MEMORY-AWARE ROUTING FOR COST-EFFICIENT LANGUAGE AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Language-model agents must balance the reliability of strong models against the cost of invoking them at every decision step. Retrieved experience offers another way to manage this trade-off: a past trajectory may provide the support a smaller model needs to match a stronger reference. However, memory can also mislead, and its usefulness depends on both the current state and the model that consumes it. Routing therefore requires deciding not only which model to invoke, but also which memory can make that model sufficient. We formulate this problem as cost-constrained selection over memory-model pairs and develop a two-stage framework that separates relative memory benefit from absolute routing reliability. The first stage shortlists memories by predicted alignment gain; the second uses historical outcome evidence to estimate model-specific reliability, with a shared-bias-low-rank formulation for cross-model prediction. We evaluate an evidence-grounded approximation on OfficeBench, ScienceWorld, and τ-bench, alongside separate component studies of sharing and shortlisting. The evaluated router achieves up to 17.8% proxy cost savings over always invoking the reference, routing 25.7% of states to smaller models at 82.0% routed precision. These findings support treating episodic memory can effectively serve as part of model allocation.

1 INTRODUCTION

AI agents solve tasks through repeated cycles of observation, reasoning, tool use, and action (Yao et al., 2023). Strong language models make these agents capable, but invoking the strongest model at every decision state is expensive; this quality-cost trade-off has motivated cost-aware LLM cascading (Chen et al., 2023). Replacing the strongest model uniformly with a smaller one is not a solution: difficult states may require precise tool arguments, long-horizon consistency, or reasoning that the smaller model cannot reliably provide. Many agent workloads, however, are repeated or nearly duplicated. Benchmarks for enterprise knowledge work and realistic web interaction contain recurring workflows and closely related web-navigation tasks (Drouin et al., 2024; Zhou et al., 2023). In these settings, a successful trajectory from a previous task can provide a concrete reference for the current state: which tool to call, which arguments matter, or which intermediate result resolves the next decision. Prior agent architectures likewise store and retrieve past experiences to inform subsequent behavior (Park et al., 2023; Shinn et al., 2023). Such episodic experience can reduce the reasoning burden of the current task and may allow a cheaper, less capable model to produce the action that would otherwise require a strong model.

This potential changes the routing problem: the model required at a decision state may depend on the experience available to it. Yet existing model-routing policies primarily estimate difficulty, expected quality, or a routing preference from the current query or state itself (Ding et al., 2024; Ong et al., 2024; Hu et al., 2024). They do not ask whether a retrieved episode changes what a particular model can accomplish. Our central assumption is that a memory from a sufficiently similar past task can turn some states that originally require a large model into states that a smaller model can solve reliably. This is a conditional assumption: the same memory can be useful for one model, unhelpful for another, or insufficient for all of them.

This motivates our central question: *When can retrieved episodic experience enable a cheaper model to safely replace a stronger model at an agent decision state?* Answering this question requires

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.

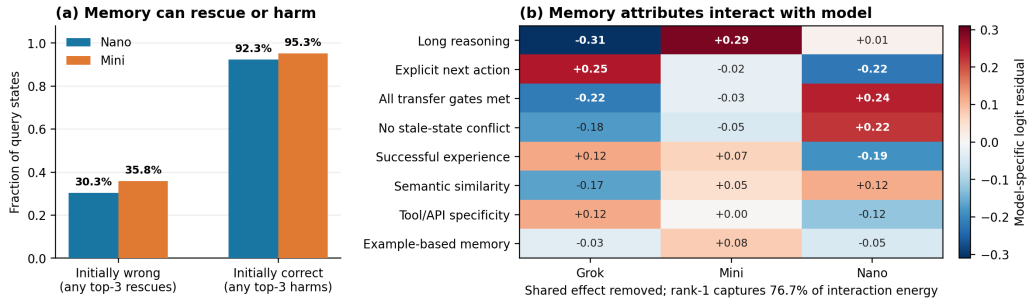

Figure 1: Memory can rescue or harm action alignment (a); model-specific concept effects (b). Red/blue indicate effects above/below the cross-model mean.

resolving two coupled decisions: which memory provides useful support for the current state, and which model can reliably exploit that support. First, textual similarity is not enough to decide whether a memory is similar *enough*: a related trajectory may not supply the evidence needed for the current action. Second, even if a memory is useful, the router must determine which candidate model can exploit it. This is a model-dependent question about action generation, not merely about task difficulty or retrieval relevance.

Our preliminary analysis illustrates why both decisions matter (Figure 1). A lightweight development diagnostic in Figure 1 makes both dependencies visible. Among states where a small model's no-memory majority action disagrees with the reference, at least one of the three retrieved memories restores agreement for 30.3% of Nano states and 35.8% of Mini states. Yet memory is not uniformly helpful: among states the model initially solves, at least one retrieved memory breaks agreement in 92.3% and 95.3% of cases. Moreover, Figure 1(b) illustrates that different models respond differently to the same memory characteristics. Longer reasoning is relatively more favorable for Mini, whereas an explicit next action is relatively more favorable for Grok. These patterns suggest that choosing a useful memory also requires considering which model will use it.

These observations motivate a memory-aware routing framework that couples memory and model selection while separating two quantities: how much a memory helps, and whether the resulting model-memory pair is reliable enough to use. To this end, we propose a grounded memory-aware routing framework. Its neural formulation estimates how much a memory changes each candidate model's reference agreement and uses the model-marginalized gain to shortlist candidate trajectories. Historical memory-conditioned predictions then support the model decision. Experiments on OFFICEBENCH, SCIWORLD, and τ-bench demonstrate up to 17.8% proxy cost savings over always invoking the reference model, with the SCIWORLD operating point routing 25.7% of states to smaller models at 82.0% routed precision.

Our contributions are threefold:

- **Memory-conditioned model routing.** We formulate agent routing as selecting a memory-model pair under a reference-action agreement constraint. This formulation makes explicit that the model needed at a decision state depends on the retrieved experience.
- **A gain-reliability decomposition.** We develop a two-stage formulation that separates relative memory benefit from absolute routing reliability. Gain-based shortlisting identifies promising memories, while historical outcome evidence and a shared-bias-low-rank predictor capture differences in how candidate models exploit them.
- **Empirical characterization and routing evaluation.** We evaluate the proposed method on three benchmarks. The evaluated router achieves up to 17.8% proxy cost savings over fallback execution.

2 RELATED WORK

LLM routing and cascading methods allocate queries among models with different quality-cost trade-offs. One family predicts query difficulty or expected answer quality, then sends easy queries

to a smaller model and escalates the rest. FrugalGPT learns cost-aware LLM cascades (Chen et al., 2023); Hybrid LLM predicts query difficulty under a user-specified quality level (Ding et al., 2024); and RouteLLM learns a strong-versus-weak routing preference from pairwise preference data (Ong et al., 2024). A complementary family obtains an additional signal after the small model is invoked: AutoMix uses few-shot self-verification before escalation (Aggarwal et al., 2023), while BEST-Route jointly chooses a model and a best-of-n sampling budget (Ding et al., 2025). RouterBench provides a common multi-LLM evaluation framework and inference-outcome dataset for comparing these query-level strategies (Hu et al., 2024).

Agent routing expands the decision from selecting one model to selecting an execution structure. MasRouter jointly determines collaboration mode, role allocation, and LLM assignment in a multi-agent system (Yue et al., 2025); MoMA routes across both models and agents using intent recognition and a context-aware selection mechanism (Guo et al., 2025). Most closely related, Agent-as-a-Router frames coding-task routing as a Context-Action-Feedback loop and accumulates execution-grounded experience over time (Zhou et al., 2026). These works establish that routing can benefit from execution context and feedback. Our setting differs in decision granularity and in the role of memory. To the best of our knowledge, no work has explored the effect of episodic memory on routing decisions.

3 Problem Formulation

At each agent decision state, a frozen hybrid retriever provides a shortlist of query-memory pairs (C_i, M_{ij}), augmented with the null candidate $M_{i0} = \varnothing$. The system can invoke one of K candidate models $\mathcal{K} = \{1, \ldots, K\}$ with heterogeneous capabilities and costs, together with a reliable fallback model m_{ref}. For each candidate k, let $y_{ijk} = 1$ when at least two of three stochastic actions under memory M_{ij} are judged equivalent to the fixed reference action. The cost-constrained target is to select the least costly candidate whose reliability $p_{ijk} = P(y_{ijk} = 1 \mid C_i, M_{ij}, k)$ meets the required threshold, using m_{ref} when none qualifies.

Different models can exploit the same memory differently, so we retain the full set of labels $\{y_{ijk} : M_{ij} \in \mathcal{M}_i,\ k \in \mathcal{K}\}$. For a routing policy π and tolerated failure rate δ, the deployment objective is

$$\min_{\pi} \quad \mathbb{E}[C_i(\pi(i))] \qquad \text{s.t.} \qquad P\big(y_{i,\pi(i)} = 1\big) \geq 1 - \delta, \tag{1}$$

where $C_i(k)$ includes the cost of model k under the retrieved context. This formulation treats routing as multi-model cost minimization under a quality constraint rather than a binary large-versus-small decision.

4 Methodology

Given an agent state and a bank of raw episodic trajectories, our goal is to choose a memory-model pair that reduces inference cost while preserving agreement with the reference. The key quantity is how much a memory changes a model's behavior on the current task.

4.1 Overall Framework

For state C_i, let $\mathcal{M}_i$ contain the candidate memories, including $M_{i0} = \varnothing$. Candidate model $k \in \mathcal{K}$ has cost $c_i(k)$ and memory-conditioned reliability $p_{ijk} = P(y_{ijk} = 1 \mid C_i, M_{ij}, k)$. With tolerated failure rate δ, the joint routing objective is

$$(M_i^\star, k_i^\star) \in \arg \min_{M_{ij} \in \mathcal{M}_i,\, k \in \mathcal{K}} c_i(k) \qquad \text{s.t.} \qquad p_{ijk} \geq 1 - \delta. \tag{2}$$

The reliable reference model is invoked if no candidate pair meets the constraint. To isolate what memory contributes, consider the same task and model with and without M_{ij}. Let q_{ijk} be the probability that one sampled action agrees with the reference. We define memory gain as the change in that agreement probability:

$$\text{Gain}(k, M_{ij}, C_i) = q_{ijk} - q_{i0k}. \tag{3}$$

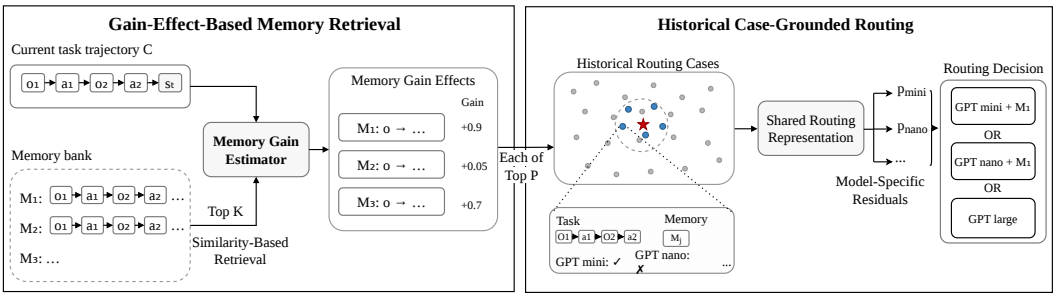

Figure 2: Our proposed two-stage method: gain-based memory shortlisting followed by cost-constrained model selection.

This is a within-model, within-task effect: a positive value means the memory improves reference alignment for model k. The routing probability p_{ijk} concerns the majority-alignment event over repeated samples, so gain alone does not establish feasibility.

This motivates the two-stage framework in Figure 2. Direct joint routing requires model-specific reliability for every memory-model pair, while the available rollouts support relative comparisons of memory effects more readily than absolute reliability estimates. We therefore learn memory rankings within each model, aggregate their gain estimates to shortlist memories, and predict reliability only for the shortlisted pairs. For the first stage, we aggregate gain over the model dimension:

$$\mathrm{Gain}_{\mathrm{marg}}(M_{ij}, C_i) = \tau \log \left[\frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} \exp\left(\frac{\mathrm{Gain}(k, M_{ij}, C_i)}{\tau} \right) \right]. \tag{4}$$

This smooth maximum approaches the largest model-specific improvement as $\tau \to 0$. Let s_{ij} be the learned score for $\mathrm{Gain}_{\mathrm{marg}}(M_{ij}, C_i)$. Stage I keeps the P highest-scoring nonempty memories, $\mathcal{S}_i = \mathrm{TopP}(\mathcal{M}_i \setminus \{M_{i0}\}; s_{ij})$. The no-memory option is retained separately. Stage II estimates $\widehat{p}_{ijk}$ for each shortlisted memory-model pair and solves the original cost-constrained decision on this smaller set:

$$(\widehat{M}_i, \widehat{k}_i) \in \arg \min_{M_{ij} \in \{M_{i0}\} \cup \mathcal{S}_i,\, k \in \mathcal{K}} c_i(k) \qquad \text{s.t.} \qquad \widehat{p}_{ijk} \geq 1 - \delta. \tag{5}$$

If the shortlist contains a memory from an optimal pair, restricting the search preserves the solution to Equation 2; the gain-based shortlist does not guarantee this condition. Both stages share a routing representation learned from the same rollout outcomes, so Stage I can screen on alignment gains while Stage II retains the information needed to distinguish models.

4.2 Learning Model-Marginalized Memory Gain

Stage I learns two things from a state-memory pair: model-conditioned gain scores that can be aggregated to rank memories, and an embedding that preserves finer model-specific information for Stage II. Textual similarity does not directly reveal either signal.

We obtain supervision from paired rollouts at a fixed agentic decision step. Let $i = (t, \ell)$ identify step ℓ of task t, with pre-action state C_i and fixed reference action A_i^{ref}. At that same step, we cross all candidate memories with all candidate models; only memory and model vary. Under the same prompt template and decoding settings, each cell produces $R = 3$ stochastic next actions:

$$\begin{aligned} A_{ijk}^{(r)} &\sim f_k(\cdot \mid C_i, M_{ij}), \\ (M_{ij}, k) &\in (\{M_{i0}\} \cup \mathcal{R}_K(C_i)) \times \mathcal{K}, \quad r = 1, \ldots, R. \end{aligned} \tag{6}$$

We judge each action against the fixed reference. Write $b_{ijk}^{(r)} = \mathbf{1}[\mathrm{Align}(A_{ijk}^{(r)}, A_i^{\mathrm{ref}})]$ for the resulting binary judgment. The observed alignment rate is $a_{ijk} = \frac{1}{R} \sum_{r=1}^{R} b_{ijk}^{(r)}$. For every condition we retain the state, raw memory, model identity, reference action, sampled actions, individual judgments, alignment rate, and majority label $y_{ijk} = \mathbf{1}[a_{ijk} \geq 2/3]$.

Comparing the memory and no-memory conditions for the same step and model gives the observed gain $a_{ijk} - a_{i0k}$. Thus each fixed (i, k) supplies a ranking of memories by their effect on that model's reference agreement.

The pair encoder T_ϕ represents each trajectory by a task embedding followed by step embeddings, each combining that step's action and environment observation. Current-trajectory steps query the memory trajectory through cross-attention, conditioning the current state on relevant past steps. Attention pooling gives $z_{ij} = T_\phi(C_i, M_{ij})$. A shared model-conditioned head predicts gain relative to no memory, $s_{ijk} = g_\varphi(z_{ij}, k) - g_\varphi(z_{i0}, k)$, so $s_{i0k} = 0$. Applying the aggregation in Equation 4 to $\{s_{ijk}\}_k$ gives the Stage-I retrieval score s_{ij}. Stage II uses the same z_{ij} for model-specific routing.

For each fixed step i and model k, the two lists below run over the same memory candidates, including the no-memory anchor $j = 0$. We rank memories within that model, then sum the losses over models:

$$\mathcal{L}_{\text{gain}} = \sum_i \sum_{k \in \mathcal{K}} \text{ListwiseRanking}((s_{ijk})_j,\ (a_{ijk} - a_{i0k})_j)\,. \tag{7}$$

The encoder also receives the model-specific routing loss from Section 4.3, encouraging z_{ij} to retain differences that the scalar score cannot express. At inference, s_{ij} ranks the recalled memories and their embeddings are passed to Stage II.

4.3 Learning Model-Specific Memory Effects from Experience

For each memory M_{ij} shortlisted by Stage I (or the no-memory option), Stage II takes the current state C_i and historical routing cases $\mathcal{H}$, and returns a reliability estimate for every candidate model:

$$\widehat{\mathbf{p}}_{ij} = F_\Theta(C_i, M_{ij}; \mathcal{H}) = \big(\widehat{p}_{ijk}\big)_{k \in \mathcal{K}}. \tag{8}$$

Each component estimates the probability of majority agreement with the reference at this fixed decision step. The vector is the information needed to choose a reliable model-memory pair under the cost constraint in Equation 5.

Observed Structure and Inductive Bias. How should F_Θ share information across models while retaining their different responses to memory? Figure 1 shows that memory can either restore or disrupt reference agreement. A single shared predictor would miss those differences; independent predictors would discard the shared signal. We use this pattern as an inductive bias, not as a claim that all model pools have rank-one responses.

We therefore expand the predictor in Equation 8 around the no-memory reliability vector $\widehat{\mathbf{p}}_i^{(0)}$. Let u_{ij} be a routing feature built from the current pair and historical cases, as specified below. For a nonempty memory, we use

$$\begin{aligned} F_\Theta(C_i, M_{ij}; \mathcal{H}) &= \sigma\Big(\text{logit}\,\widehat{\mathbf{p}}_i^{(0)} + h(u_{ij})\mathbf{1} + \boldsymbol{\beta} + r(u_{ij})\boldsymbol{\lambda}\Big)\,, \quad j > 0, \\ F_\Theta(C_i, M_{i0}; \mathcal{H}) &= \widehat{\mathbf{p}}_i^{(0)}. \end{aligned} \tag{9}$$

Here sigmoid and logit act elementwise. The shared effect h transfers evidence across models; $\boldsymbol{\beta}$ captures their average differences in response to memory; and the shared residual direction r, scaled by model loadings $\boldsymbol{\lambda}$, permits structured departures. This restricts how the models can differ without forcing their predictions to agree.

Historical-Case Features. The effect must also be grounded in what models have done, rather than inferred from trajectory text alone. We retain a training-only bank of historical routing cases

$$\mathcal{H} = \{(C_n, M_n, k, y_{nk}) : n = 1, \ldots, N,\ k \in \mathcal{K}\}, \tag{10}$$

where $\mathbf{y}_n = (y_{n1}, \ldots, y_{nK})$ is the observed outcome profile for case (C_n, M_n). Using the Stage-I encoder $z_{ij} = T_\phi(C_i, M_{ij})$, we learn a metric over routing cases and summarize the outcomes of nearby cases with soft K-nearest-neighbor weights:

$$\alpha_{ijn} = \text{softmax}_{n \in \mathcal{N}_{ij}}(\text{sim}_\theta(z_{ij}, z_n)/\tau_H)\,, \qquad \mathbf{e}_{ij} = \sum_{n \in \mathcal{N}_{ij}} \alpha_{ijn}\mathbf{y}_n. \tag{11}$$

Here $z_n = T_\phi(C_n, M_n)$ and $\mathcal{N}_{ij}$ contains the nearest historical cases under the learned metric; training excludes cases from the query task. The feature $\mathbf{e}_{ij}$ exposes how each candidate model

behaved on comparable state-memory pairs, while z_{ij} retains the current pair's details. We set $u_{ij} = (z_{ij}, \mathbf{e}_{ij})$.

Training. The no-memory vector in Equation 9 is $\widehat{\mathbf{p}}_i^{(0)} = (q_{\omega,k}(C_i))_{k\in\mathcal{K}}$, learned from null-memory rollouts. It captures each model's baseline capability; the historical-evidence-conditioned terms learn what the memory changes. The neural objective jointly trains the trajectory encoder, model-conditioned gain head, historical-case metric, and model-specific routing head on the paired rollouts from Section 4.2; the no-memory predictor is fitted on null-memory rollouts. Let $\mathcal{L}_{\text{route}} = \sum_{i,j,k} \ell(\widehat{p}_{ijk}, y_{ijk})$ use a proper scoring loss ℓ for majority agreement, and let $\mathcal{L}_{\text{fractional}}$ supervise the observed alignment fractions a_{ijk}. The principal joint objective is

$$\mathcal{L} = \mathcal{L}_{\text{route}} + \gamma\mathcal{L}_{\text{fractional}} + \alpha\mathcal{L}_{\text{gain}}. \tag{12}$$

The gain loss shapes the common routing representation, while the routing loss trains the KNN metric and model-specific reliability from observed outcomes. Historical outcomes used as KNN features are drawn only from other training tasks, so a case never supplies its own label as evidence.

Online Prediction. At prediction time, Stage I shortlists memories, the historical cases provide $\mathbf{e}_{ij}$ for each pair, and Equation 9 estimates each model's reliability. Model-specific calibration replaces the common target threshold in Equation 5 by validated thresholds τ_k. This plug-in policy minimizes cost over its predicted-feasible shortlist,

$$(\widehat{M}_i, \widehat{k}_i) \in \arg\min_{M_{ij}\in\{M_{i0}\}\cup\mathcal{S}_i,\, k\in\mathcal{K}} c_i(k) \qquad \text{s.t.} \qquad \widehat{p}_{ijk} \geq \tau_k. \tag{13}$$

If no candidate pair is supported, the system invokes m_{ref}. This is a cost-minimal choice within the predicted-feasible set, not a guarantee of true reliability or of preserving the optimum before shortlisting.

5 Experiments

5.1 Experimental Setup

We evaluate next-action routing on OfficeBench, SciWorld, and τ-bench against a fixed `gpt-5.4` reference action. A task-disjoint bank contains 225 raw trajectories; 525 query tasks yield 6,404 decision states, with 335/85/105 query tasks assigned to fit/validation/test. At each state, Nano and Mini act with no memory and with each of three retrieved trajectories, generating three stochastic actions per condition. Majority action equivalence to the reference defines the routing label. We report routed coverage and precision, overall sampled-action alignment, and proxy cost saving (Nano:Mini:reference $=$ 1:5:15), with Wilson intervals and task-level bootstrap bands. Full retrieval, labeling, split, and metric protocols appear in Appendix A.

5.2 Baselines and Implementation

We compare against query-only quality-gap and preference routers, zero-shot and supervised LLM routers, pair MLP/KNN controls, retrieve-then-route, model-agnostic memory utility, and shared or independent memory-effect estimators. All use the same task-disjoint split, reference actions, and action-equivalence criterion; claim-focused memory controls share the same top-three retrieval and no-memory gate. Implementation and selection details appear in Appendix A.

5.3 Main Results

Table 1 reports end-to-end routing separately for the three datasets. The zero-shot LLM makes a direct test-time decision under a fixed 80% safety instruction and never observes labels or a reference action. SFT uses its validation-calibrated policy. The adapted controls use the common validation safety-constrained search detailed in Appendix A; the query-capability ablation and proposed router also select thresholds on validation tasks. No test label selects a threshold; test-label upper bounds are reported separately in Appendix Table 7.

Validation calibration yields a supported nonzero operating point on all three environments. On SciWorld, it reaches 25.7% coverage at 82.0% precision and 17.8% cost saving. On OfficeBench,

Table 1: End-to-end routing by dataset (%). Cov.: coverage; Prec.: routed precision; OA: overall sampled-action alignment; Save: proxy cost saving. N/A: no small-model routes; †: identical query-only decisions.

	OFFICEBENCH				SCIWORLD				TAUBENCH			
Method	Cov.	Prec.	OA	Save	Cov.	Prec.	OA	Save	Cov.	Prec.	OA	Save
Fixed policies												
Always reference	0.0	N/A	100.0	0.0	0.0	N/A	100.0	0.0	0.0	N/A	100.0	0.0
Always Nano (no memory)	100.0	36.4	51.2	93.3	100.0	25.4	37.1	93.3	100.0	34.6	44.6	93.3
Always Mini (no memory)	100.0	48.1	57.2	66.7	100.0	52.2	57.9	66.7	100.0	42.3	52.9	66.7
Language-model routers												
`gpt-5.4` zero-shot	47.7	39.2	75.1	32.6	26.2	21.1	80.3	17.8	32.7	29.4	79.5	22.3
`Qwen3-1.7B` SFT	0.0	N/A	100.0	0.0	2.9	96.6	99.9	1.9	0.0	N/A	100.0	0.0
Query-level adapted routers (no memory)												
Hybrid-LLM-style quality gap	0.0	N/A	100.0	0.0	33.5	79.9	94.0	22.4	0.0	N/A	100.0	0.0
RouteLLM-style preference	0.0	N/A	100.0	0.0	29.0	75.1	93.7	19.6	0.0	N/A	100.0	0.0
Memory-aware adapted controls												
Top-3 + pair MLP	0.0	N/A	100.0	0.0	22.0†	84.5	97.0	14.7	8.7	33.3	96.2	5.8
Top-3 + pair KNN	0.0	N/A	100.0	0.0	22.0†	84.5	97.0	14.7	0.0	N/A	100.0	0.0
Rank-1 + pair MLP	0.0	N/A	100.0	0.0	22.0†	84.5	97.0	14.7	0.0	N/A	100.0	0.0
Rank-1 + fixed pair scorer	0.0	N/A	100.0	0.0	23.8	81.9	96.1	16.2	0.0	N/A	100.0	0.0
Model-agnostic memory utility	7.0	40.0	96.4	5.3	22.0†	84.5	97.0	14.7	20.2	38.1	91.0	14.2
Shared memory effect (linear)	17.8	52.6	92.5	11.8	22.0†	84.5	97.0	14.7	0.0	N/A	100.0	0.0
Independent effects (linear)	15.4	66.7	94.9	10.3	22.1	84.2	96.9	14.8	8.7	33.3	94.2	8.1
Proposed router and ablation												
Query capability only	6.5	50.0	97.2	4.5	22.0	84.5	97.0	14.7	0.0	N/A	100.0	0.0
Proposed router	**7.0**	**86.7**	**98.9**	**4.7**	**25.7**	**82.0**	**95.8**	**17.8**	**4.8**	**80.0**	**99.4**	**3.2**

it routes 7.0% at 86.7% precision and saves 4.7%. On TAUBENCH, it routes 4.8% at 80.0% observed precision and saves 3.2%. The small number of selected TauBench states limits conclusions about broad coverage. Dagger-marked controls make exactly the same 220 Mini no-memory decisions as the SCIWORLD query-capability gate, with no memory routes.

The LLM-router controls expose complementary failures. Despite being instructed to route only above 80% estimated safety, zero-shot `gpt-5.4` routes 397 states (30.2%) at only 26.4% aggregate precision; all routed decisions carry self-reported confidence of at least .84, so its confidence is not behaviorally calibrated. The SFT router does learn a reliable tail: its globally validation-calibrated threshold routes 29 SCIWORLD states at 96.6% precision. However, it abstains entirely on the other datasets and all 29 routes use Mini without memory. Thus direct language-model reasoning is overconfident, whereas end-to-end SFT is safe only in a narrow no-memory region; neither baseline identifies a deployable memory-conditioned branch.

Routing composition. Figure 3 visualizes the mix of no-memory and memory-conditioned routing decisions for the same final policy as Table 1. In OFFICEBENCH, 100% of selected small-model routes use memory; of these memory routes, 86.7% are safe and 20.0% are strict rescues. In SCIWORLD, 66.7% of memory routes are safe and 41.7% are strict rescues, while no-memory small-model routes cover 22.0% of all states. In TAUBENCH, 80.0% of memory routes are safe. These branch-level results motivate the support and uncertainty analysis below.

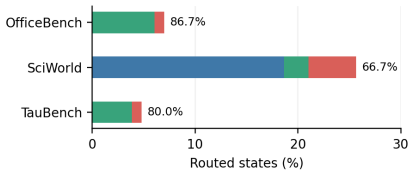

Figure 3: Routing composition: safe without memory (blue), safe with memory (green), and unsafe (red). Labels show safe/selected memory routes.

Precision-coverage trade-off. Here we evaluate whether our method offers a better routing trade-off

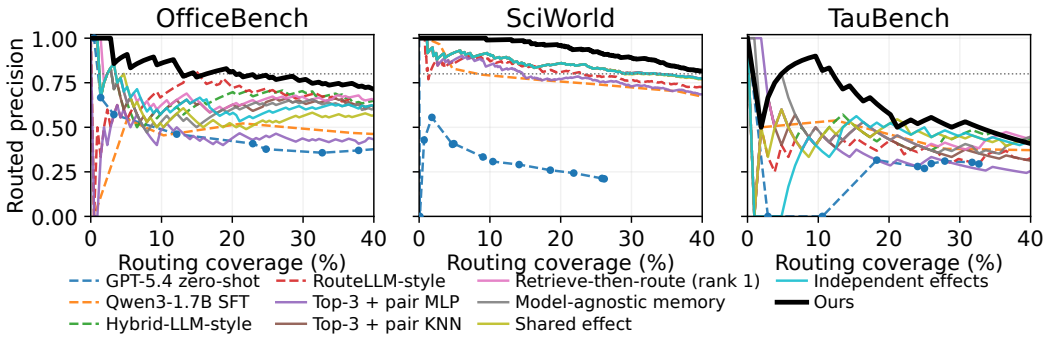

Figure 4: Precision-coverage (dotted: 80%); policy and omitted-point details: §5.3.

Table 2: Fixed-policy ablations. O/S/T: OfficeBench/SciWorld/TauBench; Mem.: safe/selected memory routes. Coverage, precision, and savings are fractions.

Variant	Cov.	Prec.	Save	O Cov./Prec.	S Cov./Prec.	T Cov./Prec.	Mem.	Rescue
Full proposed router	.210	.822	.145	.070/.867	.257/.820	.048/.800	41/56	18
− execution adaptation	.210	.797	.145	.070/.867	.257/.793	.048/.800	34/56	14
− Nano memory routes	.189	.843	.126	.070/.867	.229/.843	.048/.800	24/29	7
− memory branch	.167	.845	.111	.000/-	.220/.845	.000/-	0/0	0

across operating points. We sweep routing confidence thresholds and measure routed precision against small-model coverage, including both memory-conditioned and no-memory routes. Figure 4 shows that our method lies above the compared baselines over most of the plotted coverage range, with particularly clear separation on OFFICEBENCH and SCIWORLD. At comparable coverage, it generally maintains higher reference-action agreement; conversely, it can route more states to smaller models at a similar precision level. These results support a better empirical precision-coverage trade-off.

5.4 ABLATION STUDY

To identify the sources of coverage and precision, Table 2 separates execution adaptation from routing-branch removal. Learned scores, thresholds, and originally selected model-memory pairs remain fixed, without retraining. Disabled routes fall back to the reference. Removing the SCIWORLD execution adaptation for the same 36 pairs reduces memory safety from 24/36 to 17/36 and strict rescues from 15 to 11; aggregate routed precision falls from 82.2% to 79.7%. Disabling Nano memory routes reduces aggregate memory support from 56 to 29 and strict rescues from 18 to seven. Removing the complete memory branch leaves 220 no-memory routes and no strict rescues, with coverage falling from 21.0% to 16.7% despite higher routed precision. Thus memory expands coverage beyond the no-memory gate, and Nano contributes substantially to the observed rescues.

5.5 DECOUPLED ANALYSIS

Shared Low-Rank Prediction Heads. We test whether shared low-rank prediction heads improve prediction of model-specific memory outcomes under limited supervision. Using four candidate models with matched input representations, we compare five configurations: shared-plus-bias, shared heads with rank-one, rank-two, or rank-three residuals, and fully independent nonlinear heads. Table 3 shows that rank-one sharing achieves the highest mean pair-safety F1 across five training subsamples: 28.29%, compared with 27.78% for shared-plus-bias and 27.57% for independent heads. Increasing the rank does not improve performance. These results suggest that a compact model-specific residual can usefully balance shared statistical strength and cross-model flexibility.

Gain-based shortlisting. To test whether gain ranking preserves useful routing options beyond textual similarity, Table 4 fixes Stage II and compares hybrid ranking, gain ranking, and all-three

Table 3: Four-model sharing: macro pair-safety F1 (%), for one fixed subsample and averaged over five subsamples.

Structure	Fixed	Mean
Shared + bias	25.99	27.78
Shared + bias + rank 1	**29.83**	**28.29**
Shared + bias + rank 2	27.86	27.09
Shared + bias + rank 3	27.86	27.54
Independent nonlinear	27.86	27.57

Table 4: SciWorld shortlisting (%). Ret.: cheapest-safe-pair retention; Cov./Prec.: coverage/precision. Time: ms/state.

Shortlist	Ret.	Cov./Prec.	ms
Hybrid, $k = 1$	40.9	23.3/83.3	3.24
Gain, $k = 1$	**50.5**	23.8/82.8	3.75
Hybrid, $k = 2$	75.7	23.7/83.5	5.96
Gain, $k = 2$	**78.1**	24.2/82.6	6.69
All three	100.0	24.3/82.7	8.83

evaluation. At shortlist size one, gain ranking retains 50.5% of cheapest-safe opportunities versus 40.9% for hybrid ranking. With two memories, gain ranking approaches all-three coverage (24.2% versus 24.3%) with lower downstream scoring time (6.69 versus 8.83 ms/state). This illustrates a recall-computation trade-off: learned filtering preserves more opportunities at a fixed shortlist size. Setup and uncertainty details appear in Appendix A.4.

5.6 Qualitative Analysis

Figure 5 examines transfer through a matching action prerequisite. The current task concerns a peach plant, while the retrieved trajectory concerns an orange plant; both require filling a jug before watering. Here the jug is empty and the sink is off, so pouring skips a prerequisite. Nano's reference agreement changes from 0/3 without memory to 3/3 with the complete memory and a matched-step pointer, producing `activate sink`.

Current query: peach plant	**Memory: orange plant (excerpt)**
Seed in flower pot 1; empty jug in sink. Sink is off.	put jug in sink → activate sink ... then pour jug into flower pot 1
Reference: activate sink	
Without memory	**With memory + step pointer**
pour water from the jug into flower pot 1	activate sink
0/3 reference-aligned	**3/3 exact reference matches**

SciWorld / Nano. Same prerequisite: fill the jug before watering.

Figure 5: Paired Nano replay on SciWorld: 0/3 aligned actions without memory versus 3/3 exact matches with memory and a step pointer.

6 Conclusion

Memory changes the model-selection problem: the same agent state can require different model capabilities depending on which experience is retrieved, and the same memory can affect candidate models differently. We formulated this dependence as cost-constrained memory-model selection and developed a two-stage framework separating memory gain from routing reliability. Experiments show both substantial memory-enabled routing opportunities and measurable end-to-end savings. Our findings position episodic memory can effectively serve as an input to model allocation.

AI use statement

We used generative AI tools for manuscript editing and coding assistance, including debugging and scripts for processing and visualizing experimental results.

Reproducibility statement

Appendix A details the structural transition dataset construction and splits, model architectures, training objectives, hyperparameters, and anchor context selection. The Experimental Setup and Additional Experimental Details sections specify evaluation datasets, oracle budgets, the reachability criterion, metrics, and the docking protocol.

References

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023.

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

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. WorkArena: How capable are web agents at solving common knowledge work tasks? *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.

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. *arXiv preprint arXiv:2304.03442*, 2023.

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

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid LLM: Cost-efficient and quality-aware query routing. *arXiv preprint arXiv:2404.14618*, 2024.

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. *arXiv preprint arXiv:2310.12963*, 2023.

Dujian Ding, Ankur Mallick, Shaokun Zhang, Chi Wang, Daniel Madrigal, Mirian Del Carmen Hipolito Garcia, Menglin Xia, Laks V. S. Lakshmanan, Qingyun Wu, and Victor Rühle. BEST-Route: Adaptive LLM routing with test-time optimal compute. *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

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 with preference data. *arXiv preprint arXiv:2406.18665*, 2024.

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. RouterBench: A benchmark for multi-LLM routing system. *arXiv preprint arXiv:2403.12031*, 2024.

Yanwei Yue, Guibin Zhang, Boyang Liu, Guancheng Wan, Kun Wang, Dawei Cheng, and Yiyan Qi. MasRouter: Learning to route LLMs for multi-agent systems. *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 2025.

Xiyu Guo, Shan Wang, Chunfang Ji, Xuefeng Zhao, Wenhao Xi, Yaoyao Liu, Qinglan Li, Chao Deng, and Junlan Feng. Towards generalized routing: Model and agent orchestration for adaptive and efficient inference. *arXiv preprint arXiv:2509.07571*, 2025.

Pengfei Zhou, Zhiwei Tang, Yixing Ma, Jiasheng Tang, Yizeng Han, Zhenglin Wan, Fanqing Meng, Wei Wang, Bohan Zhuang, Wangbo Zhao, and Yang You. Agent-as-a-Router: Agentic model routing for coding tasks. *arXiv preprint arXiv:2606.22902*, 2026.

An Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Zilong Wang, Yuedong Cui, Li Zhong, Zimin Zhang, Da Yin, Bill Yuchen Lin, and Jingbo Shang. OfficeBench: Benchmarking language agents across multiple applications for office automation. *arXiv preprint arXiv:2407.19056*, 2024.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. ScienceWorld: Is your agent smarter than a 5th grader? *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 2022.

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

A Reproducibility Details

A.1 Full Experimental Setup

Benchmarks and evaluation unit. We use three agentic environments with different action spaces. OfficeBench contains long-horizon office workflows spanning multiple applications (Wang et al., 2024); SciWorld is an interactive text environment for grounded elementary-science tasks (Wang et al., 2022); and τ-bench evaluates tool-using agents that interact with a simulated user while following domain policies (Yao et al., 2025). Our unit of evaluation is one *next-action decision state*, not native episode success. Each state contains the original task instruction and the complete action-observation history available before the next action. Consequently, all reported "safety" and "alignment" results measure agreement with a fixed reference policy; they must not be interpreted as the native benchmark's execution-based task-success score.

Reference trajectories and eligible states. The source cache was created by rolling out the reference deployment `gpt-5.4` in each environment. At every visited state, five stochastic reference actions were grouped by semantic action equivalence using `gpt-5.4-mini`; we retained a state only when the largest cluster contained at least three actions and fixed its representative as the reference action. We further require a nonempty context and reference. This yields 9,100 eligible states from 750 tasks: 1,559 states/210 tasks in OfficeBench, 6,740/426 in SciWorld, and 801/114 in τ-bench. The consistency filter reduces ambiguous supervision but does not certify that the reference action is environment-optimal.

Memory/query isolation and hybrid retrieval. Within each dataset, we randomly assign 30% of eligible tasks to the memory bank and the remaining 70% to query tasks (seed 42), before any router split. The resulting bank contains one untruncated raw stored trace–the longest available context trace–for each of 225 bank tasks; none of these tasks appears as a query. Retrieval uses views only for indexing: the query has a global task-instruction view and a local (a_{i-1}, o_i) view (a null previous action at the initial state), while each memory has its original instruction and every native local step. For each view, we rank-normalize dense cosine and BM25 scores to $[0, 1]$ and combine them as $0.8s_{\text{dense}} + 0.2s_{\text{BM25}}$, using `text-embedding-3-small`, $k_1 = 1.5$, and $b = 0.75$. Global retrieval recalls 20 trajectories by task-view score. We locally rerank each recalled trajectory by its maximum local-view score, then use $0.5s_{\text{global}} + 0.5s_{\text{local}}$ and retain the top three. Retrieval is run once and IDs, ranks, scores, and matched-step indices are cached. No future query action is indexed. Generation receives the complete current history and the complete stored memory trace, not the index views.

Factorial rollouts. The 525 query tasks contain 6,404 states. For every state we evaluate four contexts (no memory and each of the three retrieved trajectories) with two candidate deployments, `gpt-5.4-nano` and `gpt-5.4-mini`. A single request produces three stochastic completions ($n = 3$, temperature 0.7, at most 512 completion tokens), giving $6{,}404 \times 4 \times 2 = 51{,}232$ model-context cells and 153,696 raw actions. The system prompt asks for exactly one final `Action:` line and explicitly describes a retrieved trajectory as a past example whose entities and arguments must be adapted to the current state. Thus the factorial matrix varies memory and model while holding the query state, reference, prompt template, and decoding parameters fixed.

Action-equivalence labels. We first parse the final `Action:` line and mark an exact match only when the lower-cased tool name and every normalized structured argument match the reference. Remaining outputs are judged in batches of three by `gpt-5.4-mini` at temperature 0. The judge is instructed to assess the same immediate action, to require semantic equivalence of the tool and every material argument, and to reject changes to IDs, entities, dates, amounts, addresses, products, or requested operations. Candidates are labeled independently within the batch. Let $a_{ijk} \in \{0, 1/3, 2/3, 1\}$ be the aligned-sample fraction; the binary routing target is $y_{ijk} = \mathbb{1}[a_{ijk} \geq 2/3]$. We call a memory a *strict rescue* when $y_{ijk} = 1$ but $y_{i0k} = 0$ for the same query and model. Appendix A records the exact truncation used only by the judge and the remaining generation details.

Task-disjoint splits. We split the 525 query tasks 80/20 within each dataset, again with seed 42. The outer training pool has 420 tasks, 5,088 states, and 15,264 query-memory rows; the heldout test set has 105 tasks, 1,316 states, and 3,948 rows. We then reserve 20% of the outer-training tasks within each dataset for validation. The final fit/validation/test partitions contain 335/85/105 tasks, 4,074/1,014/1,316 states, and 12,222/3,042/3,948 memory rows. Each row retains labels for both candidate models; the shared no-memory cell is stored once per state. Tasks never cross bank, fit, validation, or test partitions. Hyperparameters, feature choices, calibration thresholds, and minimum-support rules use fit and validation only.

Metrics. For test state i, let $r_i = 1$ if the policy invokes a candidate model, y_i and a_i be the selected cell's majority-safe label and aligned-sample fraction, and c_i its proxy cost; fallback has $a_i = 1$ and $c_i = c_{\text{ref}}$. We report

$$\begin{aligned} \text{Coverage} &= \frac{1}{N}\sum_i r_i, & \text{Precision} &= \frac{\sum_i r_i y_i}{\sum_i r_i}, \\ \text{OA} &= \frac{1}{N}\sum_i \big[r_i a_i + (1 - r_i)\big], & \text{Saving} &= 1 - \frac{\sum_i c_i}{N c_{\text{ref}}}. \end{aligned} \tag{14}$$

Precision is undefined when no state is routed. Based on the official token prices, we use normalized cost proxies Nano $= 1$, Mini $= 5$, and reference $= 15$. Precision-coverage is primary because abstaining to the valid reference fallback is part of the decision, not a negative-class error. We additionally report memory-branch precision and strict-rescue count. Routed-precision intervals are 95% Wilson intervals; risk-coverage bands use 500 task-level bootstrap replicates so steps from one trajectory remain dependent. Pair-level AUROC, average precision, and precision in the top 5% are relegated to the appendix because they do not evaluate the deployed abstaining policy.

A.2 Baselines and Implementation Details

Baselines. All learned methods use the same task-disjoint split and cached action-equivalence labels. We retain only pair MLP and pair KNN as generic learned controls. Our *Hybrid-LLM-style* query router uses the fit-trained no-memory quality predictor to estimate the gap to the reference; our *RouteLLM-style* router learns a weak-versus-reference agreement probability from the current-state embedding. These are adaptations of the query-level quality-gap and preference-routing ideas (Ding et al., 2024; Ong et al., 2024), not exact reproductions of their original training datasets or quality metrics. Neither accesses a retrieved memory. We also evaluate a test-only zero-shot `gpt-5.4` router and a supervised `Qwen3-1.7B` router (Yang et al., 2025).

Claim-focused memory controls. Memory-aware controls use the same hybrid-retrieval top-three candidate set and the same task-disjoint splits. *Retrieve-then-route* fixes hybrid rank one before applying either the pair MLP or a fixed pair scorer, so the router cannot change memories. *Model-agnostic memory utility* fits one Ridge regressor to the Nano/Mini mean alignment-rate change from no memory, selects one memory independently of model, and then applies that fixed model-specific scorer. *Shared* and *independent* effect controls fit, respectively, one pooled or two model-specific Ridge regressors to this rate change using identical cached query-memory embedding features; their predicted effect is added to the same fixed no-memory probability. Regularization is selected on inner-validation tasks. Fixed policies and label oracles provide lower and upper references. All newly trained scorers remain fixed between validation and test.

Adapted-baseline calibration. For each dataset, the query-only adaptations choose Nano/Mini thresholds on validation; the memory-aware controls keep a fixed no-memory gate and choose only memory thresholds. We maximize validation proxy saving subject to at least five routed states and 80% routed precision; any active memory branch additionally requires at least five memory routes at 80% precision. Failure to satisfy these constraints yields reference-only fallback. This lightweight exploratory grid is shared by the adapted controls. The proposed two-stage router selects its supported operating point using validation tasks. As small validation tails can overfit, observed test precision is reported without claiming a formal 80% risk guarantee. All adapted controls reuse existing rollouts, labels, embeddings, and pair scores; they require no additional LLM calls.

A.3 Additional Reproducibility Details

Dataset accounting. Table 5 gives the exact task and state counts after reference-consistency filtering. The 30% memory-bank split is disjoint from all query columns. Each query state contributes three retrieved-memory rows and four generation contexts after including the shared null-memory context.

Table 5: Task-disjoint data splits; fit/validation/test columns count decision states.

Dataset	Bank tasks	Query tasks	Query states	Fit	Validation	Test
OfficeBench	63	147	1,132	731	187	214
SciWorld	128	298	4,737	2,986	753	998
τ-bench	34	80	535	357	74	104
All	225	525	6,404	4,074	1,014	1,316

Reference and judge implementation. Reference consistency uses five stochastic `gpt-5.4` samples and a three-of-five semantic cluster threshold. The factorial candidate rollout uses three stochastic samples and a two-of-three safe threshold. The strict parser takes the last `Action:` line; parseable calls are equal only when the lower-cased function name and recursively normalized JSON arguments are identical. When all three candidates strictly match, no judge call is made. Otherwise one `gpt-5.4-mini` request labels all three candidates, with an explicit instruction to judge each independently. The judge sees the first task line plus the last 3,000 characters of state, and up to 2,500 characters each for the reference and candidate.

Baseline hyperparameters. Logistic baselines standardize their inputs and select $C \in \{.01, .1, 1, 10\}$ separately for each model by validation AUROC plus AP. The direct pair MLP has widths 96 and 32, GELU activations, dropout .15/.10, and class-weighted BCE; AdamW uses learning rate 8×10^{-4}, weight decay .003, at most 350 epochs, and patience 40. Pair LightGBM uses 350 trees, learning rate .025, minimum child support 35, feature fraction .75, $\ell_2 = 5$, and selects leaves from $\{7, 15, 31\}$. Static KNN searches $k \in \{5, 15, 50\}$ and query-similarity weight $\{.25, .5, .75\}$, with the complementary weight on memory similarity and softmax temperature .08. Pair baselines use the same 64-dimensional projected query and memory embeddings and their absolute-difference/product interactions; retrieval-only logistic receives global, local, combined, rank, and matched/query-step features. These choices are selected on the inner validation tasks. Their Table 1 operating points use the validation-only safety-constrained search described above.

LLM-router hyperparameters. The zero-shot router receives the complete current history and all three complete retrieved trajectories in one `gpt-5.4` call per outer-test state (temperature zero, 700 completion-token limit). Its prompt directly selects Nano, Mini, or reference and at most one memory, requires an estimated majority-safety probability of at least .8 before selecting a candidate model, and otherwise requests reference fallback. It receives no rollout labels, demonstrations, validation examples, or reference action. The SFT router predicts majority-safe versus unsafe separately for each candidate-model/context cell. We LoRA-tune `Qwen3-1.7B` for one epoch on a seed-42, dataset-model-label-stratified sample of 8,000 cells from the 32,592 fit cells after balancing the two labels within each dataset and candidate model. LoRA uses rank 8, scale 16, dropout .05, and all linear layers; AdamW uses learning rate 10^{-4}, effective batch size 16, cosine decay, 5% warmup, and weight decay .01. Inputs use deterministic head-tail compaction to 1,536 tokens and non-thinking mode. We assign equal-length single-token targets A (safe) and B (unsafe), and score their next-token softmax in one forward pass for every one of the 8,112 validation and 10,528 test cells. The policy then selects the cheapest eligible model/context configuration using a single validation-selected threshold; the outer-test split is scored once.

Metric details. "OA" in Table 1 is the expected sampled-action alignment of the deployed policy under the three cached draws, with reference fallback assigned alignment one by definition. It is not classification accuracy. Pair-level AP is computed independently for Nano and Mini and macro-averaged; P@5 is precision among the top 5% scored query-memory rows for each model, also macro-averaged. Appendix Table 7 uses outer-test labels and is strictly diagnostic. All rollout

Table 6: Heldout memory-rescue opportunities and model-specific effects. Ref.-req.: states unsafe for both models without memory.

Dataset	Ref.-req.	Rescue	Frontier	Specific	Opposite
OFFICEBENCH	82	70.7%	39.7%	48.1%	4.5%
SCIWORLD	416	45.7%	30.2%	43.0%	4.0%
TAUBENCH	45	48.9%	29.8%	42.6%	2.9%
All	543	49.7%	31.7%	43.8%	4.0%

texts, per-sample labels, judge reasons, retrieval metadata, relation-concept scores, probabilities, thresholds, and final decisions are retained in the experiment artifact.

A.4 COMPONENT STUDY DETAILS

Four-model structure control. Table 3 uses Nano, Mini, GPT-4.1-mini, and GPT-4o-mini with 25% of fit-task labels (261 pairs in the fixed subsample), evaluated on 108 SCIWORLD test pairs from 36 tasks. Rank-one's mean advantage over independent heads is 0.71 percentage points, with a task-bootstrap 95% confidence interval of $[-0.57, 2.32]$.

Stage-I shortlist control. Table 4 fixes Stage II and the original execution prompts on 998 SCIWORLD states. Retention measures preservation of a cheapest-safe pair among the 301 states where memory lowers the cheapest safe cost. Timing includes gain scoring, historical-evidence aggregation, and both Stage-II heads with cached upstream features, excluding retrieval and LLM execution; it uses 24 states and four warmed repeats. Gain shortlisting improves retention over hybrid at fixed k, but does not match all-three recall.

A.5 FIGURE PROTOCOLS

Paired replay. Figure 5 shows one failed action and an abbreviated historical excerpt from a selected heldout state. Generation uses the complete current history and retrieved trajectory.

B HELDOUT EVIDENCE FOR THE CENTRAL ASSUMPTION

We test whether the routing opportunity posited in Section 1 is present in the heldout factorial rollout matrix. A state is *reference-required* when neither Nano nor Mini is majority-safe without memory. A *downgrade rescue* occurs when at least one of its six top-three memory-model cells is safe. A *frontier improvement* occurs whenever memory lowers the cheapest safe action from its no-memory value; this cost-based diagnostic is distinct from the alignment gain in Equation 3. Finally, a query-memory pair has a *model-specific effect* when adding that same memory changes the safe/unsafe label for exactly one candidate model; an *opposite effect* rescues one model while harming the other. Rescue rates condition on reference-required states, frontier rates on all states, and effect rates on query-memory pairs.

The opportunity is substantial but conditional. Memory rescues 270 of 543 reference-required states (49.7%; 95% task-bootstrap CI $[45.1, 54.9]$) and improves the cheapest safe frontier on 417 of 1,316 states (31.7%; $[28.9, 34.9]$). Yet 43.8% of the 3,948 query-memory pairs change only one model's outcome ($[41.7, 45.8]$), and 159 pairs (4.0%; $[3.3, 4.8]$) rescue one model while harming the other. Rank-one retrieval captures only 46.0% of the gains available in the top three. These results support both premises of dual routing–memory can create a downgrade opportunity, and its effect depends on the selected model–while also showing why similarity-only rank-one routing is insufficient.

C LABEL-ORACLE UPPER BOUNDS

Table 7 reports non-deployable headroom obtained by selecting routes with outer-test majority-safe labels. These rows are separated from the main comparison because they use information unavailable to a router at deployment.

Table 7: Label-oracle upper bounds using test labels (not deployable). Metrics follow Table 1, reported as fractions.

Policy	OFFICEBENCH ($n = 214$) Cov.	Prec.	OA	Save	SCIWORLD ($n = 998$) Cov.	Prec.	OA	Save	TAUBENCH ($n = 104$) Cov.	Prec.	OA	Save
No-memory label oracle	.617	1.000	.942	.508	.583	1.000	.943	.456	.567	1.000	.936	.471
Hybrid rank-zero label oracle	.528	1.000	.928	.422	.246	1.000	.955	.203	.317	1.000	.965	.255
Hybrid top-three label oracle	.799	1.000	.880	.680	.518	1.000	.898	.432	.644	1.000	.910	.542

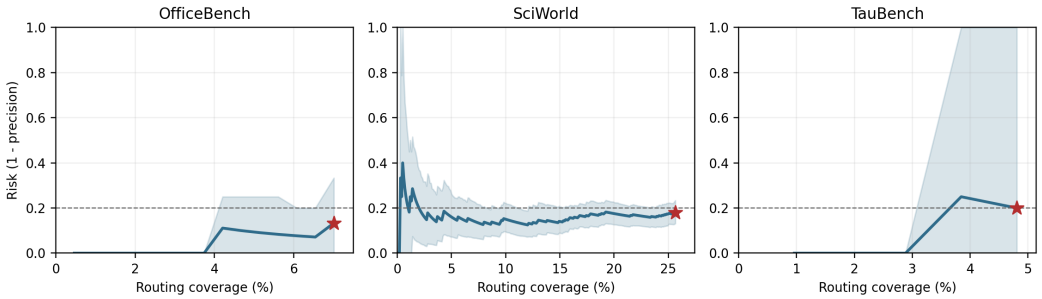

Figure 6: Risk-coverage for the deployed router. Bands: 95% task-bootstrap intervals; stars: deployed policies; dashed line: 20% risk.

D CALIBRATION SUPPORT AND UNCERTAINTY

Table 8 reports the support for the same final policy as Tables 1-2. The saved thresholds and execution prompt are selected on validation. Wilson intervals quantify binomial uncertainty over routed states and are widest on TAUBENCH. The aggregate memory-conditioned branch has 28/40 safe validation routes and 41/56 safe test routes, with 18 strict test rescues. Overall validation and test safety are 150/183 and 227/276, respectively.

Table 8: Routing support and 95% Wilson intervals. Mem.: memory routes; Rescue: same-model unsafe-to-safe transitions.

Dataset	Val. safe/routes	Test safe/routes	Test precision [95% CI]	Mem. safe/routes	Rescue
OFFICEBENCH	10/12	13/15	.867 [.621, .963]	13/15	3
SCIWORLD	135/165	210/256	.820 [.769, .862]	24/36	15
TAUBENCH	5/6	4/5	.800 [.376, .964]	4/5	0
All	150/183	227/276	.822 [.773, .863]	41/56	18

Figure 6 tightens the final router's gates on the heldout split; the stars are the same deployment points as Table 1. Confidence bands resample complete heldout tasks to preserve within-trajectory dependence, using the same score-margin cutoffs in every replicate and omitting replicates with no routed state at that cutoff. At the selected coverage levels, the 95% task-bootstrap risk intervals are $[0, .333]$, $[.132, .234]$, and $[0, 1]$ for OFFICEBENCH, SCIWORLD, and TAUBENCH, respectively.

E MEMORY-BRANCH COMPARISON

Figure 7 isolates memory-conditioned decisions from the end-to-end curves in Figure 4. The frozen proposed scorer and memory choices are unchanged as its memory threshold varies. Query-only routers have no memory branch and are therefore compared in the main end-to-end figure.

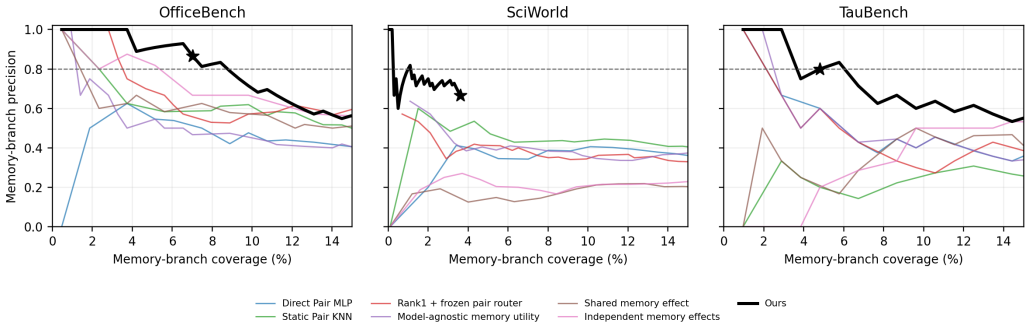

Figure 7: Memory-only precision-coverage. Stars: deployed memory branches; dashed line: 80% precision. Coverage is over all states.

Table 9: Heldout examples of memory rescue, memory harm, and unsafe routed decisions.

ID	Type	Dataset	Branch	Reference action	Historical next action
R1	rescue	OfficeBench	counterfactual-memory	python3 /apps/excel_app/e...	python3 /apps/excel_app/e...
R2	rescue	SciWorld	counterfactual-memory	look around	look around
R3	rescue	TauBench	counterfactual-memory	Please reply yes to submi...	respond
R4	rescue	OfficeBench	counterfactual-memory	python3 /apps/excel_app/r...	python3 /apps/excel_app/r...
H1	harm	OfficeBench	counterfactual-memory	python3 /apps/excel_app/e...	python3 /apps/excel_app/e...
H2	harm	SciWorld	counterfactual-memory	look around	look around
H3	harm	TauBench	counterfactual-memory	modify_pending_order_item...	modify_pending_order_items
H4	harm	OfficeBench	counterfactual-memory	python3 /apps/excel_app/e...	python3 /apps/excel_app/e...
F1	false_positive	OfficeBench	memory	python3 /apps/excel_app/w...	python3 /apps/excel_app/r...
F2	false_positive	OfficeBench	memory	python3 /apps/excel_app/e...	python3 /apps/excel_app/e...
F3	false_positive	SciWorld	no_memory	open door to kitchen	–
F4	false_positive	SciWorld	no_memory	open door to bedroom	–

F Qualitative Rescue, Harm, and Failure Cases

Table 9 samples four Mini rescue cases, four Mini harm cases, and four unsafe deployed routes from heldout states. The direct-transfer rescues include matching spreadsheet tool sequences and exact environment actions. The harm cases show that even an apparently matching historical next action can destabilize the sampled continuation, so action overlap alone is not a sufficient compatibility test. One TauBench rescue uses a semantically unrelated memory yet happens to produce the reference action; this illustrates why an observed unsafe-to-safe transition is evidence of association rather than proof that the memory causally supplied the capability. The deployed false positives include both a memory-conditioned tool mismatch and confident no-memory navigation errors.