A FIX IS NOT A RULE: LEARNING WHEN TO REUSE AGENT CORRECTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model agents can learn from failures, generate behavioral corrections, and retain them for future tasks. Yet successfully fixing one failure does not determine when a correction should be reused: the same correction may help, have no effect, or cause harm across different tasks. Agents therefore need to learn both how to correct mistakes and the conditions under which a correction is useful. We propose an experience selection method based on correction scope learning. By observing the direct effects of the same correction across task contexts, the method predicts its utility and guides selective reuse. Experiments on LifelongAgentBench and WebShop, together with supplementary experiments using GLM-5, DeepSeek-V4-Flash, and Qwen3.6-27B, show that the same correction can have positive or negative effects across tasks. Further experiments with GLM-5 show that, under the same activation budget, HCAR-Risk improves mean deployment gain by 0.642 per correction over activation-matched random selection and increases the proportion of selected correction–task pairs with positive effects from 51.9% to 87.8%. These findings establish correction scope as a key learning problem in agent experience reuse: agents need to accumulate effective corrections and use direct evidence of their effects to decide when to apply them.

1 INTRODUCTION

Self-improving agents are increasingly able to learn from their failures. Recent approaches allow agents to retain feedback (Shinn et al., 2023), refine responses (Madaan et al., 2023), and acquire reusable skills (Wang et al., 2023). These mechanisms enable agents to accumulate experience and improve over time. However, accumulating experience introduces a fundamental challenge: a correction that solves one failure may not be appropriate for future tasks.

A correction learned from one task provides evidence that it can address the original failure, but it does not reveal when the same correction should be applied again. For example, a verification-oriented correction may improve tasks requiring high factual accuracy while introducing unnecessary overhead in tasks where verification is not needed. Therefore, self-evolving agents need not only to learn how to correct mistakes, but also to learn the conditions under which a correction remains effective.

In this work, we study the problem of correction reuse in self-evolving agents. We refer to a retained correction as a behavioral change (BC). We distinguish two decisions that are often conflated: whether a correction is useful for the failure that produced it, and whether the same correction should be reused in future tasks. We study the second problem by measuring the effects of the same BC across different task contexts and learning when it should be activated.

HCAR learns correction scope from direct observations of BC effects (Figure 1). Rather than applying retained corrections indiscriminately, it estimates whether a correction is likely to improve a given task context and selectively activates corrections when appropriate.

We evaluate correction reuse in LifelongAgentBench and WebShop, and measure future effects in separate controlled cohorts generated by GLM-5, DeepSeek-V4-Flash, and Qwen3.6-27B. Across these settings, previously learned corrections exhibit substantial task dependency: a correction that helps one context may have no benefit or even harm another. In a controlled GLM-5 evaluation, learning correction scope from the same correction's direct effects improves reuse compared with selection without scope estimation.

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.

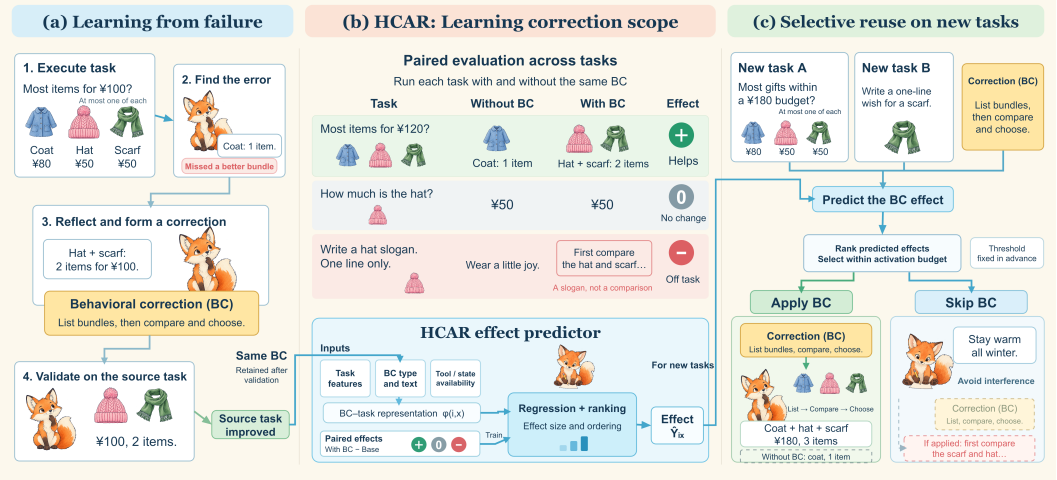


Figure 1: **Learning when to reuse a behavioral correction.** (a) An agent reflects on a failure and retains a behavioral correction (BC) that improves its source-task outcome. (b) Paired evaluations with and without the same BC reveal positive, neutral, and negative effects across tasks. HCAR learns from these effects and correction–task features to predict the benefit of reuse. (c) On new tasks, predicted effects guide selective application within an activation budget, while skipped tasks use the base policy. The shopping and writing examples are illustrative.

Our contributions are:

- **A new perspective on agent self-evolution.** We identify correction scope learning as a missing capability in self-improving agents, separating learning a correction from deciding when to reuse it.
- **A correction scope learning framework.** We propose a method that uses the observed effects of the same correction across task contexts to guide selective reuse.
- **An empirical study of correction generalization.** Across multiple LLM-based agent systems and task environments, we show that retained corrections have heterogeneous future effects and that scope-aware reuse improves deployment decisions. .

2 Related Work

Correction and in-trajectory recovery. Reflection and self-revision enable agents to improve their behavior after receiving feedback (Shinn et al., 2023; Madaan et al., 2023; Qu et al., 2024; Dou et al., 2025; Renze & Guven, 2024). Other approaches transform corrections into guidelines or training signals (Askari et al., 2024; He et al., 2026), enforce constraints during generation (Palmeira Ferraz et al., 2024; Zhao et al., 2025), detect or recover from misaligned actions (Fang et al., 2025; Wang et al., 2026), and compile user corrections into runtime checks (Zhou et al., 2026). These works mainly study how agents generate, execute, or incorporate corrections after failures. Our question concerns the subsequent reuse decision: after a correction has improved its original task, when should the agent apply the same correction to a later task?

Experience retention, retrieval, and skill lifecycles. Beyond immediate correction, agents can extract reusable skills from prior trials (Zhao et al., 2024; Wang et al., 2023) and manage them through memory and skill lifecycle mechanisms (Packer et al., 2024; Liang et al., 2026; Li et al., 2026; Lin et al., 2026; Zhang et al., 2026; Mao et al., 2026). Memory systems retrieve reflections (Park et al., 2023), update or reorganize stored records (Zhong et al., 2023; Xu et al., 2025), and distinguish different types of stored knowledge (Latimer et al., 2025); surveys and evolving-memory evaluations further characterize this design space (Zhang et al., 2024; Xu et al., 2026; Huang et al., 2026). Retaining or retrieving a correction makes it available for future use, but does not determine

whether applying it will improve the current task. We address this reuse decision by measuring the effect of activating a retained BC under a specific task context.

Evidence-guided skill decisions. Recent approaches model skill reliability and utility for managing agent memory and skills. Bayesian-Agent estimates skill reliability (Wu et al., 2026), while RoMeRL models utility for memory updates (Yang et al., 2026). We study a direct decision problem: whether activating a specific BC changes the outcome of a given task. We estimate this effect through paired Base-versus-BC rollouts under a fixed policy and task.

Effect heterogeneity and policy learning. Our study is related to heterogeneous treatment effects and policy learning, including causal partitioning, causal forests, and meta-learners (Athey & Imbens, 2016; Wager & Athey, 2017; Athey et al., 2018; Künzel et al., 2019). These methods study how intervention effects vary across contexts and how decisions can be optimized based on such effects (Athey & Wager, 2020). In our setting, the intervention is activation of a retained correction, and its effect is measured through paired agent rollouts under a fixed policy.

Agent evaluation. Existing agent evaluations measure reasoning, interaction, and tool-use capabilities across diverse environments, including interactive reasoning tasks (Yao et al., 2023b), general agent benchmarks (Liu et al., 2025), tool-using tasks (Mialon et al., 2023), API interaction (Qin et al., 2023), and database applications (Nan et al., 2023). Our evaluation focuses on a different question: given a retained correction, can an agent determine whether it should be reused for a particular task?

3 Problem formulation

A correction introduces two different decisions. The first is whether the correction successfully solves the failure that produced it. The second is whether the same correction should be reused on future tasks. We distinguish these two questions as *source admission* and *deployment scope*: source admission records improvement on the originating task, while deployment scope describes the tasks where the correction is expected to remain useful.

Source-admitted behavioral changes. Let π denote a fixed agent policy. A retained correction is represented as a behavioral change (BC), denoted by $\Delta\pi_i$, which specifies a modification to the agent behavior. A BC is generated from a failure of the base agent.

For a source task s_i, let u^{BC}_{i,s_i} and $u^{\mathrm{Base}}_{i,s_i}$ denote the native utility obtained with and without BC i in a paired rollout. We retain a BC when

$$u^{\mathrm{BC}}_{i,s_i} - u^{\mathrm{Base}}_{i,s_i} > 0. \tag{1}$$

This gate admits corrections that improved one observed source-task pair; it does not determine whether they will improve future tasks.

Future effect and deployment scope. A task context x comprises a specific task instance and the observable conditions under which the agent executes it. For a future context, we evaluate the base policy and the policy with BC i under identical conditions. Let U denote the task utility. The effect of applying BC i is defined as

$$Y_{ix} = U(\pi_i^{BC}, x) - U(\pi, x). \tag{2}$$

A correction is expected to be useful only in contexts where its expected effect is positive. We define the deployment scope of BC i as

$$\mathcal{A}_i = \{x : \mathbb{E}[Y_{ix}] > 0\}. \tag{3}$$

The scope definition motivates a decision rule for future BC–context pairs.

Correction selection objective. For a set of future contexts T, let $g_{ix} \in \{0, 1\}$ indicate whether BC i is applied to context x. The overall gain from correction selection is

$$G(g) = \sum_i \sum_{x \in T} g_{ix} \bar{Y}_{ix}, \tag{4}$$

where $\bar{Y}_{ix}$ denotes the mean observed paired effect across repeated evaluations. For BC i, we write $G_i(g) = \sum_{x \in T} g_{ix} \bar{Y}_{ix}$, so that $G(g) = \sum_i G_i(g)$.

The goal of correction scope learning is therefore to select BC–context pairs with positive expected effects rather than applying all retained corrections indiscriminately.

These definitions lead to three empirical questions:

RQ1: How do source-admitted BCs affect future tasks?

RQ2: Can observed effects of a BC guide its reuse on new contexts?

RQ3: How does correction reuse change when direct evidence is limited or when the correction itself is unseen?

4 METHOD

Given source-admitted behavioral changes $\{\Delta\pi_i\}_{i=1}^N$, our goal is to learn when each correction should be reused. For each BC–context pair (i, x), we estimate the future effect Y_{ix} defined in Section 3 and select corrections with high expected benefit.

Our method consists of three stages. First, we collect direct evaluations of each correction across calibration contexts by comparing Base and Base+BC rollouts. Second, we learn a function that predicts correction effectiveness from BC and task characteristics. Third, we selectively apply corrections to future contexts according to the predicted effectiveness.

BC–context representation. A correction decision depends on both task characteristics and correction properties. We represent each BC–context pair using

$$\phi(i, x) = [q(x), m_i, a_{ix}, m_i \otimes q(x), m_i a_{ix}, w_i], \tag{5}$$

where $q(x)$ describes task characteristics, m_i represents the correction mechanism type, a_{ix} indicates whether the task provides the tools or state required to execute the correction, and w_i represents the correction description.

The representation contains only information available before applying the correction. It does not include task outcomes, rollout results, or future effects.

Scope estimator. We learn a predictor

$$\hat{Y}_{ix} = f_\theta(\phi(i, x)), \tag{6}$$

which estimates the expected effect of applying BC i to context x. In the controlled evaluation, $f_\theta(\phi) = \theta^\top \phi + b$ and each training label $y_j = \bar{Y}_{i_j x_j}$ averages three paired repeats.

The magnitude of a predicted effect matters for deciding whether to activate a BC, while the relative ordering of BC–context pairs matters under a limited activation budget. We fit both using rank-augmented Ridge regression:

$$\min_{\theta, b} \|y - X\theta - b\mathbf{1}\|_2^2 + \lambda \sum_{(p,n)} \left[(y_p - y_n) - \theta^\top(\phi_p - \phi_n)\right]^2 + \gamma\|\theta\|_2^2. \tag{7}$$

For each of 1,200 sampled ranking pairs (p, n), both observations belong to the same BC, have unequal training effects, and satisfy $y_p > y_n$; either effect may be positive, zero, or negative. BCs

are sampled uniformly, followed by two distinct contexts within that BC. Pairs are sampled with replacement and have equal weight. We use $\lambda = 0.25$ and $\gamma = 3$. The learned score

$$z_{ix} = \theta^\top \phi(i, x) + b \tag{8}$$

is used to rank BC–context pairs according to their predicted benefit.

Correction selection. Predicted effects become reuse decisions only after choosing how many pairs to activate. We select this coverage on validation contexts and keep it fixed for held-out evaluation. HCAR-Risk operationalizes selection risk through the observed positive-effect precision of the activated validation pairs.

HCAR-Risk selects a global activation rate from $\{0.05, 0.10, \ldots, 1.00\}$. At each rate, it ranks validation pairs by z_{ix} and measures the proportion with $\bar{Y}_{ix} > 0$. Among rates with at least 70% positive-effect precision, it chooses the one with the largest validation gain $G(g)$, breaking ties toward lower activation. If no rate qualifies, it applies no corrections. After refitting on training and validation data, the selected rate is applied to the highest-scoring held-out pairs. The gate uses zero, independent of the descriptive sign margin δ. Sampling and out-of-fold score calibration are detailed in Appendix A.

Limited-evidence calibration. The controlled within-BC setting uses many direct observations of each correction. To examine reuse when such evidence is scarce, we also evaluate local calibration in WebShop: a separate predictor is fitted for each BC from ten direct probe contexts and then tested on fresh contexts. This study keeps the correction fixed and changes the amount and setting of its own evidence; cross-correction transfer is evaluated separately.

5 Experiments

We first measure future correction effects, then test selective reuse from same-correction evidence and transfer to harder settings.

5.1 Experimental setup

Models and environments. We use LifelongAgentBench (LAB), WebShop, and a controlled task environment. The base agents in LAB and WebShop both use DeepSeek-V4-Flash, providing task-native outcomes for lifelong task execution and interactive shopping, respectively (Zheng et al., 2025; Yao et al., 2023a). The controlled environment provides explicit task factors and execution conditions for analyzing correction effects and selection strategies. It supports two studies: independently collecting and evaluating corrections with GLM-5, DeepSeek-V4-Flash, and Qwen3.6-27B; and evaluating same-correction scope learning with GLM-5. Each model generates its own corrections rather than receiving corrections collected under another model.

Correction collection and paired evaluation. Corrections originate from genuine failures of the base agent. Given a failed task, the agent generates a candidate BC from the task and failure feedback. We retain it only when its source evaluation satisfies the admission criterion in Equation 1. LAB and WebShop contribute 49 and 20 analysis-ready BCs, respectively. Both scheduled 30 future contexts per BC, yielding 1,468 and 583 valid paired observations. Source-screening counts and admission accounting appear in Appendix A.1.

The three-model controlled evaluation admits ten BCs per model under the same one-candidate, one-validation procedure. Each BC is evaluated on five novel contexts with three paired repeats, producing 50 mean-effect observations per model.

Every evaluation compares Base with Base+BC on the same task, using the same base policy, generation budget, environment, and evaluator. For repeat r,

$$Y_{ix}^{(r)} = U_{ixr}^{BC} - U_{ixr}^{Base}, \qquad \bar{Y}_{ix} = \frac{1}{R} \sum_{r=1}^{R} Y_{ix}^{(r)}. \tag{9}$$

Thus $\bar{Y}_{ix}$ is the sample mean of paired effects. LAB and the original WebShop evaluation use one paired run; controlled evaluations use three. Agent failures within the fixed budget receive the native evaluator's failure score. Provider failures that produce no outcome are treated as missing, not imputed as zero.

Data splits and model configuration. Same-correction scope learning uses 30 GLM-5 BCs and 60 controlled contexts, split before evaluation into 36 training, 12 validation, and 12 test contexts per BC. Every BC–context pair has three paired repeats. We fit on training contexts, select the configuration on validation contexts, and refit on all 48 calibration contexts before evaluating the twelve held-out contexts. Training and test sets contain the same BCs but disjoint contexts, giving 360 test pairs. The model jointly fits the calibration data using the feature map $\phi(i, x)$ in Equation 5.

The WebShop direct-calibration study retains the existing 20 BCs. Each receives ten real direct-probe effects. Evaluation uses ten fresh contexts per BC with three paired repeats, giving 200 BC–context mean-effect test observations from 600 paired repeats. Fresh targets are disjoint from source tasks, probes, and earlier evaluation tasks. Local fits a separate Ridge predictor for each BC using a low-dimensional representation of task-text embeddings. Predictor and activation rules are fixed before fresh target outcomes are collected; probe composition and fitting details appear in Appendix C.

For cross-correction transfer, LAB and WebShop are split by BC: each test BC is excluded from global-model training and provides three support contexts at prediction time. Evaluation uses the future contexts of the 49 LAB and 20 WebShop BCs described above. Unlike same-correction learning, this test asks whether experience with other BCs helps predict a new one.

Comparison methods. The primary reference is activation-matched random selection, which selects BC–task pairs at the same activation rate without using effect predictions. Persistence applies each BC to every test context. Semantic Retrieval ranks pairs by textual similarity between the test task and the BC's source task. LLM Judge decides whether to apply a BC from the task and correction texts, without observing test effects.

HCAR-E selects by predicted benefit, while HCAR-Risk uses the validation rule in Section 4. HCAR-Risk, random selection, and Semantic Retrieval use the same 25% activation budget: 90 of the 360 test pairs, rather than a separate quota for each BC. Oracle selects pairs with positive observed test effects as a reference upper bound.

A post-hoc comparison on the same controlled split examines per-BC mean effects, within-BC task kernels, plain Ridge, partial pooling, and a feasibility-first rule. All use the same activation budget. WebShop direct calibration compares Local with SupportMean, which assigns a BC's mean probe effect to every new task, and SupportKernel, which weights probe effects by task similarity. Each applies a BC when its predicted effect is positive; the random reference matches Local's activation rate separately for each BC.

Evaluation metrics. Let T_i be the valid test contexts for BC i. Deployment gain $G_i(g)$ follows Equation 4; an inactive pair contributes zero. The primary controlled comparison is the mean per-BC gain difference:

$$\Delta_G = \frac{1}{N_{\mathrm{BC}}} \sum_i \left[G_i(g_{\mathrm{HCAR}}) - G_i(g_{\mathrm{random}}) \right]. \tag{10}$$

LAB and WebShop method comparisons report mean task utility,

$$V_i(g) = \frac{G_i(g)}{|T_i|}, \qquad V(g) = \frac{1}{N_{\mathrm{BC}}} \sum_i V_i(g), \tag{11}$$

which averages over tasks within each BC and then over BCs. Total gain $G(g)$, mean per-BC gain difference Δ_G, and mean task utility $V(g)$ thus use different normalizations.

We also report activation rate (the fraction of selected pairs), positive-effect precision among selected pairs, and false-positive cost (utility lost on selected harmful pairs); their formulas appear in Appendix B.1. Confidence intervals resample BCs using a BC-cluster bootstrap. Random-selection utilities are analytic expectations, not individual random draws. Additional context and two-way resampling analyses appear in Appendix B.

Table 1: Future-effect distributions of source-admitted corrections. Percentages are computed separately within each setting.

Setting / base model	BCs	Valid pairs	Positive (%)	Near-zero (%)	Negative (%)	Mixed-sign BCs
(a) Interactive task environments						
LAB	49	1,468	11.7	83.1	5.2	29/49
WebShop	20	583	16.1	67.4	16.5	20/20
(b) Three-model controlled evaluation						
GLM-5	10	50	62.0	14.0	24.0	7/10
DeepSeek-V4-Flash	10	50	60.0	10.0	30.0	9/10
Qwen3.6-27B	10	50	48.0	20.0	32.0	8/10

The three-model evaluation uses $\delta = 0.0200$ and three-repeat mean effects. LAB and WebShop retain their original sign classifications, whose numerical cutoffs were not preserved; threshold sensitivities appear in Appendix A. Raw utilities are not pooled across settings.

Table 2: Selective correction reuse on held-out controlled contexts with GLM-5.

Method	Gain ↑	Activation (%)	Precision (%) ↑	FP cost ↓
Persistence	28.1936	100.0	51.9	18.5198
Activation-matched random	7.0484	25.0	51.9	4.6300
Semantic Retrieval	22.7624	25.0	73.3	2.0311
LLM Judge	21.3320	20.6	71.6	1.6088
HCAR-E	30.8383	61.9	65.9	8.9798
HCAR-Risk	26.3054	25.0	87.8	0.9950
Oracle	46.7134	51.9	100.0	0.0000

Gain denotes $G(g)$. Random selection is reported in expectation. HCAR-Risk, random selection, and Semantic Retrieval use the same 25% activation budget; other rows use their own decision rules.

5.2 Correction effects depend on the task

We first measure future effects of source-admitted corrections. Given a margin δ, an effect is positive when $\bar{Y}_{ix} > \delta$, near-zero when $|\bar{Y}_{ix}| \leq \delta$, and negative when $\bar{Y}_{ix} < -\delta$. A correction is mixed-sign if it has both positive and negative future contexts. Table 1 summarizes the interactive environments and the three-model controlled evaluation.

In LAB, 29 of 49 corrections have both positive and negative future effects; all 20 WebShop corrections exhibit this pattern. The controlled evaluation finds mixed-sign effects for seven GLM-5, nine DeepSeek-V4-Flash, and eight Qwen3.6-27B corrections (Figure 2). Success on the source task therefore does not authorize unrestricted reuse. The same correction can help one task and harm another: its future effectiveness depends on the correction–task relationship.

5.3 Learning scope improves selective reuse

We use a correction's effects on calibration contexts to predict its benefit on new contexts and decide whether to apply it. Table 2 reports results for the GLM-5 controlled evaluation.

At the same activation budget, HCAR-Risk improves mean per-correction deployment gain over activation-matched random selection by 0.6419 (95% confidence interval: 0.4307–0.8490). Positive-effect precision rises from 51.9% to 87.8%, while false-positive cost falls from 4.6300 to 0.9950. Direct effects help identify more beneficial correction–task pairs and reduce harmful applications.

Semantic Retrieval and LLM Judge achieve total gains of 22.7624 and 21.3320, with positive-effect precision of 73.3% and 71.6%, respectively. HCAR-Risk achieves higher gain and precision, showing that observed effects provide useful selection information beyond source-task similarity and language-model judgment.

Persistence applies every correction and obtains 28.1936 total gain, but incurs 18.5198 false-positive cost. HCAR-Risk activates only a quarter of the pairs, retaining a similar total gain with 0.9950 false-positive cost. HCAR-E activates 61.9% of pairs and achieves higher total gain, alongside greater

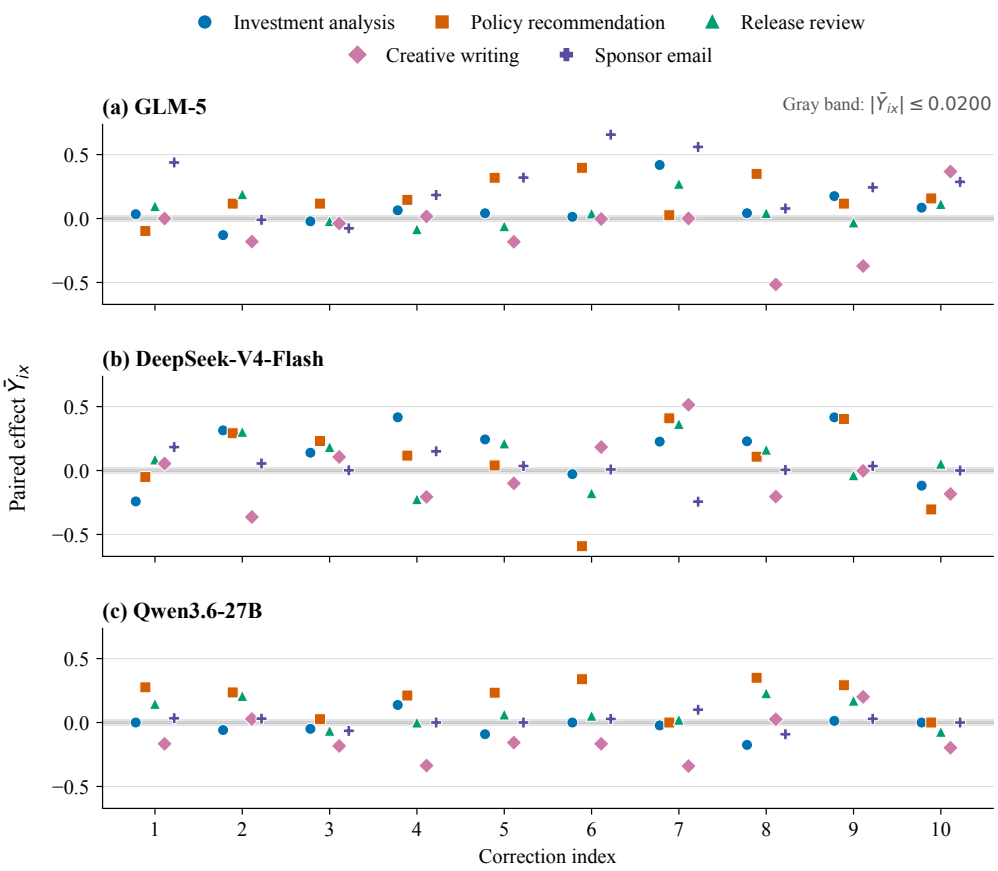


Figure 2: Future paired effects of each correction in the three-model controlled evaluation. Each point averages three paired repeats on one of five novel contexts. Colors and shapes identify an investment note, a policy recommendation, a system release review, a creative story opening, and a sponsor email. Each denotes one fixed task instance, not an entire task category (Appendix Table 5). The gray band denotes the near-zero interval with $\delta = 0.0200$. Corrections are collected independently for each model, so matching indices across panels do not identify the same correction.

losses from negative effects. These operating points distinguish aggregate benefit from selection precision.

Comparing effect predictors. To separate the value of direct effect evidence from the value of a particular predictor, Table 3 compares methods on the same calibration data. Each selects 90 test pairs, holding activation fixed.

The per-correction mean obtains 12.8801 total gain. Adding task information improves gain with kernels, Ridge, and partial pooling. HCAR-Risk exceeds the per-correction mean, task kernel, and feasibility-first rule, with positive confidence intervals for all three contrasts.

Plain Ridge and partial-pooling models perform similarly to HCAR-Risk. Thus, task-dependent direct effects matter more here than the particular predictor.

5.4 Direct calibration and cross-correction transfer

We test direct calibration in WebShop with $K = 10$ probes from each BC and cross-correction transfer in LAB and WebShop with $K = 3$ support contexts for each held-out BC. Table 4 reports mean task utility.

Table 3: Effect predictors with the same calibration data and activation budget.

Predictor	$G(g)\uparrow$	Precision (%) $\uparrow$	Δ_G (95% CI)
HCAR-Risk	26.3054	87.8	Reference
Plain Ridge	26.6021	87.8	-0.0099 $(-0.0503,$ $0.0293)$
Feasibility-first	21.9144	67.8	0.1464 $(0.0696,$ $0.2349)$
Per-correction mean	12.8801	48.9	0.4475 $(0.2592,$ $0.6439)$
Within-correction task kernel	22.4974	71.1	0.1269 $(0.0130,$ $0.2417)$
Partial pooling: BC intercept	27.3355	88.9	-0.0343 $(-0.0880,$ $0.0092)$
Partial pooling: BC slope	26.2218	84.4	0.0028 $(-0.0415,$ $0.0472)$

$G(g)$ is total deployment gain; precision is the percentage of selected pairs with positive effects. For each row's policy g, $\Delta_G = N_{\mathrm{BC}}^{-1} \sum_i [G_i(g_{\mathrm{HCAR}}) - G_i(g)]$; positive values favor HCAR-Risk. Parentheses give 95% BC-bootstrap confidence intervals. This post-hoc comparison uses the same calibration split and 90 test activations. Feasibility-first uses execution conditions and random budget filling, reported in expectation.

Table 4: Mean task utility for same-correction direct calibration and cross-correction transfer.

(a) Limited-evidence calibration: WebShop, $K = 10$

Comparator h	$V(g)$	$V(h)$	Δ_V (95% CI)
Matched random	0.0263	0.0230	0.0033 $(-0.0024,$ $0.0098)$
SupportKernel	0.0263	0.0281	-0.0017 $(-0.0088,$ $0.0047)$
SupportMean	0.0263	0.0284	-0.0021 $(-0.0089,$ $0.0040)$

(b) Cross-correction transfer: $K = 3$ support contexts per test correction

Environment	BCs	Valid pairs	$V(g)$	$V(h)$	Δ_V (95% CI)
LAB	49	1,468	0.0586	0.0558	0.0028 $(-0.0016,$ $0.0073)$
WebShop	20	583	-0.0027	-0.0041	0.0014 $(-0.0030,$ $0.0066)$

K counts direct-effect contexts per BC, excluding source and test tasks; repeats do not increase K. $V(\cdot)$ is mean task utility (Equation 11) and $\Delta_V = V(g) - V(h)$. In (a), g is Local and h the listed comparator; in (b), g is HCAR and h matched random. Parentheses give 95% BC-bootstrap confidence intervals. Valid pairs count BC–context observations. Collection and audit details appear in Appendix C.

With ten direct-probe contexts in WebShop, Local achieves 0.0263 mean task utility, exceeding activation-matched random selection by 0.0033, with a confidence interval that includes zero. SupportKernel and SupportMean obtain 0.0281 and 0.0284, respectively; this evaluation does not establish a Local advantage over either simple alternative.

Cross-correction transfer yields utility differences of 0.0028 in LAB and 0.0014 in WebShop, with both intervals including zero. These results suggest that correction scope remains difficult to learn reliably from limited direct evidence and does not readily transfer from existing corrections to new ones under the evaluated settings. Retrospective transfer diagnostics are reported in Appendix C.

6 Conclusion

We study how agents should decide where to reuse a correction after learning to address a failure. Experiments on LAB and WebShop, together with controlled evaluations using three base models, show that the same source-admitted correction can produce positive or negative effects on different tasks: improvement on the source task does not determine its future deployment scope. In the GLM-5 controlled evaluation, direct effect evidence from the same correction improves deployment gain and reduces harmful applications at a matched activation budget. These findings show that effective agent experience reuse requires learning not only how to correct an error, but also when to use that correction again.

ETHICS STATEMENT

This study evaluates behavioral corrections only in benchmark environments and controlled settings, without human participants or personal data. Deployment beyond these settings should account for task-specific risks, user preferences, and operational costs.

AI USE STATEMENT

We used generative AI solely for language polishing and assistance with code development. The authors reviewed all AI-assisted outputs and take full responsibility for the content of the manuscript and its associated research artifacts.

REPRODUCIBILITY STATEMENT

The appendix describes task construction, data splits, model fitting, correction selection, and supplementary analyses. Predictions and analysis rules for the WebShop evaluation on fresh test tasks were fixed before test rollouts began. Appendix C documents the subsequent interim inspection of results and the unchanged prediction, selection, and analysis rules.

REFERENCES

Arian Askari, Christian Poelitz, and Xinye Tang. Magic: Generating self-correction guideline for in-context text-to-sql, 2024.

Susan Athey and Guido Imbens. Recursive partitioning for heterogeneous causal effects. *Proceedings of the National Academy of Sciences*, 113(27):7353–7360, 2016. ISSN 1091-6490. doi: 10.1073/pnas.1510489113.

Susan Athey and Stefan Wager. Policy learning with observational data, 2020.

Susan Athey, Julie Tibshirani, and Stefan Wager. Generalized random forests, 2018.

Zi-Yi Dou, Cheng-Fu Yang, Xueqing Wu, Kai-Wei Chang, and Nanyun Peng. Re-rest: Reflection-reinforced self-training for language agents, 2025.

Haishuo Fang, Xiaodan Zhu, and Iryna Gurevych. Preemptive detection and correction of misaligned actions in llm agents, 2025.

Keqing He, Dayuan Fu, Lele Yang, Weiran Xu, and Chao Wang. Selfcorrect-agent: Toward robust and generalizable llm-based agents. *Neurocomput.*, 670(C), June 2026. ISSN 0925-2312. doi: 10.1016/j.neucom.2025.132546.

Wei-Chieh Huang, Weizhi Zhang, Yueqing Liang, Yuanchen Bei, Yankai Chen, Tao Feng, Xinyu Pan, Zhen Tan, Yu Wang, Tianxin Wei, Shanglin Wu, Ruiyao Xu, Liangwei Yang, Rui Yang, Wooseong Yang, Chin-Yuan Yeh, Hanrong Zhang, Haozhen Zhang, Siqi Zhu, Henry Peng Zou, Wanjia Zhao, Song Wang, Wujiang Xu, Zixuan Ke, Zheng Hui, Dawei Li, Yaozu Wu, Langzhou He, Chen Wang, Xiongxiao Xu, Baixiang Huang, Juntao Tan, Shelby Heinecke, Huan Wang, Caiming Xiong, Ahmed A. Metwally, Jun Yan, Chen-Yu Lee, Hanqing Zeng, Yinglong Xia, Xiaokai Wei, Ali Payani, Yu Wang, Haitong Ma, Wenya Wang, Chenguang Wang, Yu Zhang, Xin Eric Wang, Yongfeng Zhang, Jiaxuan You, Hanghang Tong, Xiao Luo, Xue Liu, Yizhou Sun, Wei Wang, Julian McAuley, James Zou, Jiawei Han, Philip S. Yu, and Kai Shu. A survey of agent memory in the second half: Towards self-evolving and long-horizon agents, 2026.

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.

Chris Latimer, Nicoló Boschi, Andrew Neeser, Chris Bartholomew, Gaurav Srivastava, Xuan Wang, and Naren Ramakrishnan. Hindsight is 20/20: Building agent memory that retains, recalls, and reflects, 2025.

Zhuo Li, Guodong Du, Zesheng Shi, Weiyang Guo, Weijun Yao, Yuan Zhou, Jiabo Zhang, and Jing Li. Skill weaving: Efficient llm improvement via modular skillpacks, 2026.

Sirui Liang, Pengfei Cao, Jian Zhao, Wenhao Teng, Xiangwen Liao, Jun Zhao, and Kang Liu. Learning how to remember: A meta-cognitive management method for structured and transferable agent memory, 2026.

Huawei Lin, Peng Li, Jie Song, Fuxin Jiang, and Tieying Zhang. Muse-autoskill: Self-evolving agents via skill creation, memory, management, and evaluation, 2026.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating llms as agents, 2025.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback, 2023.

Bo Mao, Jie Zhou, Yutao Yang, Xin Li, Xian Wei, Qin Chen, Xingjiao Wu, and Liang He. Learning while acting: A skill-enhanced test-time co-evolution framework for online lifelong learning agents, 2026.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants, 2023.

Linyong Nan, Ellen Zhang, Weijin Zou, Yilun Zhao, Wenfei Zhou, and Arman Cohan. On evaluating the integration of reasoning and action in llm agents with database question answering, 2023.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems, 2024.

Thomas Palmeira Ferraz, Kartik Mehta, Yu-Hsiang Lin, Haw-Shiuan Chang, Shereen Oraby, Sijia Liu, Vivek Subramanian, Tagyoung Chung, Mohit Bansal, and Nanyun Peng. Llm self-correction with decrim: Decompose, critique, and refine for enhanced following of instructions with multiple constraints. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 7773–7812. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-emnlp.458.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior, 2023.

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

Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. Recursive introspection: Teaching language model agents how to self-improve, 2024.

Matthew Renze and Erhan Guven. Self-reflection in large language model agents: Effects on problem-solving performance. In *2024 2nd International Conference on Foundation and Large Language Models (FLLM)*, pp. 516–525. IEEE, 2024.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023.

Stefan Wager and Susan Athey. Estimation and inference of heterogeneous treatment effects using random forests, 2017.

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, 2023.

Pan Wang, Yihao Hu, Hang Wang, Zirui Lv, Xin Zhang, Jianshe Li, Jiang-Ming Yang, Wei Wu, and Yongqi Tong. Diagnosis before recovery: Turning agent failures into selective self-correction, 2026.

Xiaojun Wu, Cehao Yang, Honghao Liu, Xueyuan Lin, Wenjie Zhang, Zhichao Shi, Xuhui Jiang, Chengjin Xu, Jia Li, and Jian Guo. Bayesian-agent: Posterior-guided skill evolution across llm agent harnesses, 2026.

Jundong Xu, Qingchuan Li, Jiaying Wu, Yihuai Lan, Shuyue Stella Li, Huichi Zhou, Bowen Jiang, Lei Wang, Jun Wang, Anh Tuan Luu, Caiming Xiong, Hae Won Park, Bryan Hooi, and Zhiyuan Hu. Evoarena: Tracking memory evolution for robust llm agents in dynamic environments, 2026.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents, 2025.

Yi Yang, Zhennan Chen, Yihong Zhuang, Tiehan Fan, Yinan Chen, Jian Li, Jian Yang, and Ying Tai. Romerl: Balancing feedback coverage and the memory-reward trap in self-evolving agent memory via reduced-order utility states, 2026.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents, 2023a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023b.

Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. Memskill: Learning and evolving memory skills for self-evolving agents, 2026.

Zeyu Zhang, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Quanyu Dai, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model based agents, 2024.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners, 2024.

Xutong Zhao, Tengyu Xu, Xuewei Wang, Zhengxing Chen, Di Jin, Liang Tan, Yen-Ting, Zishun Yu, Zhuokai Zhao, Yun He, Sinong Wang, Han Fang, Sarath Chandar, and Chen Zhu. Boosting llm reasoning via spontaneous self-correction, 2025.

Junhao Zheng, Xidi Cai, Qiuke Li, Duzhen Zhang, ZhongZhi Li, Yingying Zhang, Le Song, and Qianli Ma. Lifelongagentbench: Evaluating llm agents as lifelong learners, 2025.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory, 2023.

Yujun Zhou, Kehan Guo, Haomin Zhuang, Xiangqi Wang, Yue Huang, Zhenwen Liang, Pin-Yu Chen, Tian Gao, Nuno Moniz, Nitesh V. Chawla, and Xiangliang Zhang. Getting better at working with you: Compiling user corrections into runtime enforcement for coding agents, 2026.

A TASKS AND CALIBRATION DETAILS

A.1 SOURCE SCREENING AND ADMISSION

LAB retained 50 corrections from 100 failure cases; 49 met the subsequent data-integrity requirements. WebShop screened 500 source tasks, yielding 173 failures and 20 admitted corrections. In the three-model controlled evaluation, GLM-5, DeepSeek-V4-Flash, and Qwen3.6-27B screened 38, 45, and 30 source tasks, yielding 28, 28, and 19 failures and ten admitted BCs each.

A.2 TASK CONTEXTS AND PAIRED EVALUATION

A context x is a task instance together with the conditions observable to the agent. In the controlled environment, these include the prompt, supplied materials, constraints, and available tools. In LAB and WebShop, they are the benchmark task and its environment observations. Table 5 describes the five tasks shown in Figure 2.

The unit of effect measurement is a correction–context pair. Repeated evaluations of that pair are averaged into $\bar{Y}_{ix}$, rather than counted as additional contexts. Accordingly, K counts distinct probe or support contexts for each correction, excluding its source and test tasks. Task contexts may be shared across corrections.

Table 5: Task requirements for the five contexts in Figure 2.

Task	Requirements
Investment analysis	Write an investment note using supplied evidence on margin decline, cash runway, and demand growth; include a recommendation and risks.
Policy recommendation	Recommend a policy using supplied evidence on uptake, cost, and rural access; include a recommendation and risks.
Release review	Assess release readiness using latency, error rate, the service-level objective, and an unresolved dependency conflict; give a release decision and risks.
Creative writing	Write a 130–170-word opening to a night-train story, emphasizing imagery and voice, without citations, headings, recommendations, or risk sections.
Sponsor email	Write a 90–130-word sponsor email stating the decision needed, a Friday deadline, and one risk, without citations or headings.

In the controlled calibration study, Base and Base+BC receive the same prompt and documents, use the same two-stage agent and model, and are scored by the same deterministic rubric. Their execution order is determined by a hash of the correction, context, and repeat identifiers. Requests use temperature zero without a shared decoding seed; the pairing therefore matches task inputs and evaluation, not model-sampling noise.

A.3 TASK REPRESENTATION AND CORRECTION EXECUTION

Task-factor values are assigned deterministically from the context index before outcome collection, taking values of 0 or 3 before normalization. The task generator translates these values into constraints such as the number of required sources or the depth of planning dependencies. The factors describe the constructed tasks; some fields are not independently varied within every task domain.

The availability indicator a_{ix} records whether the task supplies the tools or state needed to execute a correction. Analysis tasks support information acquisition, planning tasks support planning transformations, verification tasks support verification, and reasoning tasks support reasoning transformations. Creative-writing and communication tasks provide none of these specialized aids. Availability thus describes execution conditions, not whether applying the correction will improve the outcome. Table 6 illustrates the operations performed by retained corrections.

A.4 FITTING AND SELECTION

The controlled calibration study uses 30 BCs, each with 36 training, 12 validation, and 12 test contexts. Each observed effect averages three paired repeats. The fitting and selection sequence is:

Table 6: Examples of source-admitted corrections in the controlled calibration study. Descriptions abbreviate the collected workflows.

Correction type	Task condition	Required actions	Completion rule
Information acquisition	A compliance decision depends on potentially outdated policy records.	Retrieve the records, check policy versions and effective dates, and link each comparison to supporting evidence.	Withhold the determination if dates or required comparisons remain unverified.
Reasoning transformation	A decision involves alternatives with competing constraints.	Enumerate the options, normalize their metrics, and compare every option against each constraint.	Select an option after completing the comparison.
Planning transformation	A schedule assessment depends on scheduler state and allowed remedies.	Retrieve the baseline completion, deadline, and conflicts; apply the specified minimal remedy and check the revised completion.	Report absent scheduler fields as missing rather than infer their values.
Verification	An authorization decision requires a validation operation.	Run validation and record failed checks, observed values, and limits.	Authorize only when validation has run and every required check passes.

1. Fit the workflow-text vocabulary on the 30 BCs, then fit the rank-augmented Ridge predictor on the 30×36 training pairs using Equation 5 and 1,200 sampled within-BC ranking pairs.

2. Calibrate the scores using the out-of-fold affine map described below. Rank the 30×12 validation pairs and choose the activation rate with the greatest validation gain among those meeting 70% positive-effect precision; break ties toward the lower rate.

3. Refit the predictor on all 30×48 calibration pairs and refit the score map without using test outcomes.

4. Rank the 30×12 held-out test pairs, activate the selected global fraction, and evaluate those decisions using the test effects.

This is a global selection over BC–context pairs, not an equal quota for each BC.

Correction descriptions use unigram–bigram TF–IDF features capped at 96 dimensions, fitted on the 30 calibration BCs. Ranking pairs are sampled with replacement using seed 20260811: choose a BC uniformly, then two distinct contexts uniformly within that BC, and discard ties. Each sampled pair has the same weight irrespective of its repeat-level variance. The same 1,200-pair procedure is repeated when fitting on all 48 calibration contexts per BC. The Ridge intercept is unpenalized. A five-fold out-of-fold affine map fitted to plain-Ridge training predictions is applied to the rank-augmented scores before validation ranking; the map is refitted on all calibration contexts before test ranking. Both observed slopes are positive, so these maps preserve the rankings.

Semantic retrieval uses TF–IDF unigrams and bigrams from the source and context prompts. It ranks pairs by the cosine similarity between each task and the correction's source prompt, selecting the top 25%. The LLM judge receives the task prompt and correction workflow, without effect labels, and returns an apply-or-skip decision. Its 20.6% activation rate follows directly from those decisions.

B SUPPLEMENTARY ANALYSES

B.1 METRIC DEFINITIONS

For valid test contexts T_i, activation rate, positive-effect precision, and false-positive cost are

$$\text{Activation}(g) = \frac{\sum_i \sum_{x \in T_i} g_{ix}}{\sum_i |T_i|}, \tag{12}$$

$$\text{Precision}(g) = \frac{\sum_i \sum_{x \in T_i} g_{ix} \mathbf{1}[\bar{Y}_{ix} > 0]}{\sum_i \sum_{x \in T_i} g_{ix}}, \tag{13}$$

$$C_{\text{FP}}(g) = \sum_i \sum_{x \in T_i} g_{ix} \max(-\bar{Y}_{ix}, 0). \tag{14}$$

False-positive cost measures task utility lost from harmful BC activations, not separately priced latency or API costs.

B.2 CALIBRATION, ALLOCATION, AND UNCERTAINTY

Table 7 adds confidence intervals for comparisons with the secondary baselines and examines the predictor's feature groups. Each feature ablation repeats fitting and validation-based activation selection on the same data. Removing availability features leaves no activation rate that meets the 70% validation-precision requirement. Using task, mechanism, and availability features without workflow text retains most of the full model's gain.

Table 7: Additional comparisons and feature ablations in controlled calibration.

(a) Secondary comparisons

Comparator	Δ_G	95% CI
Semantic retrieval	0.1181	(0.0269, 0.2067)
Blinded LLM judge	0.1658	(0.0691, 0.2676)
Persistence	−0.0629	(−0.2056, 0.0890)
HCAR-E (ungated)	−0.1511	(−0.2638, −0.0379)

(b) Feature ablations

Features	Activation (%)	$G(g)$	Precision (%)
Full HCAR-Risk	25.0	26.3054	87.8
No task–mechanism interaction	20.0	24.2790	91.7
No availability features	0.0	0.0000	—
Task + mechanism + availability	25.0	25.6294	86.7
Workflow only	5.0	3.5799	50.0

Δ_G is the mean per-correction gain of HCAR-Risk minus the comparator; intervals use 10,000 correction-level bootstrap resamples (seed 20260810). Comparators retain their activation rates from Table 2. Each ablation reselects activation on validation data. Without availability features, no rate meets the 70% precision requirement; no pairs are selected, so precision is undefined.

The primary bootstrap samples corrections with replacement, retaining all 12 test contexts for each sampled correction. The random baseline uses its analytic expected gain at 25% activation. Because corrections share test contexts, Table 8 also reports context and two-way resampling with the selection decisions held fixed. The gain-difference interval remains above zero under each resampling scheme.

The same table varies the validation-precision requirement to show the trade-off between selecting more pairs and avoiding harmful applications. These are supplementary analyses; the primary requirement remains 70%. Figure 3 shows the corresponding relationship between predicted scores, observed effects, and activation rate.

Table 9 varies the ranking weight, Ridge penalty, and within-BC pair construction on the frozen split. These post-hoc variants select 20–30% of test pairs. With no ranking term, gain is 26.6021 at the same 25% activation rate, close to the frozen model's 26.3054; the ranking term is not the sole source of selection quality.

Table 8: Sensitivity of controlled-calibration results to resampling and selection requirements.

(a) 95% confidence intervals under alternative resampling units

Resampling unit	Δ_G	Precision	$C_{\mathrm{FP}}/N_{\mathrm{BC}}$
Correction (primary)	(0.4307, 0.8490)	(0.8077, 0.9383)	(0.0068, 0.0719)
Shared context	(0.2363, 1.0792)	(0.7891, 0.9615)	(0.0030, 0.0721)
Correction and context	(0.2023, 1.2062)	(0.7356, 0.9882)	(0.0000, 0.1068)

(b) Validation-precision requirements

Required precision (%)	Activation (%)	$G(g)$	Precision (%)	C_{FP}
60	65.0	31.3466	64.7	9.5281
70	25.0	26.3054	87.8	0.9950
80	15.0	22.8334	90.9	0.1356
90	10.0	20.4049	100.0	0.0000

Panel (a) holds test decisions fixed and reports precision as a proportion. Panel (b) selects activation using the specified validation requirement and reports test metrics. The primary requirement is 70%. Gain and false-positive cost follow the definitions in Section 5.

Table 9: Post-hoc sensitivity of controlled correction selection to the ranking objective.

Variant	λ	γ	Activation (%)	Gain ↑	Precision (%) ↑
No ranking term	0.00	3	25.0	26.6021	87.8
Lower ranking weight	0.10	3	25.0	26.3805	87.8
Frozen primary	0.25	3	25.0	26.3054	87.8
Higher ranking weight	0.50	3	30.0	27.7172	84.3
Weaker Ridge penalty	0.25	0.3	25.0	26.6956	87.8
Stronger Ridge penalty	0.25	30	20.0	23.4211	83.3
Opposite-sign pairs	0.25	3	25.0	25.5458	85.6

All variants use the frozen 36/12/12 split, 1,200 sampled within-BC pairs where applicable, and the same 70% validation-precision rule. Opposite-sign pairs require one positive and one negative effect; the primary uses all unequal-effect pairs. The primary row reproduces the frozen result; other rows are post-hoc diagnostics.

Equal allocation across corrections. HCAR-Risk selects no pairs for five of the 30 corrections, with a median of four selected pairs per correction. Requiring exactly three selections per correction yields total gain 25.6861 and a mean per-correction gain difference of 0.6213 over matched random (95% CI: 0.4646–0.7856). Thus the improvement also appears under equal allocation across corrections. Among the original 90 selections, 82.2% exceed an effect margin of 0.0200 and 75.6% exceed 0.0500, compared with 87.8% above zero.

B.3 Source improvement and future effects

Table 10 reports the source improvement and five-context future effect profile of each correction in the three-model controlled evaluation. Table 11 relates source improvement to the mean, standard deviation, and range of those future effects. All correlation intervals include zero; within these ten-correction cohorts, a larger observed source improvement does not reliably predict the future profile.

Some correction workflows contain source-format tokens such as `[S1]`. To check whether these tokens account for mixed-sign effects, we exclude workflows containing them. Mixed signs remain in 7/10 GLM-5, 3/3 DeepSeek-V4-Flash, and 6/8 Qwen3.6-27B corrections under the same $\delta = 0.0200$ classification rule.

Sensitivity to the neutral band. Figure 4 varies the margin δ used to classify LAB and WebShop effects. As δ increases from zero to 0.5000, the mixed-sign count remains 29/49 in LAB and falls from 20/20 to 15/20 in WebShop. The pattern therefore extends beyond effects close to zero. These

Table 10: Source improvement and future effects for individual corrections in the three-model controlled evaluation.

GLM-5

BC	Source ΔU	$+$	0	$-$	Min. $\bar{Y}_{ix}$	Median $\bar{Y}_{ix}$	Max. $\bar{Y}_{ix}$
G01	0.0044	3	1	1	−0.0973	0.0337	0.4383
G02	0.0972	2	1	2	−0.1804	−0.0100	0.1902
G03	0.0048	1	1	3	−0.0767	−0.0220	0.1167
G04	0.1375	3	1	1	−0.0837	0.0640	0.1837
G05	0.0285	3	0	2	−0.1822	0.0413	0.3200
G06	0.0324	3	2	0	−0.0035	0.0395	0.6560
G07	0.0084	4	1	0	0.0006	0.2717	0.5600
G08	0.0624	4	0	1	−0.5154	0.0433	0.3493
G09	0.0504	3	0	2	−0.3714	0.1167	0.2435
G10	0.0708	5	0	0	0.0850	0.1573	0.3673

DeepSeek-V4-Flash

BC	Source ΔU	$+$	0	$-$	Min. $\bar{Y}_{ix}$	Median $\bar{Y}_{ix}$	Max. $\bar{Y}_{ix}$
D01	0.3500	3	0	2	−0.2418	0.0541	0.1833
D02	0.0276	4	0	1	−0.3631	0.2917	0.3140
D03	0.3500	4	1	0	0.0017	0.1393	0.2310
D04	0.3500	3	0	2	−0.2245	0.1167	0.4163
D05	0.2800	4	0	1	−0.1000	0.0397	0.2430
D06	0.2063	1	1	3	−0.5917	−0.0283	0.1833
D07	0.0156	4	0	1	−0.2433	0.3640	0.5144
D08	0.5524	3	1	1	−0.2041	0.1083	0.2287
D09	0.6944	3	1	1	−0.0371	0.0350	0.4157
D10	0.0104	1	1	3	−0.3043	−0.1183	0.0531

Qwen3.6-27B

BC	Source ΔU	$+$	0	$-$	Min. $\bar{Y}_{ix}$	Median $\bar{Y}_{ix}$	Max. $\bar{Y}_{ix}$
Q01	0.0432	3	1	1	−0.1663	0.0333	0.2753
Q02	0.0372	4	0	1	−0.0587	0.0303	0.2353
Q03	0.0700	1	0	4	−0.1833	−0.0646	0.0263
Q04	0.1376	2	2	1	−0.3367	0.0000	0.2113
Q05	0.1376	2	1	2	−0.1569	0.0000	0.2333
Q06	0.0688	3	1	1	−0.1663	0.0279	0.3387
Q07	0.1376	2	1	2	−0.3408	0.0000	0.1000
Q08	0.0750	3	0	2	−0.1750	0.0253	0.3500
Q09	0.3750	4	1	0	0.0130	0.1708	0.2917
Q10	0.2250	0	3	2	−0.1975	0.0000	0.0000

Each correction has five future contexts. The $+/0/-$ columns count context effects using $\delta = 0.0200$; each future effect averages three paired repeats. Source ΔU is the observed paired improvement used for admission.

Table 11: Association between source improvement and future-effect profiles.

Model	Future statistic	ρ	95% CI
GLM-5	Mean	−0.2121	(−0.8571, 0.4717)
GLM-5	Sd	−0.0909	(−0.7949, 0.7737)
GLM-5	Range	−0.1273	(−0.8101, 0.6754)
DeepSeek-V4-Flash	Mean	0.1779	(−0.6604, 0.8627)
DeepSeek-V4-Flash	Sd	−0.2516	(−0.8512, 0.4858)
DeepSeek-V4-Flash	Range	−0.2270	(−0.7732, 0.5017)
Qwen3.6-27B	Mean	−0.1902	(−0.9579, 0.5992)
Qwen3.6-27B	Sd	−0.1657	(−0.7492, 0.5631)
Qwen3.6-27B	Range	−0.2638	(−0.7864, 0.5009)

Entries are Spearman correlations with correction-level bootstrap 95% intervals. Each model contributes ten corrections; future summaries use five contexts per correction.

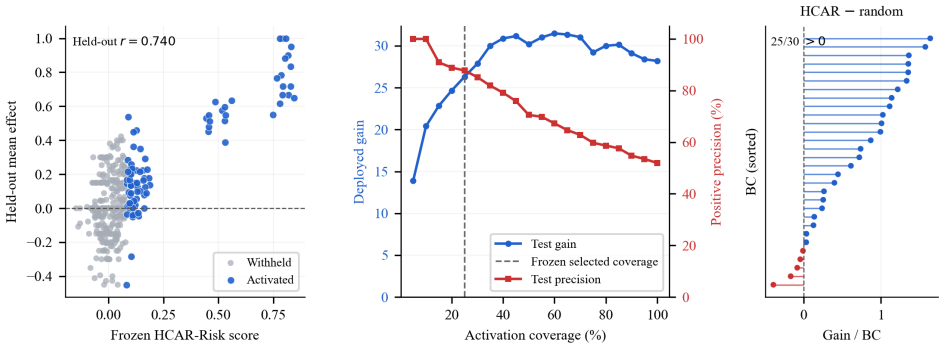


Figure 3: Selection behavior in the controlled calibration study. **Left:** Predicted scores and test effects (Pearson $r = 0.740$); blue points mark the pairs selected at 25% activation. **Center:** Gain and positive-effect precision across activation rates. **Right:** HCAR-Risk exceeds the expected gain of matched random for 25 of 30 corrections. Dashed lines mark zero gain or the activation rate chosen on validation data.

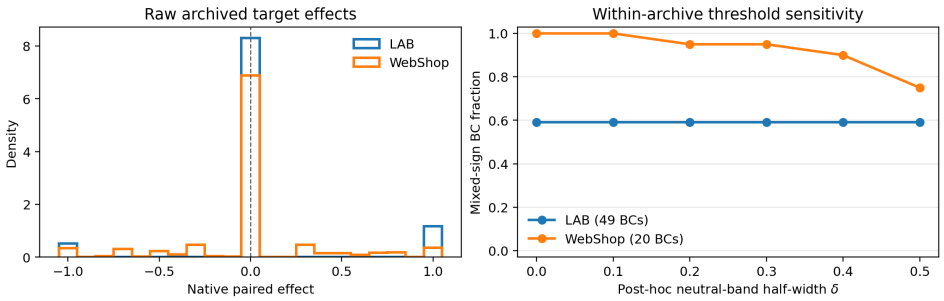


Figure 4: Future-effect distributions and sensitivity to the neutral band. **Left:** Observed paired effects in LAB and WebShop. **Right:** Fraction of corrections with both positive and negative effects as the classification margin δ increases. Utilities retain their environment-specific scales.

supplementary thresholds do not recover the original numerical cutoffs, which were not retained in the archived reports. Each curve uses its environment's native utility scale.

Why source success alone does not identify scope. Consider a correction $\Delta\pi_i$ with an observed source improvement $d > 0$ and a distinct future context x. Two data-generating processes can agree on the source outcome and all observed source information while assigning future effects $+a$ and $-a$, respectively, for any $a > 0$. Both satisfy the source admission gate in Equation 1, but only the first places x in $\mathcal{A}_i$. Distinguishing them requires additional assumptions connecting source and future tasks, or evidence about the correction beyond its source outcome.

C Direct Calibration and Transfer Diagnostics

C.1 Fresh-Target Direct Calibration

Each WebShop BC has ten real direct-probe effects: three inherited support contexts and seven additional probes. The feature processing, regularization selection, and activation rules were fixed before fresh target outcomes were collected. The 200 BC–context test observations average 600 paired repeats; no repeat was imputed.

C.2 CROSS-CORRECTION TRANSFER

The transfer comparisons in Section 5.4 have one-sided bootstrap and sign-flip p-values of 0.1123 and 0.1165 in LAB, and 0.2684 and 0.2961 in WebShop, respectively. Excluding all WebShop corrections affected by provider-side missing outcomes leaves 17 corrections and a mean task-utility difference of 0.0002 over matched random.

Table 12 examines two possible sources of predictive information using previously collected outcomes. Low-rank models estimate shared task-effect patterns across corrections. In WebShop, ranks one through three yield held-out-correction correlations near zero. LAB uses correction-specific task identifiers and lacks the shared task axis needed for this analysis.

The local diagnostic instead fits each correction from ten of its own observed effects. Diverse probes are compared with randomly selected probes using the same local model; SupportMean and SupportKernel provide simple alternatives. Local exceeds SupportKernel in WebShop but not LAB. These retrospective comparisons use earlier outcomes and are distinct from the fresh-target WebShop direct-calibration experiment.

Table 12: Retrospective analyses of shared structure and local calibration.

(a) Shared low-rank structure

Environment	Rank	BCs	Mean Spearman correlation
WebShop	1	20	−0.0190
WebShop	2	20	−0.0760
WebShop	3	20	0.0160

(b) Local calibration from ten historical effects per correction

Environment	BCs	Local	Random probes	SupportMean	SupportKernel	Δ_V
LAB	49	0.0531	0.0540	0.0599	0.0585	−0.0054
WebShop	19	0.0285	0.0202	0.0045	0.0208	0.0077

Panel (a) evaluates correlations on held-out corrections; its saved summaries have three-decimal precision. Panel (b) reports mean task utility $V(g)$, with Δ_V defined as Local minus SupportKernel. Random probes use the same local predictor with a random rather than diverse probe set. These analyses reuse historical outcomes, unlike the fresh-target WebShop comparison in Section 5.4.

C.3 INTERIM INSPECTION IN THE FRESH-TARGET WEBSHOP STUDY

Predictions and analysis rules were fixed before target rollout. During execution, a descriptive inspection accessed 98 completed correction–target pairs. No corrections, probes, targets, predictor settings, activation rules, baselines, or inference procedures changed after that inspection. The interim results were not used to stop collection. The study was therefore not continuously blinded, and the access was documented in an amendment. The final analysis includes all 200 test pairs, each evaluated with three paired repeats, for 600 paired runs.