FROM UNCERTAINTY TO ACTION: LEARNING TO STEER LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Steering an LLM agent means deciding whether to correct it, at which step, and with which mechanism. Uncertainty is often used to decide when to correct an agent, but whether it can guide these decisions remains unclear. We steer agent trajectories separately at every non-terminal step with each of four mechanisms and run each continuation to completion. The resulting stepwise outcome table (SOT) holds about $82{,}000$ counterfactual continuations of $1{,}864$ trajectories from three benchmarks and two agents. It shows that uncertainty can identify failing trajectories, but that no single signal reliably locates the step at which steering helps. We therefore propose VoS (Value of Steering), a trajectory-level monitor, offline or online, that learns from SOT the value of steering at each step and decides where to steer by it. A harm-budgeted trigger decides whether to steer, limiting the fraction of successful trajectories that VoS disturbs. VoS improves on unmodified execution in all 12 settings of benchmark, agent, and offline or online use, by 7.8 points on average, and outperforms the strongest of five existing uncertainty-triggered methods in 11, by 2.9 points on average. Ablations show that training on measured outcomes and a tight harm budget are both essential.

1 INTRODUCTION

Uncertainty quantification (UQ) for large language models (LLMs) has often centered on answer correctness, hallucination detection, and confidence calibration (Kadavath et al., 2022; Lin et al., 2022; Kuhn et al., 2023; Manakul et al., 2023; Tian et al., 2023; Xiong et al., 2024; Farquhar et al., 2024; Geng et al., 2024), and is typically assessed through error detection, reliability ranking, or calibration metrics (Hendrycks & Gimpel, 2017; Guo et al., 2017; Malinin & Gales, 2021; Vashurin et al., 2025). As LLMs are increasingly deployed as agents that plan, use tools, and interact with environments to complete multi-step tasks (Yao et al., 2023; Schick et al., 2023; Wang et al., 2024c;a), uncertainty estimates should also help improve task execution, not only assess the reliability of model outputs.

Multi-step execution creates opportunities to correct an agent before a task fails, since early errors otherwise propagate through the remaining steps (Cemri et al., 2025; Zhang et al., 2025), and prior work offers mechanisms for doing so, from prompting an agent to reflect on its own progress to supplying external feedback and assistance (Shinn et al., 2023; Madaan et al., 2023; Kim et al., 2025; Gou et al., 2024; Chen et al., 2024; Zhao et al., 2024; Wang et al., 2025). We call a targeted modification of an agent's subsequent execution at a selected point of its trajectory *steering*. Steering at every step adds computation and may disrupt otherwise successful execution (Huang et al., 2024; Kamoi et al., 2024), so it should be used selectively, and uncertainty is a natural signal for this selection, as in selective prediction and deferral to an expert (El-Yaniv & Wiener, 2010; Geifman & El-Yaniv, 2017; Mozannar & Sontag, 2020).

This requires three decisions: *whether* to steer a trajectory, *where*, that is, at which step, and with *which* mechanism. These decisions depend on whether a particular correction would improve the outcome, which the predicted risk of failure alone cannot tell: a likely failure may not be recoverable, a correction to a successful trajectory can introduce new errors, and the most uncertain step need not offer the greatest benefit. We therefore ask: *Can uncertainty guide effective steering of LLM agents?*

Among uncertainty-triggered methods for agents, Agentic UQ (AUQ) (Zhang et al., 2026b) combines confidence-triggered reflection with memory, and ReDAct (Piatrashyn et al., 2026) defers

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.

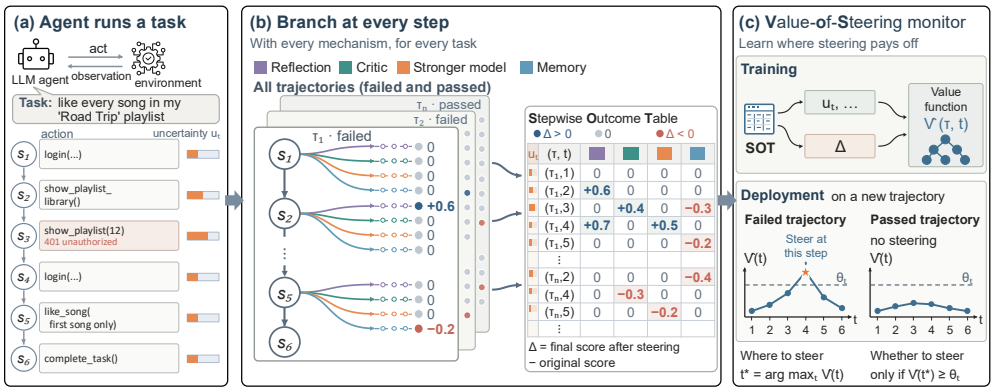


Figure 1: Overview. (a) An agent with step-level uncertainty. (b) SOT: every step steered with every mechanism and run to the end. (c) VoS: where to steer by a value function learned from SOT, whether by a threshold.

uncertain actions to a stronger model. Their aggregate task gains, however, do not isolate the value of choosing the trajectory, the step, and the mechanism. DIAL (Li et al., 2026) and Zhang et al. (2026a) begin to address this with counterfactual comparisons, but only at the states that a trigger selects, whereas deciding where to steer requires comparing the outcomes of steering at different steps of the same trajectory. We therefore introduce a stepwise counterfactual evaluation that steers every step of each trajectory with each of four mechanisms and records the change in the final task score in a stepwise outcome table (SOT), including the choices that a given trigger would not have made. SOT captures both recovery on failed trajectories and harm to successful ones, and it lets us compare predictive metrics with steering gains as well as train and calibrate monitors.

Our results show that uncertainty helps identify failing trajectories but remains an imperfect guide to the value of correction. That accurate failure prediction need not yield effective intervention has been noted for agent oversight in general (Vasudev et al., 2026; Zhang et al., 2026a); our stepwise outcomes show how this gap arises for uncertainty signals: a higher failure-detection AUROC does not yield greater task improvements, and the order in which a signal ranks the steps of a trajectory barely matches their gain. The self-evaluation score $P(\text{True})$ (Kadavath et al., 2022), for example, is among the weakest failure detectors in our study, yet it is the signal most consistently aligned with where steering helps. The usefulness of a signal also depends on the mechanism: the four mechanisms agree moderately on which trajectories are recoverable but barely on which step to steer.

These findings shape our monitor, VoS (Value of Steering). Since uncertainty carries a useful but imperfect failure signal and steering a successful trajectory can only lower its score, *whether* to steer is decided by a harm-budgeted trigger, calibrated to limit the fraction of successful trajectories that are steered; since no single signal reliably locates the step at which steering helps, *where* to steer is decided by a value function learned from all signals, with the measured outcome of steering as the target. VoS runs offline, rolling back a completed execution before its result is accepted, or online, steering during execution; Figure 1 gives an overview.

Our contributions are:

- We present SOT, a stepwise table of about $82{,}000$ counterfactual steering outcomes on $1{,}864$ trajectories, which measures harm to successful trajectories as well as recovery on failed ones. With it, we show that uncertainty helps identify failing trajectories, but that no single signal reliably locates the step at which steering helps or tells which mechanism to use.
- We propose VoS, a trajectory-level, harm-budgeted steering monitor that works during execution or by rolling back afterwards and decides where to steer by the value of steering at each step, estimated from uncertainty signals.
- On three benchmarks with two agents, offline and online, VoS improves on unmodified execution in all 12 settings, by 7.8 points on average, and outperforms the strongest of five

uncertainty-triggered methods from prior work in 11, by 2.9 points on average. Ablations show that measured outcomes are a better training target than failure, binary, or process-reward labels, that a tight harm budget beats steering more trajectories, and that no single family of our 14 estimators suffices across settings.

2 Stepwise Counterfactual Evaluation of Steering

To judge whether, where, and with which mechanism to steer, we measure what steering achieves at every step of every trajectory, not only at the steps that some trigger selects, and evaluate steering policies on the resulting outcomes.

2.1 Setting and Notation

An LLM agent solves a task over multiple steps (Yao et al., 2023; Wang et al., 2024a; Ma et al., 2024; Yao et al., 2025): at each step it reasons about the current state, issues an action such as a code block or a tool call (Schick et al., 2023; Wang et al., 2024c), and receives an observation. We call the unmodified execution of a task its *reference trajectory*, $\tau = (s_1, \ldots, s_N)$, where s_t collects the reasoning, action, and observation of step t, and write $\tau_{\leq t}$ for its first t steps. The benchmark's evaluator assigns it a task score $S(\tau) \in [0, 1]$ and marks it as successful or failed. Each step also carries a vector of uncertainty scores $u_t \in \mathbb{R}^K$, computed from the token distributions of the agent's model when it produced step t and oriented so that higher values indicate greater uncertainty.

2.2 Steering Mechanisms

A steering mechanism m acts at a single step t: given the task instruction and the prefix $\tau_{\leq t}$, it writes one message, or abstains, and the message is inserted into the agent's context before step $t + 1$, after which the agent continues unchanged. The four mechanisms $\mathcal{M}$ that we study share this interface, so they can be compared at the same steps of the same trajectories. **Reflection** follows ReflAct (Kim et al., 2025), a goal-grounded form of self-reflection (Shinn et al., 2023; Madaan et al., 2023): the agent's own model reflects on the current state in relation to the task goal, and the message adds an instruction to reflect in the same way before every later action. **Critic feedback** follows CRITIC (Gou et al., 2024): a critic verifies the trajectory so far in a copy of the environment restored to step t and sends its critique with a corrected next action, or abstains if it finds nothing wrong. **Stronger-model assistance** follows the deferred step of ReDAct (Piatrashyn et al., 2026): a stronger model writes the next action with its reasoning, and the agent executes this action as given. **Memory-based guidance**, in the spirit of experience-based guidance (Shinn et al., 2023; Zhao et al., 2024; Wang et al., 2025), has an advisor read the prefix together with a note of recurring failure patterns, synthesized offline from failed training trajectories, and send a diagnosis and suggested next actions, followed by the note. Only stronger-model assistance calls an external model; Appendix B gives the prompts.

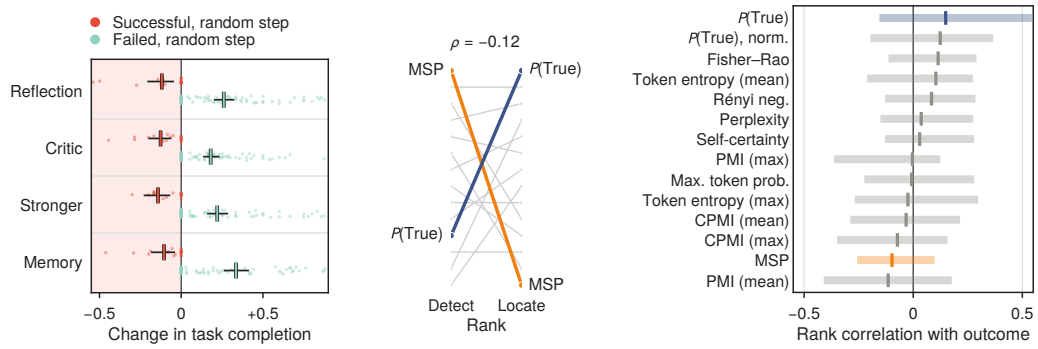
2.3 Measuring Steering Outcomes

We write $\tau^{m,t}$ for the execution that results from steering τ at step t with mechanism m, and define the *steering outcome* as the change in task score,

$$\Delta_m(\tau, t) = S\big(\tau^{m,t}\big) - S(\tau). \tag{1}$$

On a successful reference trajectory, which already attains the full score, the outcome can only be zero or negative, so these trajectories measure the harm that steering can do. To measure an outcome, we replay the recorded steps $1, \ldots, t$, insert the message, and let the agent run to completion; if the mechanism abstains, the agent continues without a message.

We measure the outcome at every non-terminal step $t < N$ of every reference trajectory, failed or successful, and for all four mechanisms. The resulting *stepwise outcome table* (SOT) $\mathcal{D} = \{\Delta_m(\tau, t)\}$ thus also covers the choices that a particular triggering rule would not have made. Each entry is observed by re-executing the agent from the same prefix, not inferred from a causal model or estimated from logged decisions (Dudík et al., 2011; Bottou et al., 2013).



(a) (b) (c)

Figure 2: AppWorld, Gemma; Figure 8 shows Qwen. (a) Change in task completion per trajectory when steered at a random step; ticks: mean with 95% bootstrap interval. Steering lowers the score of successful trajectories with every mechanism. (b) Estimators ranked by failure detection and by within-trajectory localization; ρ: Spearman correlation. The best detectors are not the best locators. (c) Per-trajectory Spearman correlation between each estimator and the outcome of memory-based guidance across steps; bar: interquartile range, tick: median. No estimator orders the steps like their outcomes.

2.4 Steering Policies and Evaluation

In our main setting, a steering policy π steers each trajectory at most once: it either leaves the trajectory alone, $\pi(\tau) = \emptyset$, or selects a step and a mechanism, $\pi(\tau) = (t, m)$ with $t < N$ and $m \in \mathcal{M}$. It thus makes three decisions: *whether* to steer a trajectory, *where* to steer it, and *which* mechanism to use. An *offline* policy sees the complete reference trajectory, but not its score, and rolls back to the step it selects; an *online* policy decides at each step, from the prefix $\tau_{\leq t}$ alone, whether to steer now. A *monitor* is a policy whose decisions depend on the trajectory only through the uncertainty scores and the step index: $u_1, \ldots, u_N$ offline and $u_1, \ldots, u_t$ online.

Evaluation and cost. We evaluate a policy by the task score it attains over all test trajectories, successful and failed, so that harm to successful trajectories counts against recovery on failed ones; the fraction of successful trajectories that it steers is its *false-trigger rate*. The cost $C(\pi)$ counts the additional agent steps executed because of steering, plus the mechanism's own call in agent-step equivalents of its tokens, averaged over the trajectories: online, only the difference to the replaced rest of the reference trajectory counts; offline, every step after the rollback.

2.5 Benchmarks, Agents, and Uncertainty Estimators

Benchmarks and agents. Our main benchmark is AppWorld (Trivedi et al., 2024), in which an agent writes code against the APIs of simulated apps and unit tests check their final state; the task score is one if all tests pass and zero otherwise, as in the task goal completion that we report, and VoS is trained on the fraction of tests passed. In ALFWorld (Shridhar et al., 2021), an agent completes household tasks in a text environment and wins or loses; in WebShop (Yao et al., 2022), it buys a product in a simulated store, scored by its match with the instruction, and succeeds at a score of one. The agents are Gemma-4-31B (Gemma Team, 2026) and Qwen3.6-35B (Yang et al., 2025; Qwen Team, 2026), run as ReAct-style agents (Yao et al., 2023) with open weights, so that token-level uncertainty can be computed at every step. Stronger-model assistance uses DeepSeek-V4-Flash (DeepSeek-AI, 2026).

Uncertainty estimators. At every step we compute 14 estimators as defined in LM-Polygraph (Fadeeva et al., 2023; Vashurin et al., 2025) with minor adaptations: sequence- and token-level probabilities (Fomicheva et al., 2020; Malinin & Gales, 2021), entropy-based and information-geometric measures of the token distributions (Malinin & Gales, 2021; Darrin et al., 2023; Kang et al., 2025), pointwise mutual information (Takayama & Arase, 2019; van der Poel et al., 2022), and self-evaluation with $P(\text{True})$ (Kadavath et al., 2022); Appendix C lists them. All come from the token distributions of the step that the agent produced, with at most a few additional scoring passes,

whose cost Table 9 reports; we exclude sampling-based estimators (Kuhn et al., 2023; Farquhar et al., 2024; Manakul et al., 2023; Lin et al., 2024), which need several generations at every step, and density-based ones (Ren et al., 2023b), which would be fitted and scored on the same data.

3 What Uncertainty Tells a Steering Policy

We now use SOT to ask what uncertainty contributes to each of the three decisions, using each signal on its own as a probe, on completed AppWorld trajectories.

3.1 Whether: Uncertainty Identifies Failing Trajectories

Uncertainty carries a trajectory-level failure signal. A classifier on summary statistics of the 14 estimators over a completed trajectory detects failed trajectories with an AUROC of up to 0.73 on Gemma and 0.71 on Qwen, in line with earlier reports that uncertainty predicts the failure of agents (Zhao et al., 2025; Duan et al., 2025; Barbi et al., 2025; Zhang et al., 2026b), so a monitor can use it to decide whether to steer. The signal is imperfect, however, and steering is not safe by default: on successful trajectories, steering at a random step lowers the score with every mechanism, on average by up to 0.14 on Gemma and 0.32 on Qwen, as Figures 2a and 8a show, so a monitor must also control how often it steers successful trajectories.

3.2 Where: The Same Signal Plays Different Roles

Detecting failure and locating the step are different abilities. Figures 2b and 8b rank the signals by two abilities: detection, how well the mean of a signal over a trajectory separates failed from successful trajectories, and localization, how well it ranks the helpful steps of a failed trajectory first, measured by a within-trajectory AUROC for each mechanism. The two rankings are unrelated or even opposed, with a rank correlation of -0.12 on Gemma and -0.41 on Qwen. $P(\mathrm{True})$ (Kadavath et al., 2022) is among the weakest failure detectors on Gemma, and its detection direction is reversed on Qwen (0.44), yet it is the signal most consistently aligned with where steering helps, with a within-trajectory AUROC above 0.5 for all four mechanisms on both agents.

Uncertainty orders the steps differently from their gain. To indicate where to steer, a signal should rank first the step at which steering helps most. The target is the outcome of steering at a step, not the correctness of the step, which is what locating the erroneous step of a solution or of a failed execution asks for (Lightman et al., 2024; Wang et al., 2024b; Zhang et al., 2025). We compare, within each failed trajectory, the order of the steps by a signal with their order by measured outcome, without training any model. The two orders are nearly unrelated, as Figures 2c and 8c show: for memory-based guidance, the median rank correlation per trajectory is highest for $P(\mathrm{True})$, at only $+0.15$ on both agents, and for every estimator it is negative on at least a quarter of the trajectories. The signal still carries some information: the step that $P(\mathrm{True})$ ranks first is the best step of the trajectory more often than a random step, in all eight combinations of agent and mechanism. **The value of a signal therefore depends on the decision it serves**: a policy can neither choose a signal by how well it detects failures nor simply steer at the step that a signal ranks first. This argues for learning from all signals jointly, with the steering outcome itself as the target.

3.3 Which: Mechanisms Disagree on Where to Steer

The four mechanisms agree moderately on which trajectories are recoverable, but barely on where to steer them. Across trajectories, their best outcomes correlate at 0.17 to 0.66; within a trajectory, their orders of the steps correlate at only 0.02 to 0.19, as Figure 9 in the appendix shows. A signal can even relate to the outcome with opposite signs for different mechanisms, as token entropy does on Gemma for memory-based guidance and stronger-model assistance, a reversal of the kind that Li et al. (2026) report across environments and models for the value of test-time compute.

4 Steering Monitors

The most direct way to turn uncertainty into a steering policy is to threshold the signal that best detects failures, as uncertainty-triggered methods commonly do (Jiang et al., 2023; Varshney et al.,

Table 1: Main results on each benchmark's own metric in % (task goal completion on AppWorld, success rate on ALFWorld, score on WebShop), as mean ± standard deviation. Gray: references, not ranked; bold and underline: best and second-best policy in each row.

		References		Steering policies					
		Unmod.	Always	AUQ	DIAL	Barbi et al.	DS-MCM	ReDAct	VoS (ours)
Offline									
AppWorld	Gemma	69.0	71.6 ±1.5	68.1 ±1.0	68.9 ±0.3	69.7 ±1.2	<u>70.4</u> ±0.9	69.3 ±0.7	**73.4** ±1.9
	Qwen	65.5	55.6 ±1.6	65.1 ±0.9	65.4 ±0.4	65.3 ±0.7	64.8 ±0.6	<u>66.0</u> ±0.6	**66.5** ±0.8
ALFWorld	Gemma	74.5	76.6 ±2.8	74.7 ±1.2	<u>80.1</u> ±1.8	80.0 ±1.8	78.4 ±1.7	76.1 ±1.0	**85.2** ±1.6
	Qwen	52.5	69.0 ±2.8	62.4 ±2.5	62.0 ±3.5	61.0 ±2.4	57.0 ±1.6	<u>64.9</u> ±2.7	**78.4** ±3.2
WebShop	Gemma	66.7	68.7 ±0.9	67.1 ±0.3	68.3 ±0.5	<u>69.4</u> ±0.2	67.6 ±0.3	68.3 ±0.7	**72.3** ±1.1
	Qwen	62.5	63.9 ±0.7	61.2 ±0.2	63.3 ±0.6	63.6 ±0.4	<u>65.0</u> ±0.0	63.2 ±0.0	**67.1** ±0.9
Online									
AppWorld	Gemma	69.0	71.6 ±1.5	69.2 ±1.0	70.0 ±1.7	69.7 ±0.8	<u>71.2</u> ±0.9	69.4 ±1.1	**73.6** ±1.8
	Qwen	65.5	55.6 ±1.6	65.1 ±1.5	<u>65.3</u> ±0.7	<u>65.3</u> ±0.8	64.7 ±1.3	64.3 ±0.7	**66.6** ±0.4
ALFWorld	Gemma	74.5	76.6 ±2.8	74.9 ±1.2	<u>82.0</u> ±2.7	81.3 ±1.2	77.4 ±2.5	76.1 ±1.2	**82.6** ±2.2
	Qwen	52.5	69.0 ±2.8	68.2 ±2.7	67.7 ±5.3	<u>70.1</u> ±1.7	62.3 ±2.1	69.4 ±2.3	**71.1** ±2.6
WebShop	Gemma	66.7	68.7 ±0.9	67.7 ±0.3	69.0 ±0.4	<u>70.1</u> ±1.0	67.5 ±0.2	68.6 ±0.5	**72.6** ±0.4
	Qwen	62.5	63.9 ±0.7	61.4 ±0.3	**66.0** ±0.8	64.7 ±1.8	64.0 ±0.6	64.2 ±0.0	<u>65.6</u> ±0.8

2023; Han et al., 2024; Barbi et al., 2025; Zhang et al., 2026b; Sun et al., 2026; Piatrashyn et al., 2026). Section 3 shows why this falls short: such a trigger also fires on successful trajectories, and the best detector is not the best locator. VoS therefore decides the two questions separately: a *steering value function*, learned from SOT, estimates what steering would achieve at each step, its value of computation in the sense of rational metareasoning (Russell & Wefald, 1991), and decides where to steer; a *harm-budgeted trigger* decides whether. Unlike the controllers of Zhang et al. (2026a) and DIAL (Li et al., 2026), which decide at each state in isolation, VoS compares all steps of a trajectory offline and limits how often successful trajectories are disturbed; unlike DIAL, it learns from the size of the effect rather than from a binary label.

4.1 Steering Value and Harm-Budgeted Triggering

The value of steering τ at step t with mechanism m is how much steering there is expected to change the final task score, given the uncertainty scores that the monitor observes: $V_m(\tau, t) = \mathrm{E}\left[\Delta_m(\tau, t) \mid u_1, \ldots, u_T\right]$, with $T = N$ offline and $T = t$ online. It is the advantage of steering over not steering (Sutton & Barto, 2018). The monitor steers at the step of highest estimated value $\hat{V}_m$ if this value reaches a threshold θ:

$$\pi(\tau) = \begin{cases} (t^\star, m) & \text{if } \hat{V}_m(\tau, t^\star) \geq \theta, \\ \emptyset & \text{otherwise,} \end{cases} \qquad t^\star = \arg\max_{t<N} \hat{V}_m(\tau, t). \tag{2}$$

Online, where the estimates arrive one step at a time, it steers at the first step with $\hat{V}_m(\tau, t) \geq \theta$.

Steering value function. Because $\mathcal{D}$ measures the outcome of every step and mechanism, not only of the steps that some trigger selected, we fit $\hat{V}_m = f_m(x_t)$ by regression on $\mathcal{D}$ without off-policy correction (Dudík et al., 2011; Bottou et al., 2013). Failure labels could not supervise it: they do not record where, or whether, steering would have changed the result, and Section 3.2 shows that the relation of a signal to failure differs from its relation to the value of steering. The features x_t describe every signal relative to the trajectory offline and to the prefix online, together with the step index, so that the regression can use each signal both to tell trajectories apart and to tell their steps apart; Appendix C gives the details. Since the mechanisms disagree on where to steer, as Section 3.3 shows, f_m is fitted per mechanism; VoS uses memory-based guidance, the most robust mechanism on both agents in Figures 2a and 8a, and we test below whether choosing the mechanism as well improves on it.

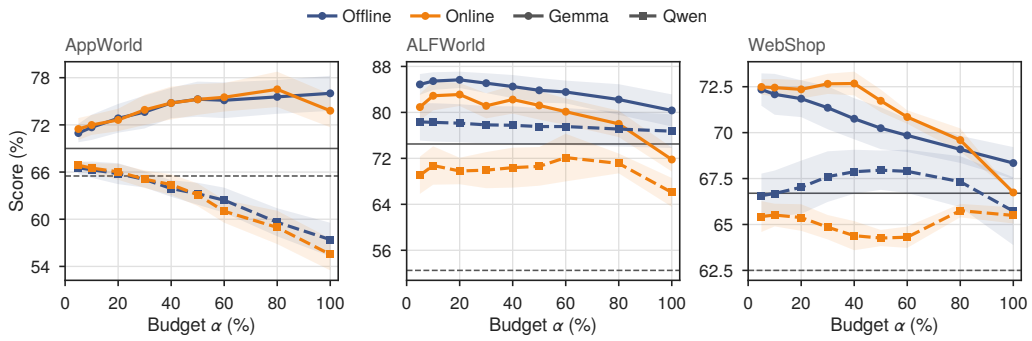


Figure 3: Final score of VoS against the budget α, on each benchmark's own metric in %, for Gemma (solid) and Qwen (dashed). Grey lines: unmodified execution; bands: ± 1 standard deviation over the splits.

Harm-budgeted triggering. Section 3.1 shows that uncertainty detects failing trajectories only imperfectly and that every trigger on a successful trajectory can only lower its score, so the monitor limits harm in two ways. First, f_m is trained on successful as well as failed trajectories, so that their negative outcomes are part of its target. Second, a budget α on the false-trigger rate sets the threshold, a criterion of the Neyman–Pearson type (Scott & Nowak, 2005): the monitor should recover as much as possible on failed trajectories while steering at most a fraction α of successful ones. θ_α is the $(1-\alpha)$ quantile of the largest out-of-fold estimate $\max_t \hat{V}_m(\tau, t)$ of the successful training trajectories (Angelopoulos et al., 2024), and α is chosen between 5% and 30% as the budget with the highest task score on the training trajectories, as detailed in Appendix C. The budget is calibrated, not guaranteed: the test trajectories are scored by a model refitted on all training trajectories, not by the out-of-fold models that set θ_α.

4.2 Main Results

Policies and protocol. We compare with five uncertainty-triggered methods: AUQ (Zhang et al., 2026b) thresholds self-evaluated confidence, DIAL (Li et al., 2026) and Barbi et al. (2025) learn from the step index and token entropy, DS-MCM (Sun et al., 2026) flags deviating reasoning entropy, and ReDAct (Piatrashyn et al., 2026) thresholds perplexity. We re-implement only their triggers on SOT, as Appendix D details, so that all methods decide on the same trajectories. As a reference, *always steer* steers every trajectory once, at a step chosen on the training half. All policies are evaluated on the ten splits of Appendix A.

VoS leads where steering every trajectory is a gamble. VoS has the highest score in 11 of the 12 settings in Table 1, also when every method steers with memory-based guidance in Table 5, and remains the best policy in 10 in Table 8, which nets out the change that re-running from the same step already produces. Always steering is a gamble: on AppWorld it helps Gemma but costs Qwen 9.9 points, whereas VoS gains more on Gemma and does not lose in any setting. Steering almost every trajectory also hurts VoS itself: raising its budget to 100% lowers its gain in nine of the twelve settings in Figure 3. **Whether to steer therefore matters as much as where**, and the balance between recovery and harm differs between settings, which is why the budget is chosen on training trajectories.

The gain comes from uncertainty, used through a learned value function. On AppWorld, DIAL and Barbi et al. (2025), which see only the step index and token entropy, gain at most 1.0 point, so the gain of VoS comes from the broader set of uncertainty signals. Thresholding a single signal does not suffice either: AUQ, DS-MCM, and ReDAct gain at most half as much as VoS on both agents, although AUQ thresholds self-evaluation, the strongest family in Figure 4.

Routing does not improve on a fixed mechanism. In Table 6, choosing at every step the mechanism of highest estimated value does not improve on memory-based guidance alone, and monitors on any other single mechanism are worse: uncertainty indicates which trajectories to steer, but not which mechanism to use at a step.

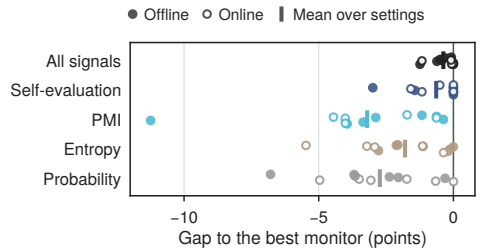


Figure 4: Gap of each monitor, trained on all signals or on one family, to the best of the five in each of the twelve settings.

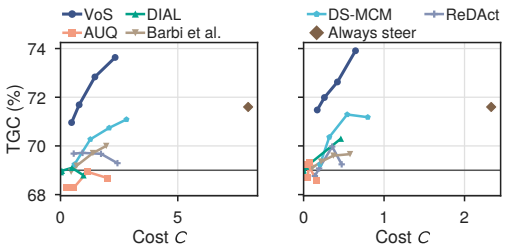


Figure 5: Task goal completion against cost for budgets α from 5% to 30% (AppWorld, Gemma; line: unmodified execution).

4.3 What the Monitor Relies On

No single family is reliable; learning from all of them is. Among the monitors trained on one family alone in Figure 4, self-evaluation is the best single family in nine of the twelve settings but not in the other three, and the token probabilities that uncertainty-triggered methods commonly use fall short of the best monitor by 2.7 points on average. The monitor on all fourteen estimators is never more than 1.2 points below the best one. A black-box signal is no substitute: the confidence that the agent verbalizes when asked (Tian et al., 2023), the signal behind AUQ, is 1.0 at 86% of Gemma's steps, so a monitor on it alone gains at most about a third of what VoS gains on AppWorld. Which uncertainty signal carries the value of steering depends on the benchmark and the agent, so the monitor learns from all of them.

Supervision from prior work. Table 2 exchanges only the training target of the monitor for the supervision of prior work: trajectory failure (Barbi et al., 2025; Zhang et al., 2026b), DIAL's binary label of whether steering helps (Li et al., 2026), step correctness judged in hindsight (Lightman et al., 2024; Zhang et al., 2025), and the Monte-Carlo value of continuing without intervention (Wang et al., 2024b). None matches the measured advantage offline: failure labels recover least, the binary label misses the size of the effect, and step correctness selects the wrong step, since the judged first error lies in the last third of a recoverable trajectory and the most helpful steps in the middle, and steering at a step judged incorrect even gains less than at one judged correct. Locating the error is not the same problem as locating the step at which steering helps, as Section 3.2 anticipated.

Table 2: VoS trained on the targets of prior work: task goal completion in % on AppWorld with Gemma.

Target	Offline	Online
Failure label	69.3	72.5
DIAL label	70.8	72.1
PRM, labelled	71.6	72.6
PRM, MC rollout	71.4	72.1
Ours	**73.4**	**73.6**

4.4 What Steering Costs

VoS buys more gain for less. Over the range of cost that its curve spans in Figure 5, VoS lies above every method from prior work, which gain almost nothing at lower cost, and at less than a third of the cost it exceeds the best score that any of them attains at *any* budget. Counting tokens gives the same picture in Figure 7, and the mechanism's own call, which C includes, is at most 11% of a steering event on Gemma and changes no ordering among the methods in Table 7.

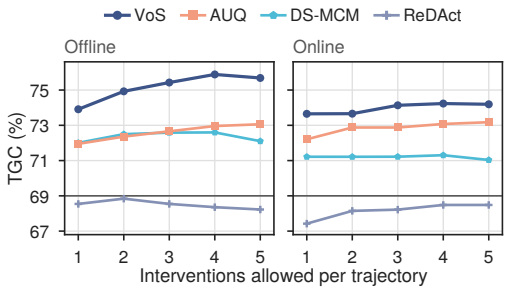


Figure 6: Task goal completion against the number of steering events allowed, in real runs (AppWorld, Gemma; line: unmodified execution).

Steering more than once adds modestly, and VoS stays ahead. In real runs with up to five steering events per trajectory, shown in Figure 6,

a single event already yields at least 70% of the best gain of every method that gains. **VoS is the best at every number of steering events, so restricting the evaluation to one event per trajectory does not favor it.**

5 Related Work

Uncertainty quantification and uncertainty as a trigger. Uncertainty estimators for LLMs score a generation by its token probabilities (Fomicheva et al., 2020; Malinin & Gales, 2021), by the consistency of sampled generations (Kuhn et al., 2023; Farquhar et al., 2024; Manakul et al., 2023; Lin et al., 2024), by its internal representations (Ren et al., 2023b), or by the model's own confidence, verbalized or scored (Lin et al., 2022; Tian et al., 2023; Xiong et al., 2024; Kadavath et al., 2022), and benchmarks such as LM-Polygraph evaluate them by how well they detect errors (Fadeeva et al., 2023; Vashurin et al., 2025). Uncertainty also triggers retrieval or verification during generation (Jiang et al., 2023; Varshney et al., 2023; Su et al., 2024), tool calls (Han et al., 2024), and requests for help (Ren et al., 2023a) or clarification (Hu et al., 2024; Suri et al., 2026). In agents, where it propagates across steps (Zhao et al., 2025; Duan et al., 2025), it triggers communication resets (Barbi et al., 2025), reflection (Zhang et al., 2026b; Sun et al., 2026), and deferral to a stronger model (Piatrashyn et al., 2026), and these methods are evaluated by the gain of the whole pipeline. We evaluate 14 such estimators, from four families, by what each contributes to the decisions of whether, where, and with which mechanism to steer, measured at every step.

Steering mechanisms and deferral. An agent can be corrected by self-reflection (Shinn et al., 2023; Madaan et al., 2023; Kim et al., 2025), tool-based critique (Gou et al., 2024), or past experience (Zhao et al., 2024; Wang et al., 2025), although self-correction without reliable feedback can degrade results (Huang et al., 2024; Kamoi et al., 2024); a query can instead be deferred to a stronger model (Chen et al., 2024; Yue et al., 2024; Aggarwal et al., 2024; Ong et al., 2025; Mozannar & Sontag, 2020) or withheld, as in selective prediction (El-Yaniv & Wiener, 2010; Geifman & El-Yaniv, 2017). The four mechanisms of Section 2.2 follow these works.

Deciding when to intervene. Accurate failure prediction does not by itself make intervention effective (Vasudev et al., 2026; Zhang et al., 2026a), and Section 3 shows why for uncertainty. DIAL (Li et al., 2026) and Zhang et al. (2026a) judge interventions by their counterfactual outcomes, but only at the states that a trigger selects, whereas SOT measures them at every step, so that a monitor can compare the steps of a trajectory. Process reward models (Lightman et al., 2024; Wang et al., 2024b) and failure attribution (Zhang et al., 2025) locate the erroneous step instead, which, as Section 4.3 shows, need not be the step at which steering helps most. Off-policy evaluation estimates counterfactual outcomes from logged decisions (Dudík et al., 2011; Bottou et al., 2013); SOT observes them by re-executing the agent.

6 Conclusion

We asked whether uncertainty can guide effective steering of LLM agents, and measured in SOT what steering achieves at every step of every trajectory with four mechanisms. Uncertainty identifies failing trajectories, but used directly, today's estimators are an inefficient guide to multi-step steering: they were designed to detect errors in an output, and no single one of them reliably locates the step at which steering helps or tells which mechanism to use. In our experiments, their value for steering is realized by learning from measured outcomes rather than by thresholding: VoS learns a value function over all signals from SOT, pairs it with a harm-budgeted trigger, and improves on unmodified execution in all 12 settings. This learning, however, needs counterfactual outcomes for each benchmark, agent, and mechanism. A natural next step is uncertainty quantification designed for steering multi-step agents, with estimators whose target is the value of steering at a step rather than the correctness of an output, so that a monitor needs less counterfactual supervision.

Limitations. SOT is expensive to build, since it re-runs the agent from every step with every mechanism. Our estimators need the agent's token distributions, so we study only open-weight agents of moderate size; black-box signals such as verbalized confidence are much less effective.

AI USE STATEMENT

We used generative AI tools to aid and polish the writing of this paper and for retrieval of related work. We did not use them to formulate research questions, design experiments, or analyze data. The authors reviewed all AI-assisted work: every edited passage was checked against its intended meaning, and every retrieved paper was read and its citation verified. We take responsibility for the final content of this work.

REFERENCES

Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, Shyam Upadhyay, Manaal Faruqui, and Mausam. AutoMix: Automatically mixing language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2310.12963.

Anastasios N. Angelopoulos, Stephen Bates, Adam Fisch, Lihua Lei, and Tal Schuster. Conformal risk control. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2208.02814.

Ohav Barbi, Ori Yoran, and Mor Geva. Preventing rogue agents improves multi-agent collaboration. In *Proceedings of the 1st Workshop for Research on Agent Language Models (REALM 2025)*, pp. 486–511, Vienna, Austria, July 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.realm-1.34/.

Léon Bottou, Jonas Peters, Joaquin Quiñonero-Candela, Denis X. Charles, D. Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. Counterfactual reasoning and learning systems: The example of computational advertising. *Journal of Machine Learning Research*, 14: 3207–3260, 2013.

Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Why do multi-agent LLM systems fail? In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, volume 38, 2025. URL https://arxiv.org/abs/2503.13657.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=cSimKw5p6R.

Maxime Darrin, Pablo Piantanida, and Pierre Colombo. RainProof: An umbrella to shield text generator from out-of-distribution data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5831–5857, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.357. URL https://aclanthology.org/2023.emnlp-main.357/.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026. URL https://arxiv.org/abs/2606.19348.

Jinhao Duan, James Diffenderfer, Sandeep Madireddy, Tianlong Chen, Bhavya Kailkhura, and Kaidi Xu. UProp: Investigating the uncertainty propagation of LLMs in multi-step agentic decision-making. *arXiv preprint arXiv:2506.17419*, 2025. URL https://arxiv.org/abs/2506.17419.

Miroslav Dudík, John Langford, and Lihong Li. Doubly robust policy evaluation and learning. In *International Conference on Machine Learning*, 2011.

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

Ekaterina Fadeeva, Roman Vashurin, Akim Tsvigun, Artem Vazhentsev, Sergey Petrakov, Kirill Fedyanin, Daniil Vasilev, Elizaveta Goncharova, Alexander Panchenko, Maxim Panov, Timothy Baldwin, and Artem Shelmanov. LM-Polygraph: Uncertainty estimation for language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 446–461, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-demo.41. URL https://aclanthology.org/2023.emnlp-demo.41/.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630, 2024. doi: 10.1038/s41586-024-07421-0. URL https://www.nature.com/articles/s41586-024-07421-0.

Marina Fomicheva, Shuo Sun, Lisa Yankovskaya, Frédéric Blain, Francisco Guzmán, Mark Fishel, Nikolaos Aletras, Vishrav Chaudhary, and Lucia Specia. Unsupervised quality estimation for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8: 539–555, 2020. doi: 10.1162/tacl_a_00330. URL https://aclanthology.org/2020.tacl-1.35/.

Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://arxiv.org/abs/1705.08500.

Gemma Team. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026. URL https://arxiv.org/abs/2607.02770.

Jiahui Geng, Fengyu Cai, Yuxia Wang, Heinz Koeppl, Preslav Nakov, and Iryna Gurevych. A survey of confidence estimation and calibration in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 6577–6595, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.366. URL https://aclanthology.org/2024.naacl-long.366/.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Sx038qxjek.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, 2017. URL https://arxiv.org/abs/1706.04599.

Jiuzhou Han, Wray Buntine, and Ehsan Shareghi. Towards uncertainty-aware language agent. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 6662–6685, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.398. URL https://aclanthology.org/2024.findings-acl.398/.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1610.02136.

Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

Zhiyuan Hu, Chumin Liu, Xidong Feng, Yilun Zhao, See-Kiong Ng, Anh Tuan Luu, Junxian He, Pang Wei Koh, and Bryan Hooi. Uncertainty of thoughts: Uncertainty-aware planning enhances information seeking in LLMs. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2402.03271.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.01798.

Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7969–7992, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.495. URL https://aclanthology.org/2023.emnlp-main.495/.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022. URL https://arxiv.org/abs/2207.05221.

Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. When can LLMs actually correct their own mistakes? A critical survey of self-correction of LLMs. *Transactions of the Association for Computational Linguistics*, 12:1417–1440, 2024. doi: 10.1162/tacl_a_00713. URL https://aclanthology.org/2024.tacl-1.78/.

Zhewei Kang, Xuandong Zhao, and Dawn Song. Scalable best-of-N selection for large language models via self-certainty. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://arxiv.org/abs/2502.18581.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Jeonghye Kim, Sojeong Rhee, Minbeom Kim, Dohyung Kim, Sangmook Lee, Youngchul Sung, and Kyomin Jung. ReflAct: World-grounded decision making in LLM agents via goal-state reflection. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 33433–33465, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.1697. URL https://aclanthology.org/2025.emnlp-main.1697/.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=VD-AYtP0dve.

Ziming Li, Jiatan Huang, Xiaoguang Guo, Guilin Wang, and Chuxu Zhang. Same signal, opposite meaning: Direction-informed adaptive learning for LLM agents. *arXiv preprint arXiv:2605.06908*, 2026. URL https://arxiv.org/abs/2605.06908.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2305.20050.

Stephanie Lin, Jacob Hilton, and Owain Evans. Teaching models to express their uncertainty in words. *Transactions on Machine Learning Research*, 2022. URL https://arxiv.org/abs/2205.14334.

Zhen Lin, Shubhendu Trivedi, and Jimeng Sun. Generating with confidence: Uncertainty quantification for black-box large language models. *Transactions on Machine Learning Research*, 2024. URL https://arxiv.org/abs/2305.19187.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. AgentBoard: An analytical evaluation board of multi-turn LLM agents. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, volume 37, 2024. URL https://arxiv.org/abs/2401.13178.

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. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2303.17651.

Andrey Malinin and Mark Gales. Uncertainty estimation in autoregressive structured prediction. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2002.07650.

Potsawee Manakul, Adian Liusie, and Mark Gales. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 9004–9017, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.557. URL https://aclanthology.org/2023.emnlp-main.557/.

Hussein Mozannar and David Sontag. Consistent estimators for learning to defer to an expert. In *International Conference on Machine Learning*, 2020. URL https://arxiv.org/abs/2006.01862.

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. URL https://arxiv.org/abs/2406.18665.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830, 2011.

Dzianis Piatrashyn, Nikita Kotelevskii, Kirill Grishchenkov, Nikita Glazkov, Ivan Nasonov, Ilya Makarov, Timothy Baldwin, Preslav Nakov, Roman Vashurin, and Maxim Panov. ReDAct: Uncertainty-aware deferral for LLM agents. *arXiv preprint arXiv:2604.07036*, 2026. URL https://arxiv.org/abs/2604.07036.

Qwen Team. Qwen3.6-35B-A3B: Agentic coding power, now open to all. https://qwen.ai/blog?id=qwen3.6-35b-a3b, 2026.

Allen Z. Ren, Anushri Dixit, Alexandra Bodrova, Sumeet Singh, Stephen Tu, Noah Brown, Peng Xu, Leila Takayama, Fei Xia, Jake Varley, Zhenjia Xu, Dorsa Sadigh, Andy Zeng, and Anirudha Majumdar. Robots that ask for help: Uncertainty alignment for large language model planners. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 661–682. PMLR, 2023a. URL https://proceedings.mlr.press/v229/ren23a.html.

Jie Ren, Jiaming Luo, Yao Zhao, Kundan Krishna, Mohammad Saleh, Balaji Lakshminarayanan, and Peter J. Liu. Out-of-distribution detection and selective generation for conditional language models. In *International Conference on Learning Representations*, 2023b. URL https://arxiv.org/abs/2209.15558.

Stuart Russell and Eric Wefald. Principles of metareasoning. *Artificial Intelligence*, 49(1–3):361–395, 1991.

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, 2023. URL https://arxiv.org/abs/2302.04761.

Clayton Scott and Robert Nowak. A Neyman–Pearson approach to statistical learning. *IEEE Transactions on Information Theory*, 51(11):3806–3819, 2005.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-0377. URL https://papers.nips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.03768.

Weihang Su, Yichen Tang, Qingyao Ai, Zhijing Wu, and Yiqun Liu. DRAGIN: Dynamic retrieval augmented generation based on the real-time information needs of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12991–13013, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.702. URL https://aclanthology.org/2024.acl-long.702/.

Zhongxiang Sun, Qipeng Wang, Weijie Yu, Jingxuan Yang, Haolang Lu, and Jun Xu. Deep search with hierarchical meta-cognitive monitoring inspired by cognitive neuroscience. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1698–1708. Association for Computing Machinery, 2026. doi: 10.1145/3805712.3809549. URL https://dl.acm.org/doi/10.1145/3805712.3809549.

Manan Suri, Puneet Mathur, Nedim Lipka, Franck Dernoncourt, Ryan A. Rossi, and Dinesh Manocha. Structured uncertainty guided clarification for LLM agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 40811–40838, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.2028. URL https://aclanthology.org/2026.findings-acl.2028/.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, second edition, 2018.

Junya Takayama and Yuki Arase. Relevant and informative response generation using pointwise mutual information. In *Proceedings of the First Workshop on NLP for Conversational AI*, pp. 133–138, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4115. URL https://aclanthology.org/W19-4115/.

Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher Manning. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5433–5442, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.330. URL https://aclanthology.org/2023.emnlp-main.330/.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.850. URL https://aclanthology.org/2024.acl-long.850/.

Liam van der Poel, Ryan Cotterell, and Clara Meister. Mutual information alleviates hallucinations in abstractive summarization. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 5956–5965, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.399. URL https://aclanthology.org/2022.emnlp-main.399/.

Neeraj Varshney, Wenlin Yao, Hongming Zhang, Jianshu Chen, and Dong Yu. A stitch in time saves nine: Detecting and mitigating hallucinations of LLMs by validating low-confidence generation. *arXiv preprint arXiv:2307.03987*, 2023. URL https://arxiv.org/abs/2307.03987.

Roman Vashurin, Ekaterina Fadeeva, Artem Vazhentsev, Lyudmila Rvanova, Daniil Vasilev, Akim Tsvigun, Sergey Petrakov, Rui Xing, Abdelrahman Sadallah, Kirill Grishchenkov, Alexander Panchenko, Timothy Baldwin, Preslav Nakov, Maxim Panov, and Artem Shelmanov. Benchmarking uncertainty quantification methods for large language models with LM-Polygraph. *Transactions of the Association for Computational Linguistics*, 13:220–248, 2025. doi: 10.1162/tacl_a_00737. URL https://aclanthology.org/2025.tacl-1.11/.

Rakshith Vasudev, Melisa Russak, Dan Bikel, and Waseem Alshikh. Accurate failure prediction in agents does not imply effective failure prevention. *arXiv preprint arXiv:2602.03338*, 2026. URL https://arxiv.org/abs/2602.03338.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6): 186345, 2024a. doi: 10.1007/s11704-024-40231-1. URL https://doi.org/10.1007/s11704-024-40231-1.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, Bangkok, Thailand, August 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.510. URL https://aclanthology.org/2024.acl-long.510/.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better LLM agents. In *International Conference on Machine Learning*, 2024c. URL https://arxiv.org/abs/2402.01030.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *International Conference on Machine Learning*, 2025. URL https://arxiv.org/abs/2409.07429.

Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. Can LLMs express their uncertainty? An empirical evaluation of confidence elicitation in LLMs. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2306.13063.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://arxiv.org/abs/2207.01206.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03629.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2406.12045.

Murong Yue, Jie Zhao, Min Zhang, Liang Du, and Ziyu Yao. Large language model cascades with mixture of thought representations for cost-efficient reasoning. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.03094.

Chubin Zhang, Zhenglin Wan, Xingrui Yu, Jingxuan Wu, Qi Wen, Pengfei Zhou, Wangbo Zhao, and Ivor Tsang. Calibration is not control: Why LLM-agent oversight needs intervention. *arXiv preprint arXiv:2606.21399*, 2026a. URL https://arxiv.org/abs/2606.21399.

Jiaxin Zhang, Prafulla Kumar Choubey, Kung-Hsiang Huang, Caiming Xiong, and Chien-Sheng Wu. Agentic uncertainty quantification. *arXiv preprint arXiv:2601.15703*, 2026b. URL https://arxiv.org/abs/2601.15703.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and Qingyun Wu. Which agent causes task failures and when? On automated failure attribution of LLM multi-agent systems. In *International Conference on Machine Learning*, 2025. URL https://arxiv.org/abs/2505.00212.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024. doi: 10.1609/aaai.v38i17.29936. URL https://ojs.aaai.org/index.php/AAAI/article/view/29936.

Qiwei Zhao, Dong Li, Yanchi Liu, Wei Cheng, Yiyou Sun, Mika Oishi, Takao Osaki, Katsushi Matsuda, Huaxiu Yao, Chen Zhao, Haifeng Chen, and Xujiang Zhao. Uncertainty propagation on LLM agent. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6064–6073, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.302. URL https://aclanthology.org/2025.acl-long.302/.

A Experimental Setup

Table 3 lists the benchmarks and the settings of the agents.

Table 3: Benchmarks and agent settings.

	AppWorld	ALFWorld	WebShop
Tasks	258	274	400
Note built from	failed train tasks	150 train games	failed train goals
Action	one Python code block calling app APIs	one text command	`search[...]` or `click[...]`
Decoding	temperature 0, seed 100		

Splits and training data. Steering policies are trained and evaluated on SOT, and everything they need—models, thresholds, and the budget α—is chosen on training data only. On AppWorld and ALFWorld, the table covers the evaluated tasks only, and every policy is trained and evaluated on ten random splits of these tasks into halves, stratified by failure. On WebShop, it also covers the 300 training goals, so a policy is trained on them and evaluated on the 100 held-out goals, with deviations over ten draws of 250 training goals. The memory note is synthesized from failed reference trajectories of each benchmark's training split, never from an evaluated task; on AppWorld and ALFWorld, these tasks carry no outcomes and are disjoint from both halves.

B Steering Mechanisms

The boxes give the prompts of the four mechanisms on AppWorld, abridged ([...]): the prompt to the model that writes the message, and the message inserted into the agent's context.

Reflection

Prompt to the agent's model:

```
Your task: ⟨instruction⟩
What you have done so far, most recent last: ⟨last 40 steps⟩
Reflect, in the first person and in at most four sentences, on your current state in
relation to the task goal. (1) State what the most recent output showed [...]. (2) List
what the task goal requires, item by item, and for each item say whether it is confirmed by
an output above, held only as an assumption, or not yet obtained. [...] (4) End by naming
the single first gap to close next. [...] Do not write code. Begin your reply with
"Reflection:" and write nothing else.
```

Message to the agent:

```
Pause. Below is your reflection on your current state in relation to the task goal,
followed by how to reason for the rest of this task.
Reflection: ⟨reflection⟩
Your next code block must act on the first gap this reflection names [...]. From now on, at
every turn, before writing any code, first reflect on your current state in relation to the
task goal, in at most four sentences, and only then write the code for this turn. [...]
```

Critic feedback

Brief to the critic, an agent in a copy of the environment at step t:

```
STOP pursuing the original task. New standing instruction: verify the work above before it
goes any further. [...] What's the problem with the trajectory above? Check it against the
live app state with the tools before you judge it [...]. Take as many steps as you need,
but only read; change nothing. [...] Then write a plain-prose critique [...]: 1.
Requirements [...] 2. Truthfulness [...] 3. Blocker [...] 4. Problem [...] Finish with
exactly one line: VERDICT: correct [...], or VERDICT: incorrect otherwise. If the verdict
is incorrect, then add the line "Here's a better next step:" and exactly one code block
[...].
```

Message to the agent:

```
Your work so far has been checked against the live app state. The critique and a corrected
next step are below. Read the critique, run the corrected step as your next action, then
continue working on the original task.
```

```
[Critique]: ⟨critique⟩
[Corrected next step]: ⟨code⟩
```

Stronger-model assistance

Prompt to DeepSeek-V4-Flash (a second call turns the reasoning into one code block):

```
TASK DESCRIPTION: ⟨instruction⟩
ENVIRONMENT HISTORY: ⟨last 40 steps⟩
AVAILABLE ACTIONS: ⟨one Python code block that calls app APIs [...]⟩
Think step by step, but keep your reasoning short, precise, and grounded only in the
history above. [...] 1. Summarize only the important facts the outputs establish [...] 2.
Identify the current subgoal [...] 3. Check the last few actions for useless repetition
[...] 4. Choose the one step that makes the most direct progress toward the current
subgoal. [...] After forming your initial reasoning, perform a brief self-critique [...]
```

Message to the agent:

```
Your next action has been decided for you. Execute exactly the code below as your next
step, without changes, then continue working on the original task from its output.
[Supervisor reasoning]: ⟨reasoning⟩
[Deferred action]: ⟨code⟩
```

Memory-based guidance

Prompt to the agent's model as advisor:

```
An agent is working on this task: ⟨instruction⟩
Here is what it has done so far, most recent last: ⟨last 40 steps⟩
Here are failure patterns this agent has shown on other tasks [...]: ⟨note⟩
[...] Decide whether any pattern applies to what this agent has just done. Then write the
guidance it should receive, in this shape and nothing else:
DIAGNOSIS: one sentence on what is wrong or missing right now. [...]
NEXT: the single concrete next action to take, naming the exact API call or computation.
[...]
THEN: at most two further actions after that.
```

Message to the agent:

```
Guidance on your progress so far is below. Follow it as your next action, then continue
working on the original task.
DIAGNOSIS: ⟨...⟩  NEXT: ⟨...⟩  THEN: ⟨...⟩
Keep these in mind for the rest of the task: ⟨note⟩
```

C Details of the Monitor

Uncertainty estimators. The monitor uses the 14 estimators of Section 2.5, computed as defined in LM-Polygraph (Fadeeva et al., 2023), in four families: probability (maximum sequence probability, maximum token probability, and perplexity); entropy (mean and maximum token entropy, self-certainty, Fisher–Rao distance, and Rényi negentropy); pointwise mutual information (PMI and conditional PMI, each as the mean and the maximum over the tokens of the step); and self-evaluation ($P(\mathrm{True})$ and its normalized form $p(\mathrm{True})/(p(\mathrm{True})+p(\mathrm{False}))$). Table 9 lists the additional computation that each family requires. A value that is missing at a step is replaced by the mean of the estimator over the same trajectory.

Features. The features x_t combine all signals with the step index. In the offline setting they describe each signal at step t relative to the complete trajectory, through its value, its rank and standardized value, its change at the next step, and summary statistics over all steps; in the online setting they describe it relative to the prefix, through its current value, running statistics, percentile, and change from the previous step. Offline, two of these forms correspond to the two abilities of Figure 2b: the summary statistics, which are the same at every step and therefore only tell trajectories apart (detection), and the rank and standardized value within the trajectory, which tell its steps apart (localization); the regression learns how much each form of each signal counts.

Learners and hyperparameters. The value function is an ensemble of gradient-boosted regression trees (Friedman, 2001; Ke et al., 2017) in the offline setting and a ridge regression (Hoerl & Kennard, 1970) in the online setting, both as implemented in scikit-learn (Pedregosa et al., 2011), with the hyperparameters in Table 4. The hyperparameters were chosen once, on AppWorld: every configuration of the grids was scored by the out-of-fold task score on the training halves of the ten

splits at a fixed budget ($\alpha = 30\%$ for Gemma and 10% for Qwen), and the configuration that scored best on both agents together was kept fixed for all splits, both agents, and all three benchmarks; ALFWorld and WebShop were not tuned. The random seed of the learner is the index of the split.

Table 4: Learners and hyperparameters of the monitor, and the grids from which the hyperparameters were chosen.

	Offline	Online
Model	Gradient-boosted regression trees (`HistGradientBoostingRegressor`)	Ridge regression after mean imputation and standardization
Hyperparameters	At most 3 leaves per tree, learning rate 0.05, 200 iterations, at least 20 samples per leaf, ℓ_2 regularization 1.0, no early stopping	$\lambda = 300$
Grid	18 configurations: 3 to 16 leaves, learning rate 0.03 to 0.1, 100 to 400 iterations, 20 or 40 samples per leaf, ℓ_2 regularization 0 to 5	7 values, $\lambda \in \{10, 30, 100, \ldots, 10^4\}$

Threshold calibration. Within the training half, a four-fold cross-validation grouped by task gives an out-of-fold estimate $\hat{V}_m(\tau, t)$ at every step of every training trajectory. For each successful training trajectory we take the largest out-of-fold estimate over its candidate steps, and θ_α is the $(1-\alpha)$ quantile of these maxima. The budget is chosen from $\alpha \in \{5, 10, 20, 30\}\%$ by running the policy with the out-of-fold estimates and θ_α on the training half and keeping the α of highest task score; the model that scores the test half is then refitted on the whole training half. The candidate steps are the non-terminal steps $t < N$. Offline, the monitor steers at the step of largest estimate if this estimate reaches θ_α; online, it steers at the first step $t \geq t_{\min} = 3$ whose estimate reaches θ_α, and the calibration uses only these steps. The budgets from 40% to 100% in Figure 3 are evaluated only to draw the curves and are never selected.

Training target and weights. A separate value function f_m is fitted for each mechanism m on all candidate steps of the failed and the successful training trajectories, with the outcome $\Delta_m(\tau, t)$ of Eq. 1 as the target; on AppWorld, the outcome is computed from the fraction of tests passed. Each step is weighted by $1/N_\tau$, so that long and short trajectories contribute the same total weight to the loss.

D Methods from Prior Work

We re-implement the triggering rule of each method on SOT; like the threshold of VoS, its threshold is set by the budget calibration of Section 4.1 on the training half.

AUQ. AUQ (Zhang et al., 2026b) triggers reflection when the agent's confidence is low, and we keep this trigger. Its uncertainty-aware memory, which has the agent write down at every step its confidence and what it is uncertain about, is not reproduced: in our runs it doubles the agent's output tokens and no task gets stuck in a loop, so it acts as a prompt that makes the agent think one step further rather than as uncertainty that decides when to steer. Its consistency-weighted best-of-N reflection is a steering mechanism rather than a trigger and is not reproduced either.

DIAL. DIAL (Li et al., 2026) decides when to steer with an ℓ_1-regularized logistic gate trained on a binary label of whether steering helps, and we keep this gate. The optimizer that the gate invokes is not reproduced: it ranks candidate actions by the environment's reward, with access to the environment's scoring function (Appendix D.1 of the original), which on AppWorld would mean scoring candidate actions with the hidden unit tests. No other method has this information, and an optimizer without it would no longer be the original, so DIAL steers with memory-based guidance.

Barbi et al. Barbi et al. (2025) predict task success with a polynomial ridge classifier, and we keep this classifier. Their intervention resets the communication between the agents of a multi-agent system; our agents act alone, so there is no communication to reset, and we steer with memory-based guidance instead.

DS-MCM. DS-MCM (Sun et al., 2026) triggers when the reasoning entropy of a step deviates from its level on successful steps; we keep this trigger and steer with memory-based guidance.

ReDAct. ReDAct (Piatrashyn et al., 2026) defers a step to a stronger model when its perplexity is high, and we keep this deferral as it is.

E Additional Results

Table 5 repeats Table 1 with memory-based guidance for every policy; Table 6 compares the monitors built on each mechanism with the routed monitor of Section 4.2, and Table 7 gives the cost of the mechanisms' own calls used in Section 4.4.

Table 5: Table 1 with memory-based guidance for every policy: AUQ and ReDAct steer with it instead of reflection and the stronger model, so that all methods differ only in when and where they steer. Gray: references, not ranked; bold and underline: best and second-best policy in each row.

		References		Steering policies (all with memory-based guidance)					
		Unmod.	Always	AUQ	DIAL	Barbi et al.	DS-MCM	ReDAct	VoS (ours)
Offline									
AppWorld	Gemma	69.0	71.6 ±1.5	<u>72.2</u> ±1.6	68.9 ±0.3	69.7 ±1.2	70.4 ±0.9	69.5 ±0.6	**73.4** ±1.9
	Qwen	65.5	55.6 ±1.6	64.3 ±0.7	<u>65.4</u> ±0.4	65.3 ±0.7	64.8 ±0.6	65.3 ±0.5	**66.5** ±0.8
ALFWorld	Gemma	74.5	76.6 ±2.8	73.6 ±0.8	<u>80.1</u> ±1.8	80.0 ±1.8	78.4 ±1.7	77.4 ±1.1	**85.2** ±1.6
	Qwen	52.5	69.0 ±2.8	55.3 ±1.7	62.0 ±3.5	61.0 ±2.4	57.0 ±1.6	<u>63.5</u> ±2.6	**78.4** ±3.2
WebShop	Gemma	66.7	68.7 ±0.9	<u>71.3</u> ±0.8	68.3 ±0.5	69.4 ±0.2	67.6 ±0.3	67.7 ±0.4	**72.3** ±1.1
	Qwen	62.5	63.9 ±0.7	63.5 ±0.5	63.3 ±0.6	63.6 ±0.4	<u>65.0</u> ±0.0	64.3 ±1.0	**67.1** ±0.9
Online									
AppWorld	Gemma	69.0	71.6 ±1.5	<u>71.9</u> ±1.6	70.0 ±1.7	69.7 ±0.8	71.2 ±0.9	70.9 ±1.2	**73.6** ±1.8
	Qwen	65.5	55.6 ±1.6	62.4 ±0.7	<u>65.3</u> ±0.7	<u>65.3</u> ±0.8	64.7 ±1.3	63.8 ±0.6	**66.6** ±0.4
ALFWorld	Gemma	74.5	76.6 ±2.8	75.3 ±1.2	<u>82.0</u> ±2.7	81.3 ±1.2	77.4 ±2.5	77.9 ±1.3	**82.6** ±2.2
	Qwen	52.5	69.0 ±2.8	63.8 ±2.2	67.7 ±5.3	<u>70.1</u> ±1.7	62.3 ±2.1	65.3 ±3.0	**71.1** ±2.6
WebShop	Gemma	66.7	68.7 ±0.9	69.8 ±0.3	69.0 ±0.4	<u>70.1</u> ±1.0	67.5 ±0.2	68.2 ±0.3	**72.6** ±0.4
	Qwen	62.5	63.9 ±0.7	62.7 ±1.3	**66.0** ±0.8	64.7 ±1.8	64.0 ±0.6	63.9 ±0.0	<u>65.6</u> ±0.8

Table 6: Improvement over unmodified execution of the monitor built on each mechanism, and of a routed monitor that chooses at every step the mechanism of highest estimated value (Gemma).

	AppWorld		ALFWorld		WebShop	
	Offline	Online	Offline	Online	Offline	Online
Memory-based guidance (VoS)	+.045	+.046	+.107	+.081	+.055	+.058
Reflection	+.008	+.010	+.080	+.018	+.026	+.002
Critic	+.022	+.002	+.056	+.027	+.051	+.042
Stronger model	+.019	+.015	+.055	+.031	+.046	+.037
Routed	+.035	+.049	+.092	+.074	+.047	+.055

Table 7: Cost of the mechanism's own call per steering event on AppWorld: prompt and completion tokens, number of calls, the tokens in agent-step equivalents, and their share of the total cost of a steering event.

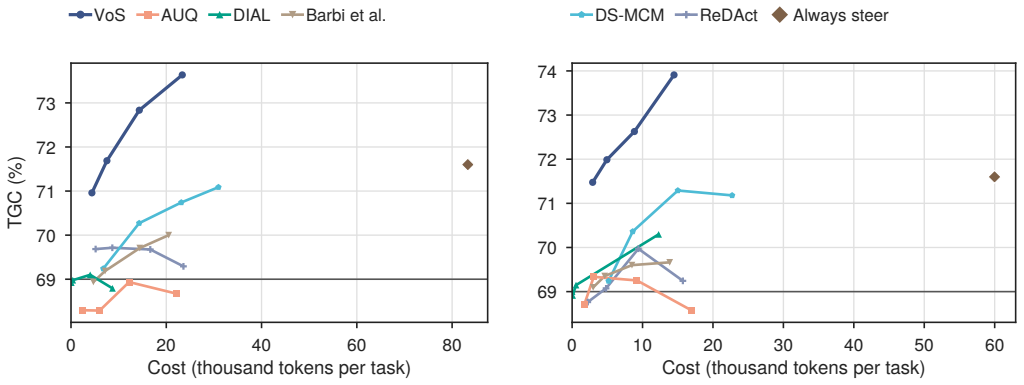
	Gemma				Qwen			
	Tokens	Calls	Steps	Share	Tokens	Calls	Steps	Share
Reflection	2,261	1.00	0.30	4.0%	4,258	1.00	0.48	4.3%
Critic	6,300	3.02	0.84	10.5%	20,094	1.06	2.28	18.8%
Stronger model	6,515	2.04	0.87	9.5%	8,748	2.01	0.99	8.3%
Memory-based guidance	3,589	1.00	0.48	5.1%	6,579	1.00	0.75	6.5%

Table 8: Change of the benchmark metric net of resampling: for every steered cell, the change produced by a bare rewind at the same step, which continues the trajectory from step t with no message, is subtracted, so a policy is credited only with what its message adds. Metrics as in Table 1, as mean ± standard deviation. The gray reference is a monitor trained on the rewind alone, which learns where re-running helps. Bold and underline: best and second-best policy in each row; shading from red (loss) to green (gain).

		Reference	Steering policies, net of the rewind					
		Rewind monitor	AUQ	DIAL	Barbi et al.	DS-MCM	ReDAct	VoS (ours)
Offline								
AppWorld	Gemma	+0.7±0.7	+0.8±1.2	+1.2±1.7	+2.1±1.3	+3.4±0.8	+1.7±0.8	**+5.2**±1.7
	Qwen	+0.1±1.1	−0.6±0.5	−0.4±0.5	−0.6±0.8	−1.4±0.9	−0.1±0.3	**+1.4**±0.7
ALFWorld	Gemma	+4.6±2.1	−0.1±0.9	+3.6±1.4	+3.3±2.0	+2.9±1.7	−1.9±1.3	**+5.0**±1.7
	Qwen	+9.4±1.9	+0.3±0.5	+7.4±5.0	+4.8±2.7	+4.5±2.2	−1.3±1.1	**+17.3**±4.1
WebShop	Gemma	+2.4±0.6	+0.3±0.1	+0.5±0.8	+0.0±0.2	+0.3±0.3	−0.3±1.2	**+4.7**±1.5
	Qwen	+1.3±0.3	−1.7±0.4	+0.3±1.0	+0.9±0.7	+1.4±0.8	+1.9±0.7	**+2.8**±1.0
Online								
AppWorld	Gemma	+0.0±0.6	+0.8±1.5	+4.7±3.6	+2.3±1.6	+3.9±0.9	+2.1±1.1	**+5.1**±1.3
	Qwen	−0.2±1.0	−0.3±0.8	−0.5±0.6	−0.7±0.6	−2.0±1.2	**+0.1**±0.5	−0.1±0.3
ALFWorld	Gemma	+1.7±0.9	−0.3±0.9	+3.0±2.9	+3.9±1.5	+2.3±1.7	−1.0±1.2	**+4.9**±2.1
	Qwen	+1.7±1.8	+0.3±0.4	+12.4±3.2	+13.3±3.2	+8.8±1.9	+2.4±1.5	**+15.7**±2.6
WebShop	Gemma	+2.6±1.2	+0.3±0.3	−0.2±1.3	+2.8±1.5	−1.0±0.5	−0.2±0.9	**+4.7**±0.9
	Qwen	+0.1±0.9	−0.4±0.6	**+2.1**±0.8	+1.6±2.3	+1.2±1.0	+0.7±0.2	+1.6±1.4

F Additional Figure Details

These figures retain the detailed comparisons behind the main-text summaries, using the same outcomes and task splits.



(a) Offline (b) Online

Figure 7: Figure 5 with the cost counted in tokens: those of the steered continuation and the mechanism call, minus, online, those of the replaced remainder.

Table 9: Cost of the uncertainty signals per agent step (AppWorld): tokens read or written in addition to the agent's own. Re-read: the scoring pass reads the agent's context again; cached: the context is kept in a prefix cache and only the appended question or prompt is read. Probabilities and entropies come from the pass that generated the action.

		Gemma		Qwen	
	Extra pass	re-read	cached	re-read	cached
Probability, entropy	none	0	0	0	0
PMI	action only	150	150	118	118
$P(\text{True})$	context + action + question	7,559	33	8,857	37
Black-box	context + action + prompt	7,604	75	8,903	75
Agent's own step		7,529		8,828	

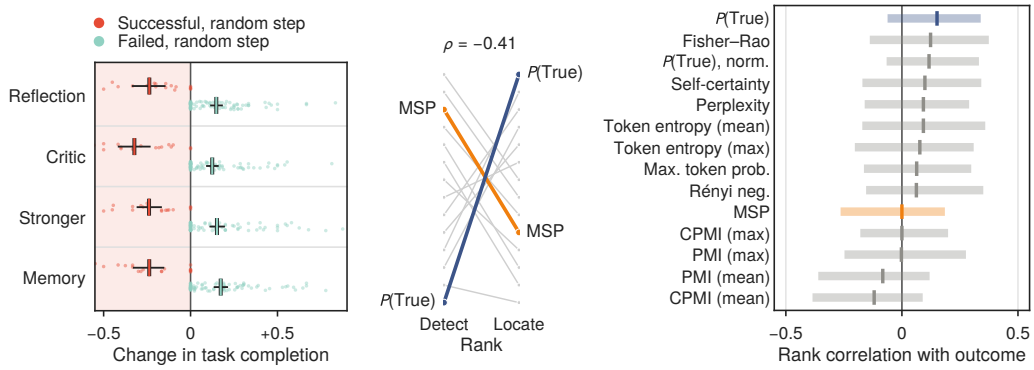


Figure 8: Figure 2 for Qwen.

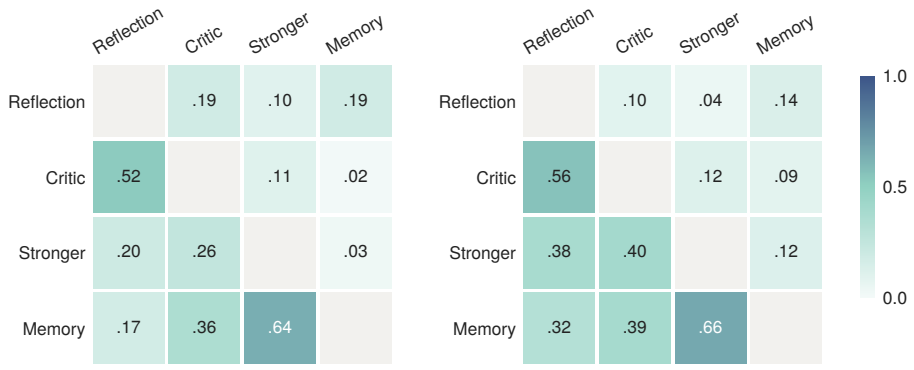


Figure 9: Agreement between mechanisms on failed trajectories (left: Gemma; right: Qwen). Below the diagonal: Spearman correlation of best outcomes across trajectories; above: across steps, averaged over trajectories.