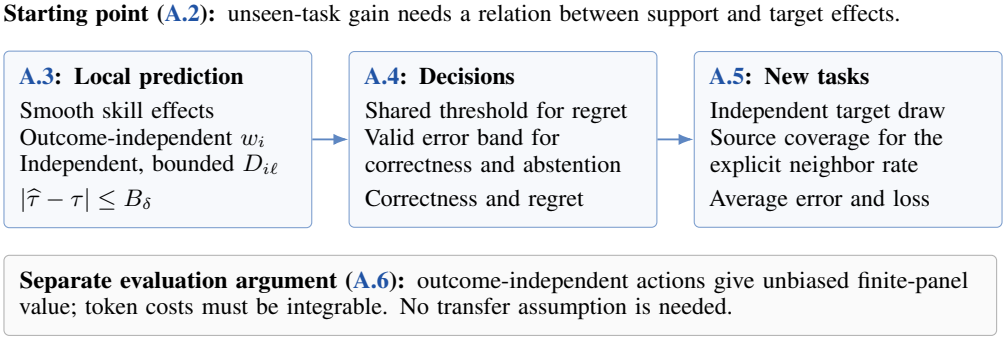

Figure 5: How the theory fits together. The main chain turns historical evidence into prediction and decision guarantees. Policy evaluation uses a separate action–outcome independence argument.

What is random? For a fixed-target bound, task identities, representations, repetition counts, and weights are fixed; probability is over support execution outcomes. The fresh-task result additionally draws a target $T \sim Q$. The neighbor-rate result also draws the support tasks. These are three different sources of uncertainty, introduced only where needed below.

A.2 Identification and observed gains

This first result explains why a transfer assumption is needed: collecting more support outcomes cannot determine an unrelated target's gain.

Proposition 4 (No free lunch for unseen-task gain)**.** *Let $S = \{t_1, \dots, t_n\} \subsetneq \mathcal{T}$ be the executed support tasks and $t_\star \notin S$. Without a restriction relating $\tau(t_\star)$ to $\{\tau(t_i)\}_{i=1}^n$, there exist two data-generating processes that agree on the joint distribution of all observed support executions but satisfy $\tau^+(t_\star) > 0$ and $\tau^-(t_\star) \le 0$. Consequently, no predictor can uniformly identify the target gain sign.*

Proof. Fix the joint support-outcome distribution. Two extensions can both have $\mu_0(t_\star) = 1/2$, with $\mu_1^+(t_\star) = 3/4$ and $\mu_1^-(t_\star) = 1/4$. They give opposite target gains but identical distributions for every observed support variable. A support-based predictor therefore has the same distribution in both worlds and cannot be correct uniformly. □

Observed signs and expected gains. If the two conditions independently succeed with probability $1/2$, then $\tau(t) = 0$, yet one paired execution gives $\Pr(\widehat{D} = 1) = \Pr(\widehat{D} = -1) = 1/4$. Thus the observed label $\mathbf{1}\{\widehat{D} > 0\}$ used for RQ1 AUROC differs from the expected gain estimated by $\widehat{\tau}$.

A.3 Local gain-prediction bound

We prove Theorem 1 by separating transfer mismatch from execution noise. Condition on a nonempty eligible support set, representations, repetition counts, and outcome-independent weights throughout

this proof. The bound includes the cosine-weighted predictor. With empty support, the algorithm's zero score is a fallback, not an estimated zero gain.

Proof of Theorem 1. Step 1: separate the two errors. Add and subtract the weighted expected gains:

$$\widehat{\tau}(t) - \tau(t) = \underbrace{\sum_i w_i(t)\big(\widehat{D}_i - \tau(t_i)\big)}_{Z_w(t)\text{: execution noise}} + \underbrace{\sum_i w_i(t)\big(\tau(t_i) - \tau(t)\big)}_{b_w(t)\text{: transfer mismatch}}. \tag{10}$$

Step 2: bound transfer mismatch. By Equation (4), nonnegative weights, and $\sum_i w_i = 1$,

$$|b_w(t)| \le \sum_i w_i(t)|\tau(t_i) - \tau(t)| \le L\rho_w(t) + \varepsilon_\phi.$$

Step 3: bound execution noise. Write $Z_w(t)$ as a sum of independent centered variables $\sum_i \sum_{\ell=1}^{r_i}(w_i/r_i)(D_{i\ell} - \mathbb{E}D_{i\ell})$. Each uncentered summand lies in an interval of length $2w_i/r_i$. Hoeffding's inequality (Hoeffding, 1963) therefore gives

$$\Pr\left(|Z_w(t)| \ge x\right) \le 2\exp\left(-\frac{x^2}{2v_w(t)}\right). \tag{11}$$

Setting the right-hand side to δ gives $|Z_w(t)| \le \sqrt{2\log(2/\delta)v_w(t)}$ with probability at least $1-\delta$. Adding the two bounds proves the result. □

Neighborhood size and family averaging. With k equally weighted neighbors and r repetitions, $v_w = 1/(kr)$. Increasing k reduces noise but can enlarge ρ_w; family averaging is useful when effects vary little within the family. At equal repetition counts, uniform weights minimize the noise term on a fixed support set, while the total bound also depends on transfer bias.

The number of neighbors can overstate the amount of averaging when a few weights dominate. Effective support makes this distinction explicit:

$$n_{\text{eff}}(t) = \left(\sum_i w_i(t)^2\right)^{-1}, \qquad v_w(t) = \frac{1}{r\, n_{\text{eff}}(t)} \quad \text{when all } r_i = r. \tag{12}$$

Substitution into Equation (9) shows how diffuse weights and repetitions reduce the noise term.

Numerical strength at small neighborhoods. For at most k nonzero weights and equal repetition count r, Cauchy–Schwarz gives $v_w \ge 1/(kr)$. At $\delta = 0.05$ and the main setting $k = 6$, the minimum noise terms are approximately 1.109 for $r = 1$ and 0.640 for $r = 3$. Since $|\tau| \le 1$, the first case cannot certify a zero-threshold decision. Here the bound explains the trade-off; numerical certification needs more evidence or tighter uncertainty estimates.

A.4 From gain error to decisions

The following results translate prediction error into three decision questions: when the binary action is correct, how much a wrong action costs, and when an uncertainty band supports a definite action. Regret control is algebraic; correctness statements use the event that the error band is valid.

A.4.1 Threshold correctness

Proof of Corollary 2. On the event $|\widehat{\tau}(t) - \tau(t)| \le B(t)$, suppose $|\tau(t) - \eta| > B(t)$. If $\tau(t) > \eta$, then $\widehat{\tau}(t) \ge \tau(t) - B(t) > \eta$; if $\tau(t) < \eta$, then $\widehat{\tau}(t) \le \tau(t) + B(t) < \eta$. Thus the decisions agree. Taking the contrapositive, disagreement on the same event implies $|\tau(t) - \eta| \le B(t)$. □

A.4.2 Routing regret

Proof of Proposition 3. If $\widehat{a}_\eta(t) \ne a_\eta(t)$, then η lies between $\widehat{\tau}(t)$ and $\tau(t)$. Hence

$$|\tau(t) - \eta| \le |\widehat{\tau}(t) - \tau(t)|.$$

Multiplying by the disagreement indicator proves Equation (8); if the actions agree, the left-hand side is zero. On the event in Theorem 1, regret is at most $B_\delta(t)$. □

A.4.3 Uncertainty bands and selective abstention

To allow abstention, let $I_\delta(t) = [\widehat{\tau}(t) - B_\delta(t), \widehat{\tau}(t) + B_\delta(t)]$ and define a three-way gate that uses the skill when the lower endpoint exceeds η, does not use it when the upper endpoint is at most η, and abstains otherwise.

Corollary 5 (Selective correctness from a valid error band)**.** *On the event $|\widehat{\tau}(t) - \tau(t)| \le B_\delta(t)$, every non-abstaining action agrees with the threshold oracle. Moreover, abstention implies $|\tau(t) - \eta| \le 2B_\delta(t)$. The conservative binary rule*

$$\widehat{a}_{\mathrm{safe}}(t) = \mathbf{1}\{\widehat{\tau}(t) - B_\delta(t) > \eta\}, \tag{13}$$

never invokes the skill when $\tau(t) \le \eta$ on this event, and its threshold-relative regret is at most $2B_\delta(t)$.

Proof. If the lower endpoint exceeds η, then $\tau(t) \ge \widehat{\tau}(t) - B_\delta(t) > \eta$; if the upper endpoint is at most η, then $\tau(t) \le \widehat{\tau}(t) + B_\delta(t) \le \eta$. If the band crosses the threshold and $\tau(t) > \eta$, then $\widehat{\tau}(t) - B_\delta(t) \le \eta$ and hence $\tau(t) \le \widehat{\tau}(t) + B_\delta(t) \le \eta + 2B_\delta(t)$. The case $\tau(t) \le \eta$ is symmetric. Finally, the conservative rule can err only by skipping when $\tau(t) > \eta$; the preceding margin argument bounds the resulting regret by $2B_\delta(t)$. □

A.5 Fresh-task generalization

The local bound concerns one fixed task. This section asks how often it fails, and how much decision loss remains, when the target is drawn from a deployment distribution. The first theorem averages over targets; the neighbor-rate corollary additionally quantifies coverage from a sampled support bank.

Condition on support-task identities, representations, and repetition counts. Let $T \sim Q$ be independent of support outcomes, with weights $w_i(T)$ measurable from the support features and T. The independent execution assumptions of Theorem 1 continue to hold. For a possibly task-dependent threshold, write $\widehat{a}_\eta(t) = \mathbf{1}\{\widehat{\tau}(t) > \eta(t)\}$ and $a_\eta(t) = \mathbf{1}\{\tau(t) > \eta(t)\}$.

Theorem 6 (Fresh-task gain generalization)**.** *Under Equation (4), for any $\delta \in (0, 1)$,*

$$\Pr_{T,\mathrm{exec}}\left(|\widehat{\tau}(T) - \tau(T)| > B_\delta(T)\right) \le \delta. \tag{14}$$

For any threshold $\eta(t)$ fixed independently of the support execution outcomes, the plug-in and oracle actions satisfy

$$\begin{aligned}\Pr_{T,\mathrm{exec}}\left(\widehat{a}_\eta(T) \ne a_\eta(T)\right) \\ \le \delta + \Pr_{T\sim Q}\left(|\tau(T) - \eta(T)| \le B_\delta(T)\right).\end{aligned} \tag{15}$$

For every $b > 0$, the last term is at most

$$\Pr_T(B_\delta(T) > b) + \Pr_T(|\tau(T) - \eta(T)| \le b). \tag{16}$$

The mean gain error also bounds mean regret:

$$\begin{aligned}\mathbb{E}_{T,\mathrm{exec}} R_\eta(\widehat{a}, T) &\le \mathbb{E}_{T,\mathrm{exec}}|\widehat{\tau}(T) - \tau(T)| \\ &\le L\mathbb{E}_T\rho_w(T) + \varepsilon_\phi + \mathbb{E}_T\sqrt{v_w(T)}.\end{aligned} \tag{17}$$

Proof. Prediction error. For every fixed target, Theorem 1 bounds the execution failure probability by δ. Averaging these conditional probabilities over T proves Equation (14).

Decision error. By Corollary 2, disagreement on the bound's event requires $|\tau(T) - \eta(T)| \le B_\delta(T)$, proving Equation (15). This margin event is contained in

$$\{B_\delta(T) > b\} \cup \{|\tau(T) - \eta(T)| \le b\},$$

which gives Equation (16).

Mean regret. In Equation (10), independence and $D_{i\ell} \in [-1, 1]$ give $\mathrm{Var}(Z_w(t)) \le v_w(t)$. Jensen's inequality bounds its expected absolute value by $\sqrt{v_w(t)}$. Adding the transfer-mismatch bound, averaging over T, and applying Equation (8) proves Equation (17). □

Margin consequence. If $B_\delta(T) \le b$ almost surely and $\Pr_T(|\tau(T) - \eta(T)| \le u) \le Cu^\alpha$ for $u \in [0, b]$, with $C, \alpha > 0$, then Equation (15) immediately gives $\Pr_{T,\mathrm{exec}}(\widehat{a}_\eta(T) \neq a_\eta(T)) \le \delta + Cb^\alpha$. This requires few targets near the decision threshold.

Turning support size into a coverage bound. As in local k-NN regression analysis (Kpotufe, 2011), the next condition requires source probability mass near every possible target representation. It allows different source and target distributions, while ruling out targets with no nearby support. This corollary specializes the weighted bound to uniform neighbor averaging.

Corollary 7 (k-neighbor generalization rate)**.** *Under the assumptions of Theorem 1, let the support representations $X_i = \phi(t_i)$ be n independent draws from a source distribution P, and let $T \sim Q$ be independent. Assume the representation space has diameter at most one and that, for every z in the support of the target representation distribution and every $u \in [0, 1]$,*

$$\Pr_{X \sim P}(d(X, z) \le u) \ge cu^\nu \tag{18}$$

for constants $c \in (0, 1]$ and a local dimension exponent $\nu > 0$. Give the k nearest support tasks equal weight and execute each support task for r independent repetitions. If $u_{n,k} = (2k/(cn))^{1/\nu} \le 1$, then, with probability at least $1 - \delta - e^{-k/4}$ over the support draw, fresh target, and executions,

$$|\widehat{\tau}(T) - \tau(T)| \le L\left(\frac{2k}{cn}\right)^{1/\nu} + \varepsilon_\phi + \sqrt{\frac{2\log(2/\delta)}{kr}}. \tag{19}$$

Proof. Coverage. Fix a target representation z and let $M_z(u)$ count support points in its radius-u ball. At $u = u_{n,k}$, Equation (18) gives $\mathbb{E}[M_z(u)] \ge 2k$. A multiplicative Chernoff bound therefore yields $\Pr(M_z(u) < k) \le e^{-k/4}$. After integrating over the fresh target, the kth-neighbor radius is at most $u_{n,k}$ with the same probability. On this event the uniform-weight radius $\rho_w(T)$ is at most $u_{n,k}$, while $v_w(T) = 1/(kr)$.

Gain error. Apply Theorem 1 conditionally on the support and target. A union bound combines the coverage-failure probability $e^{-k/4}$ with the execution-failure probability δ. □

For fixed δ, balancing $(k/n)^{1/\nu}$ and $(kr)^{-1/2}$ gives order $(nr)^{-1/(\nu+2)}$ for the coverage and noise terms in the error radius when the optimizing k is feasible. Representation error remains. The rate requires source mass near target tasks as in Equation (18); it does not follow from a within-inventory audit alone.

A.6 Evaluating frozen and cross-fitted policies

Action–outcome independence gives an unbiased estimate of policy success on a fixed set of target tasks. This calculation uses no smoothness or coverage assumptions.

Proposition 8 (Frozen finite-panel value)**.** *Condition on target tasks $t_1, \ldots, t_m$ and on a policy $\widehat{\pi}$ fixed independently of their execution outcomes. Then*

$$\widehat{V}(\widehat{\pi}) = \frac{1}{m}\sum_{j=1}^{m}\left[\widehat{\pi}(t_j)Y_{j1} + (1 - \widehat{\pi}(t_j))Y_{j0}\right] \tag{20}$$

is unbiased for the policy's mean success on that finite target inventory. Here Y_{ja} is the observed success, or its repeat mean, in condition a. The analogous estimator for tokens in the selected condition is unbiased under the same independence condition and integrable token costs.

Proof. Conditioning on tasks and frozen actions $a_j = \widehat{\pi}(t_j)$, independence gives $\mathbb{E}[Y_{ja} \mid t_{1:m}, a_{1:m}] = \mu_a(t_j)$ for $a \in \{0, 1\}$. Substituting into Equation (20) and applying linearity proves the claim. The same argument applies to any integrable condition-specific cost. □

Leave-one-task-out evaluation. Suppose the full predictor configuration is fixed independently of evaluation outcomes, and task executions are independent conditional on task identities. An action a_j fitted using only other tasks' outcomes is then independent of (Y_{j0}, Y_{j1}), giving the marginal identity

$$\mathbb{E}[a_j Y_{j1} + (1 - a_j)Y_{j0} \mid t_{1:m}] = \mathbb{E}[a_j \mid t_{1:m}]\mu_1(t_j) + (1 - \mathbb{E}[a_j \mid t_{1:m}])\mu_0(t_j).$$

Summing gives the expected value of the fitted rules. Because actions share training outcomes, this marginal identity differs from conditioning on the entire action vector in Proposition 8. Configuration selection still requires separate validation. Fixed-prediction bootstrap summaries measure resampling variability of the fitted results; full-procedure uncertainty also includes refitting and configuration selection.

B Empirical Protocols and Supplementary Results

The main results use within-inventory, leave-one-task-out $k = 6$ prediction. RQ3 varies the evidence against fixed three-repeat targets; Ridge uses grouped train/validation/test splits. The prospective panels in Appendix D retain their historical frozen configurations.

B.1 Execution records and evaluation inventory

Section 4 specifies the benchmarks, execution counts, encoders, and main policies. Each agent has separate outcome and token ledgers. Transport failures and incomplete attempts are excluded; completed empty outputs remain failures under the benchmark policy. The retained evidence comprises paired responses, predictions, embeddings, evaluator outputs, protocol manifests, and integrity hashes.

The Qwen-Turbo and GLM-5.3-Flash main inventories each contain 4,825 evaluable tasks and 12,690 condition records. They share 1,536-dimensional question vectors and the offline gate definition, with stack-specific execution limits recorded separately, including GLM's ToolQA step cap and low-effort LogicBench budget. DeepSeek contributes 28,950 condition records over the same 4,825 tasks and uses the shared 1,536-dimensional vectors for the main analysis. Its separate Ridge controls retain their archived 1,024-dimensional `Qwen3-Embedding-0.6B` vectors. Its SpreadsheetBench evaluation retains original and corrected flags for 18 corrected records and excludes task 42930 because its test case is absent. Task–skill incidence and bundle definitions appear in Appendix C.

B.2 Prediction and baseline controls

Score support and thresholds. All main results use same-family, cosine-weighted $k = 6$ neighbors and the zero-threshold signed-gain rule. Negative cosine weights are clipped to zero; if all selected weights vanish, the neighbors receive equal weights. Empty support yields score zero and Skip skill. Each target is excluded from its support. The with-skill-only and paired scores share exactly the same neighbors and weights, averaging skill-enabled outcomes or paired gains, respectively. Repetition counts are specified in Section 4.

Configuration provenance. The common $k = 6$ rule harmonizes the earlier benchmark-specific analyses retrospectively, after trajectory collection. Leave-one-task-out exclusion prevents direct use of a target's outcomes for its score; configuration selection is outside this evaluation. Keeping the same protocol with $k = 3$ yields 61.83% mean success, a 4.63% matched-rate advantage, and 26.87% execution-token savings; $k = 6$ yields 63.12%, 4.33%, and 20.83%, respectively. These averages weight all 15 panels equally. Both rules have positive matched-rate point estimates throughout: $k = 6$ has higher mean success, whereas $k = 3$ saves more tokens.

Table 3 collects the outcome-free relevance and self-judge diagnostics that complement the main prediction and policy results. Table 4 compares estimators given the same paired history on Qwen-Turbo.

Table 3: No-history baseline controls. BA gain is the absolute balanced-accuracy advantage of SKILLDELTA over self-judge.

	Relevance gate		Self-judge		
Benchmark	Success (%)	Saved tokens (%)	Use (%)	Router tokens/task	BA gain (%) [95% CI]
Qwen-Turbo					
ToolQA	36.99	3.07	94.2	1164	+30.41 [27.99, 32.75]
MedCalc-Bench	60.00	25.31	98.1	1583	+19.65 [17.11, 22.21]
BigCodeBench	50.18	46.76	99.5	4976	+1.77 [-2.49, 6.09]
LogicBench	83.03	35.69	98.3	1314	+8.55 [4.78, 12.27]
SpreadsheetBench	18.30	35.53	93.7	2955	+8.01 [-2.35, 18.68]
GLM-5.3-Flash					
ToolQA	40.84	11.17	99.4	1322	+29.86 [26.69, 32.87]
MedCalc-Bench	83.55	37.92	99.8	1691	+27.19 [23.76, 30.48]
BigCodeBench	61.71	25.99	98.9	5231	+6.63 [2.60, 10.55]
LogicBench	74.25	21.49	96.3	2224	+6.60 [3.14, 10.01]
SpreadsheetBench	57.89	16.51	91.2	3264	+16.19 [11.23, 21.07]
DeepSeek-V4-Flash					
ToolQA	30.96	-0.66	96.4	1396	+21.54 [19.11, 23.99]
MedCalc-Bench	81.48	10.80	97.8	1680	+26.75 [24.08, 29.40]
BigCodeBench	59.54	39.02	96.0	5514	-0.03 [-3.36, 3.21]
LogicBench	80.61	24.88	91.3	1821	+11.50 [7.02, 15.95]
SpreadsheetBench	51.38	20.43	44.4	3346	+13.49 [6.57, 20.36]

The relevance gate uses task–skill TF–IDF cosine and an inventory-median threshold. DeepSeek retains its published relevance scores. BA intervals use 10,000 paired task resamples with decisions fixed. Self-judge success and total token costs appear in Table 2.

Table 4: Paired-evidence estimator controls on the complete Qwen-Turbo inventories.

Benchmark	Estimator	AUROC	Success (%)	Tokens saved (%)
ToolQA	SKILLDELTA	0.879	44.55	10.37
	TF–IDF kNN	0.877	44.27	10.28
	Family mean	0.786	45.38	7.55
MedCalc-Bench	SKILLDELTA	0.781	70.64	14.82
	TF–IDF kNN	0.789	70.55	13.33
	Family mean	0.785	71.55	7.71
BigCodeBench	SKILLDELTA	0.527	47.36	49.41
	TF–IDF kNN	0.582	48.94	48.11
	Family mean	0.474	49.56	19.25
LogicBench	SKILLDELTA	0.621	85.22	29.05
	TF–IDF kNN	0.546	85.09	27.67
	Family mean	0.622	87.46	16.02
SpreadsheetBench	SKILLDELTA	0.580	17.79	36.16
	TF–IDF kNN	0.582	17.04	40.33
	Family mean	0.322	19.80	0.00

All methods use the same paired cells and zero threshold. TF–IDF kNN replaces the representation while retaining same-family, nonnegative-cosine $k = 6$ aggregation; its word unigram/bigram vocabulary uses only inventory text. Family mean averages all other tasks in the group. Savings are relative to Always-on.

B.3 POLICY VALUE AND UNCERTAINTY

Table 5 reports uncertainty for the main metrics; Figure 6 compares matched-rate selection value across agents and benchmarks. Predictions and actions are held fixed during resampling; Appendix A.6 distinguishes this analysis from uncertainty for the full fitting procedure.

Table 5: Task-bootstrap intervals for the main results. Gains are absolute success-rate differences from Always-off. Point estimates appear in Figure 2 and Table 2. Scores and actions are fixed across 10,000 task resamples.

Stack	Benchmark	AUROC: 95% CI	Success gain (%): 95% CI
Qwen-Turbo	ToolQA	[0.849, 0.907]	[13.9, 18.0]
Qwen-Turbo	MedCalc-Bench	[0.751, 0.810]	[24.5, 30.4]
Qwen-Turbo	BigCodeBench	[0.478, 0.577]	[1.8, 5.5]
Qwen-Turbo	LogicBench	[0.574, 0.667]	[7.2, 12.2]
Qwen-Turbo	SpreadsheetBench	[0.478, 0.681]	[-0.8, 5.5]
GLM-5.3-Flash	ToolQA	[0.792, 0.875]	[7.5, 10.7]
GLM-5.3-Flash	MedCalc-Bench	[0.819, 0.891]	[7.9, 11.9]
GLM-5.3-Flash	BigCodeBench	[0.520, 0.616]	[5.9, 9.7]
GLM-5.3-Flash	LogicBench	[0.550, 0.634]	[12.0, 17.4]
GLM-5.3-Flash	SpreadsheetBench	[0.638, 0.745]	[21.6, 31.1]
DeepSeek-V4-Flash	ToolQA	[0.751, 0.811]	[10.8, 14.1]
DeepSeek-V4-Flash	MedCalc-Bench	[0.793, 0.852]	[13.8, 17.8]
DeepSeek-V4-Flash	BigCodeBench	[0.526, 0.604]	[6.8, 10.1]
DeepSeek-V4-Flash	LogicBench	[0.599, 0.693]	[7.1, 11.4]
DeepSeek-V4-Flash	SpreadsheetBench	[0.607, 0.721]	[10.2, 16.5]

The gain interval includes zero for Qwen-Turbo SpreadsheetBench. Percentile intervals condition on the fixed scores and actions; they exclude configuration-selection uncertainty.

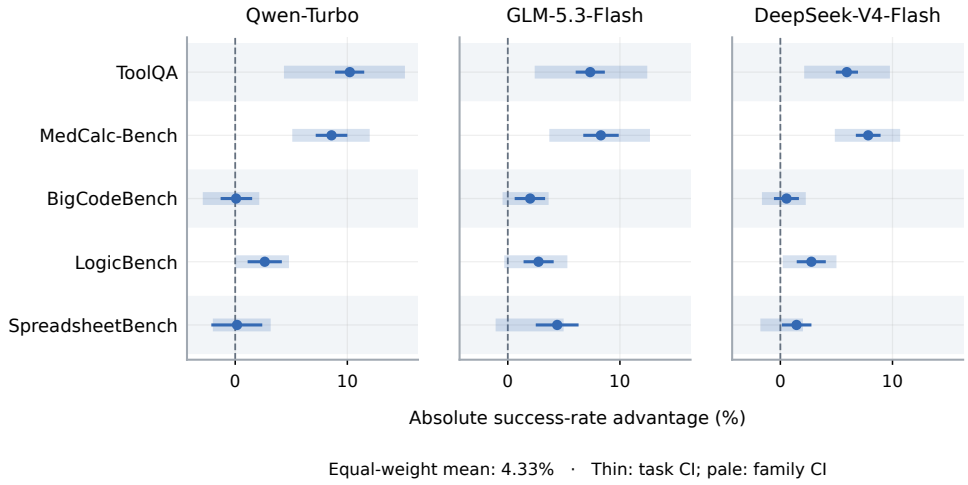

Figure 6: Selection value at the same expected skill-use rate. Points show the absolute success-rate advantage over random activation. Bars are 95% intervals from 5,000 task or whole-family resamples, keeping actions fixed and recomputing the matching rate. All estimates are positive; some intervals include zero.

Within-group selection and between-group allocation. For a fixed panel, let a_i be its retained action, p its mean use rate, and $p_{g(i)}$ the mean rate in task i's recorded group. The advantage over global matched-rate random activation decomposes exactly as

$$\frac{1}{N}\sum_i (a_i - p)\widehat{D}_i = \underbrace{\frac{1}{N}\sum_i (a_i - p_{g(i)})\widehat{D}_i}_{\text{within-group selection}} + \underbrace{\frac{1}{N}\sum_i (p_{g(i)} - p)\widehat{D}_i}_{\text{between-group allocation}}. \tag{21}$$

The first term compares against random activation at each group's use rate; the second measures the value of allocating more use to groups with larger average gains. Groups follow the recorded task structure: skill families for ToolQA, MedCalc-Bench, and LogicBench, focal groups for BigCodeBench, and task types for SpreadsheetBench. They need not identify identical skill bundles.

Figure 7 keeps every main-table action and outcome unchanged. Across the 15 panels, the equal-weight within-group mean is 0.61% [0.37, 0.93], and the between-group mean is 3.72% [3.47, 3.91], summing to the 4.33% advantage. These 95% intervals use 5,000 bootstrap draws of tasks within groups, preserving group sizes and sharing task draws across agents within each benchmark. Matching rates are recomputed in each draw. The intervals condition on the observed groups and fixed actions; they exclude predictor refitting and new execution noise.

ToolQA provides the strongest evidence for within-group selection: its three-agent mean is 4.41%, and every agent's interval excludes zero. Only five of the 15 within-group point estimates are positive. BigCodeBench ranges from -1.67% to -0.25%; the Qwen-Turbo interval is entirely negative, while the other two include zero. All remaining within-group intervals include zero. The positive aggregate advantage thus combines reliable local selection on ToolQA with substantial allocation value across groups.

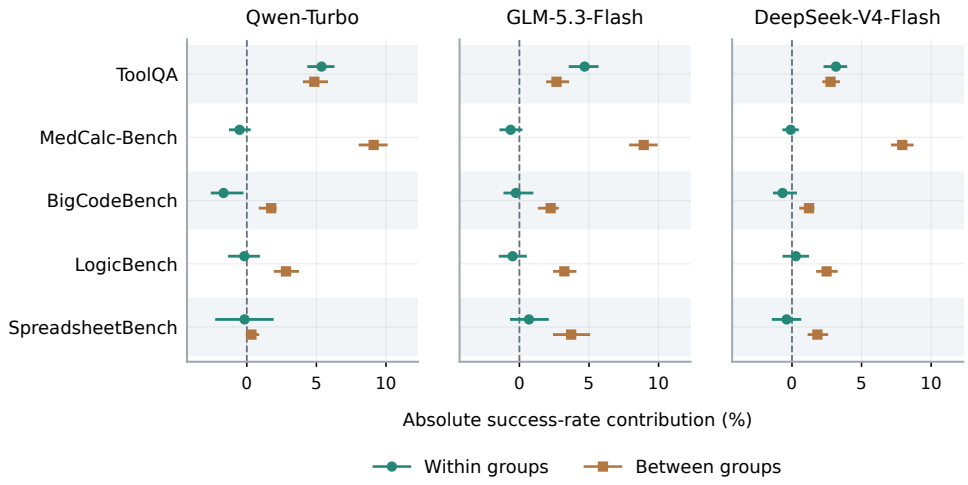

Figure 7: Where selection value comes from. Within-group selection and between-group allocation sum to the advantage over global matched-rate random activation. Bars show conditional 95% intervals from task resampling within the recorded groups.

B.4 Single- and dual-Ridge policy controls

These controls use identical grouped train/validation/test partitions across the three agents, with a separate outcome ledger for each. Both Ridge variants share a signed-gain head; Dual Ridge adds an independent incremental-token head. Five-fold grouped cross-validation selects the Ridge penalties on training data. Validation selects the least-token policy with success no lower than Always-on; matched-rule single Ridge uses the same search with $\lambda = 0$. Test outcomes enter only the final evaluation.

The cost head yields additional savings in several panels, with varying test success. On DeepSeek SpreadsheetBench, matched-rule single and Dual Ridge select identical test actions, so their saving comes from the shared policy search. The intervals in Table 6 describe the fixed test policies and exclude refitting and policy-selection uncertainty. The supplementary reproduction files include the fixed splits, fitted coefficients, and per-task predictions. An independent offline refit reproduces the selected penalties and reported policy metrics on all 15 panels.

B.5 RQ3 sweep settings

All sweeps in Figure 4 use fixed three-repeat target gains, shared question vectors, same-family support, nonnegative cosine weights, and signed-gain aggregation. Nested support pools are averaged over five orderings, with $k = \min(n, 3)$. The pool and neighborhood sweeps stop at 99, 19, and 39 supports for ToolQA, MedCalc-Bench, and LogicBench, respectively, retaining every target at every

Table 6: Ridge policy controls on identical fixed test tasks across three agents. Acc. and Save are percentages; saving is relative to Always-on. Δ is the absolute accuracy difference. Single Ridge uses the same validation search as Dual Ridge with $\lambda = 0$.

			Single Ridge		Dual Ridge		Dual–On
Benchmark	N	On Acc.	Acc.	Save	Acc.	Save	Δ (%) [95% CI]
Qwen-Turbo							
ToolQA	286	44.76	43.71	9.09	43.71	11.27	−1.05 [-3.51,+1.40]
MedCalc-Bench	220	75.91	72.27	19.17	73.64	19.55	−2.27 [-5.07,+0.46]
BigCodeBench	228	50.88	50.88	4.21	50.44	27.90	−0.44 [-3.07,+2.19]
LogicBench	152	89.04	87.94	22.83	87.94	22.83	−1.10 [-4.25,+2.03]
SpreadsheetBench	80	18.75	18.75	0.00	18.75	0.00	+0.00 [+0.00,+0.00]
GLM-5.3-Flash							
ToolQA	286	43.71	46.15	16.26	45.45	21.98	+1.75 [-1.74,+4.91]
MedCalc-Bench	220	87.27	81.82	60.33	78.64	62.44	−8.64 [-14.55,-2.70]
BigCodeBench	228	63.60	63.60	0.00	63.60	1.87	+0.00 [-1.32,+1.32]
LogicBench	152	85.75	85.09	6.24	84.43	5.08	−1.32 [-2.83,-0.22]
SpreadsheetBench	80	67.50	66.25	4.04	63.75	10.24	−3.75 [-8.75,+1.25]
DeepSeek-V4-Flash							
ToolQA	286	35.55	35.31	0.75	35.78	4.77	+0.23 [-0.94,+1.52]
MedCalc-Bench	220	90.76	92.42	8.58	92.12	10.11	+1.36 [-0.61,+3.67]
BigCodeBench	228	59.50	59.50	3.87	59.36	18.66	−0.15 [-1.61,+1.17]
LogicBench	152	87.72	86.62	6.48	87.28	12.29	−0.44 [-2.37,+1.40]
SpreadsheetBench	80	57.08	55.42	16.80	55.42	16.80	−1.67 [-5.42,+2.92]

All stacks reuse the archived task-group partitions and five training folds. Both Ridge heads include an unpenalized intercept; penalties minimize training OOF MSE. Single Ridge and Dual Ridge minimize validation tokens subject to success no lower than Always-on, with explicit Always-on fallback. Dual Ridge predicts incremental cost in thousands of tokens and uses $\widehat{g} - \lambda\widehat{c} > \eta$. The 95% intervals use 5,000 task-group bootstrap draws with the policy fixed. Qwen and GLM use their canonical outcomes and shared 1,536-dimensional question vectors; DeepSeek uses three-repeat means and its archived 1,024-dimensional vectors.

point. The repetition sweep fixes $k = 3$ and independently samples one to three executions without replacement from each support task's two arms, using 50 draws when fewer than three are retained. These three-repeat diagnostic targets are distinct from the single-execution Qwen/GLM ToolQA and MedCalc headline outcomes. When only three-repeat success counts are retained, arm-wise subsampling uses the corresponding binary multiset; it does not assume a chronological pairing of repetitions.

B.6 Supplementary archives and reproduction

The reproduction guide provides task-level recomputation of main predictions, policy metrics, intervals, and RQ3 curves from the retained outcomes and embeddings. Figure 3 enumerates every distinct threshold-induced policy, including zero and both constant policies; triangles mark the highest retrospective success. The operating points are checked against the main table. The package records input hashes, seeds, and commands for rebuilding the curves and figures. The historical diagnostics include eligibility counts, family holdouts, risk–coverage curves, local intervals, and threshold sweeps. Archived catalog-policy comparisons use a separate calibration regime with a 3% absolute success-rate margin and remain separate from the canonical policy estimates.

C Task–Skill Structure and Support Coverage

Table 1 lists the task and skill inventories. All three agents share the audited task IDs and supplied conditions, so their task–skill structure is counted once below.

Single-skill families. ToolQA has 100 tasks for each of 13 skills and 130 for one; MedCalc-Bench has 20 tasks per skill and LogicBench has 40. Their analysis groups correspond to the supplied skills. For the within-family predictors, group size limits the available support.

Bundles and sparse groups. BigCodeBench injects all annotated skills together, forming 575 distinct bundles. Its 3,140 annotation edges give each task two to six supplied skills; these edges record membership, not separate skill interventions. The 86 focal analysis groups have median size

three, including 29 singletons. Consequently, a large inventory can still offer little support within a particular group.

This structure helps interpret the selection decomposition in Appendix B.3. A singleton has no within-group support after target exclusion. Its within-group selection term is also exactly zero, since its action equals the group's use rate; the same holds for any group whose tasks all receive the same action. Small groups therefore limit both reusable evidence and opportunities for within-group selection. Group membership also does not fix the entire injected bundle, so eligible neighbors can receive different skill combinations. These features are consistent with the weak within-group results, but the present comparison does not isolate sparsity from bundle heterogeneity or representation.

One artifact across task types. SpreadsheetBench uses one shared skill for 274 cell-level and 125 sheet-level tasks. The two analysis groups distinguish task types, not different skills.

Observed help and harm. As an example, the ToolQA profiles contain 265 helpful and 35 harmful Qwen pairs, and 177 helpful and 86 harmful GLM pairs; the remainder are ties. These single-pair observations include execution noise, as discussed in Appendix A.2.

D GENERALIZATION STRESS TESTS

These Qwen-Turbo panels evaluate unseen tasks and shifted task mixtures, complementing the main retrospective inventories. All predictions were serialized before the corresponding target outcomes were collected. They retain their historical configurations, distinct from the current $k = 6$, zero-threshold main analysis. Table 7 distinguishes the target constructions and sequential follow-ups.

Table 7: Prospective generalization stress tests. Success gain is the absolute difference between SKILLDELTA and Always-off; tokens saved are relative to Always-on. The panels differ in target construction.

Transfer axis	Known	Target	AUROC	Balanced accuracy	Success gain	Tokens saved
ToolQA, unseen tasks within 8 families	96	160	.753	.689	+20.00%	3.62%
MedCalc, new tasks across 55 families	440	660	.719	.667	+16.21%	32.35%
BigCode, scale follow-up (20/family)	200	150	.716	.663	+2.00%	64.43%
BigCode, natural remaining distribution	350	200	.502	.572	+7.00%	32.88%

Note. Known is the number of paired support tasks used to freeze the predictor; Target is the outcome-unseen audit panel. BigCode scale is a sequential follow-up after a 12-task/family pilot; the natural panel is a later remaining-population extension.

Frozen predictor settings. All four panels use question-only `text-embedding-3-small` vectors and nonnegative-cosine-weighted signed gains. ToolQA and MedCalc use same-family $k = 3$ neighbors, with thresholds 0.25 and $150/440 \approx 0.341$. BigCode scale averages all 20 supports per focal family, with threshold 0.105. The natural panel uses all 35 supports in an available focal family, otherwise the global 240 nearest supports, with threshold -0.0258794. The first three thresholds are the support positive-gain prevalences; the last maximizes leave-one-out balanced accuracy on support tasks, breaking ties toward fewer skill uses. Each threshold was fixed before target execution.

Matched and broader task mixtures. ToolQA retains the support families; its success gain is positive, while the token-saving interval crosses zero. MedCalc changes the task inventory and calculator-family mixture under the same agent. Both provide prospective evidence within supported task mixtures.

Support expansion and distribution shift. The BigCode scale panel follows a pilot, so its evidence is sequential. The natural-distribution extension retains a success gain over Always-off while task-level ranking falls to chance. This shows why improvement over skipping and gain ranking must be assessed separately under shift.

The archived ToolQA frozen-action re-execution concerns trajectory stability on reused tasks and is kept separate from these new-task panels. None of these panels evaluates unseen skills.