NO SINGLE MEMORY MECHANISM WINS: A SYSTEMATIC EVALUATION OF MEMORY ROUTING POLICIES FOR LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon agents accumulate experience across sessions in memory mechanisms such as vector indexes, knowledge graphs, summary trees, and strategy banks. These mechanisms are usually evaluated separately, each on the regime it targets. In deployment, however, an agent faces a mixed stream of requests and must decide, per request, which form of memory to rely on. Whether query semantics are sufficient to guide this choice, how much can be gained by choosing well, and whether a routing policy transfers as the request mix shifts remain unclear. In this work, we present an empirical evaluation of memory routing for LLM agents over 2,614 requests and 12 memory substrates. We first examine why routing based on query semantics alone may fail for memory selection: semantically similar requests can still favor different memory mechanisms. We then assess the potential benefit of choosing among these mechanisms instead of relying on a fixed one, including whether this benefit persists within an individual memory. Finally, we compare routing strategies with different information requirements and examine their generalization beyond familiar memories and tasks. Under the reported convention, the recorded oracle scores 64.74 against 36.82 for the best fixed substrate, and the best policy we measure reaches 44.00, thereby recovering about a quarter of that gap. Straightforward strategies keyed on the task source capture most of this gain, which supports memory selection as a direction and coarse task adaptation as a starting point. Furthermore, how to make finer-grained choices for individual requests and transfer this ability to unfamiliar settings remains an open research question.

1 INTRODUCTION

Long-horizon agents accumulate experience across sessions as conversational facts, document knowledge, and procedures from past tasks (Maharana et al., 2024; Wu et al., 2025; Ouyang et al., 2026). Memory makes this experience available beyond a single context window (Packer et al., 2023; Chhikara et al., 2025). As histories grow, the agent must expose the information a request needs without reading everything it has stored. The choice of memory mechanism thus shapes both what the agent can recall and how it uses its experience.

Memory designs support different patterns of use. For instance, vector and keyword indexes retrieve isolated facts, while knowledge graphs and summary trees connect dispersed evidence. Strategy banks further distill procedures from past trajectories (Huang et al., 2026; Guo et al., 2025b; Sarthi et al., 2024; Ouyang et al., 2026). Generally, these designs are evaluated in the regimes they target, with conversational recall, document reasoning, and embodied task reuse benchmarked separately (Maharana et al., 2024; Hu et al., 2026; Shridhar et al., 2021). In deployment, however, a single assistant may encounter all three. Choosing a mechanism at design time therefore leaves open whether another would better serve the current request. This motivates *memory routing*, i.e., the per-request choice of memory.

Model and agent routers address a related decision: who should perform a task (Ong et al., 2025; Chen et al., 2024; Yue et al., 2025; Qian et al., 2025). Memory routing concerns the experience representation supplied to that performer. Some memory routers already adjust retrieval or context

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.

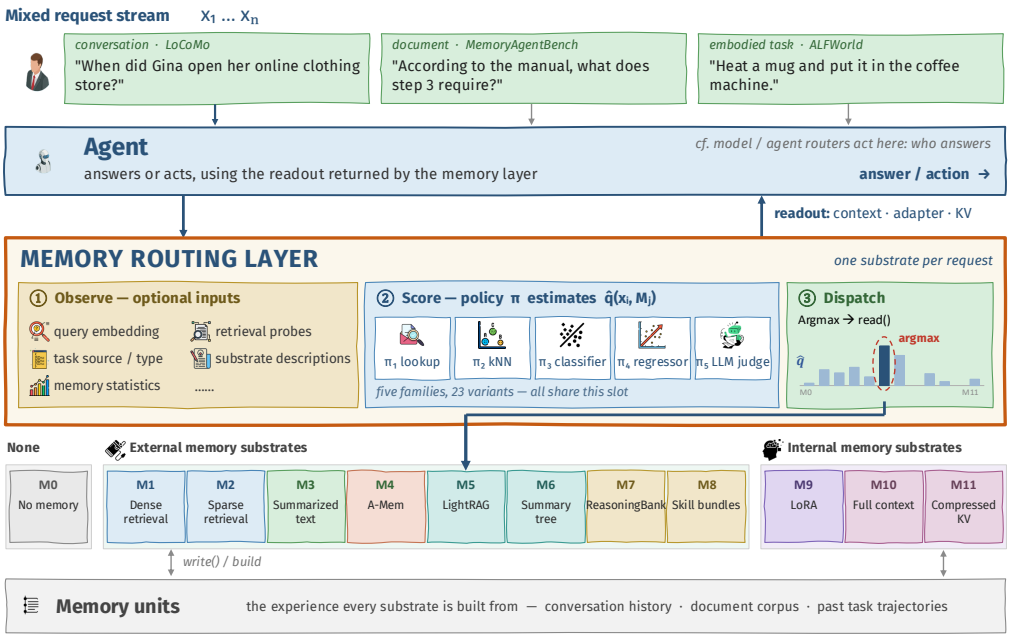


Figure 1: **Memory-routing framework.** A routing layer sits between the agent and twelve memory substrates (M_0–M_{11}) that store the same experience in different forms. For each request from a mixed stream, it observes routing-time signals (query embedding, task source and type, memory statistics, retrieval probes, substrate descriptions), scores the eligible candidates with a policy π, and reads the selected substrate; the readout is returned to the agent. Model and agent routers choose who answers; the memory router chooses what the agent reads.

selection within a store (Tobkin & Yang, 2026; Chen et al., 2026; Liu et al., 2025), and another study chooses between LoRA and RAG for personalization (Deng, 2026). These studies motivate adaptive memory use, but do not quantify the choice across the broader range of agent memory mechanisms against a common oracle. Figure 1 summarizes our evaluation setting and its three questions: why semantic routing may fail for memory selection, how much headroom substrate selection offers, and which policies might recover it.

First, we ask **why semantic routing may fail for memory selection**. Semantic routers infer routes from query representations, but memory selection can also depend on the task, the state of the memory, and how each mechanism stores and retrieves experience. We therefore compare the outcomes of semantically similar requests and use their disagreement to motivate *substrate-based routing*: selecting among the forms in which experience is stored and read back. We evaluate 12 substrates under a common backbone, including no memory. The supplied eligibility mask restricts each request's candidates and also reveals some source information. Figure 2(a) illustrates how the highest-scoring substrate varies across requests.

With memory substrates as the candidate actions, we then ask **how much headroom their selection offers**. We construct a matrix over 2,614 requests, distinguishing completed outcomes, inapplicable pairs, and run-limit failures. Inapplicable pairs and executions that reach the evaluation limit receive zero credit. Under this rule, the single-execution oracle over completed applicable outcomes scores 64.74 against 36.82 for the best fixed substrate, a relative gap of 75.85%. These scores condition on recorded execution states: maximization can select execution noise, and offline selection does not replay an online router's trajectories. Even choosing the best substrate for each memory unit leaves 18.87 of the 27.93 points unclaimed.

Finally, we ask **how a policy should recover this headroom**. We compare five policy families in 23 variants, from lookup tables to prompted language models. Our results show that routing improves over fixed memory mechanisms, and that a simple task-source lookup captures a substantial gain. The remaining headroom lies in generalization: when the router encounters unfamiliar memories or tasks, particularly when an entire competency is held out, richer request and memory signals

yield smaller additional gains. Memory selection is thus effective but far from saturated, and further improvement calls for finer-grained routing designs that generalize to new memories and tasks.

2 Related Work

Agent Memory. Memory systems for agents differ mainly in how experience is stored and read back (Huang et al., 2026). Generally, *external memory* keeps experience outside the model. One line of work retrieves history chunks from dense or sparse indexes (Packer et al., 2023; Chhikara et al., 2025). Another line organizes the history into knowledge graphs and summary trees, thereby supporting reasoning across chunks (Guo et al., 2025b; Gutiérrez et al., 2024; Sarthi et al., 2024). Further designs maintain notes that are linked and rewritten as new experience arrives (Xu et al., 2025), or distill strategies and skills from past trajectories (Ouyang et al., 2026; Zhang et al., 2026). In contrast, *internal memory* places experience inside the model, e.g., in the context window, in a LoRA adapter trained on the history (Hu et al., 2022), or in a compressed attention state. However, these systems are evaluated on separate benchmarks, e.g., LoCoMo and LongMemEval for long conversations (Maharana et al., 2024; Wu et al., 2025), MemoryAgentBench for retrieval, test-time learning, long-range understanding, and conflict resolution (Hu et al., 2026), and ALFWorld for the reuse of task experience (Shridhar et al., 2021). Consequently, the design space of memory mechanisms is broad, and no single mechanism is uniformly superior, which turns the choice among them into a non-trivial problem for a deployed agent.

LLM Routing. LLM routers decide which model should serve a request to trade quality against cost. Representative approaches learn from preference data (Ong et al., 2025), cascade from cheap to expensive models (Chen et al., 2024), or classify the domain or intent of the request (Tran et al., 2025; Wang et al., 2025). To benchmark them, existing studies precompute every model on every query, take the per-query best as an oracle, and score routers by their distance to it (Hu et al., 2024; Huang et al., 2025; Lu et al., 2026; Li et al., 2026). Agent routers further extend the idea to pools of agents (Yue et al., 2025; Qian et al., 2025; Guo et al., 2025a; Cui et al., 2026; Feng et al., 2026). These approaches route among models or agents. Memory and context routing instead determines which information reaches a model. Retrieval-level methods decide whether, how, or how deeply to retrieve from a single store (Asai et al., 2024; Jeong et al., 2024; Tobkin & Yang, 2026; Chen et al., 2026); RCR-Router selects subsets of a shared memory for each agent according to its role, task stage and token budget (Liu et al., 2025). Another work chooses between LoRA and RAG in a user-personalization setting (Deng, 2026). Our study compares heterogeneous memory mechanisms from a more systematic manner.

3 The Request Pool

Agent memory is usually evaluated one benchmark at a time, with each system tested on the kind of experience it was designed to store. This assumes that the appropriate memory mechanism is known before a request arrives. A deployed agent instead faces a mixed stream of questions and tasks. We therefore combine three benchmarks into one request pool and use the same evaluation protocol throughout.

Three Kinds of Experience. A long-horizon agent accumulates facts from conversations, knowledge from documents, and procedures from completed tasks. LoCoMo (Maharana et al., 2024) contributes 1,975 questions after we remove 11 repeated question–category entries within a dialogue. MemoryAgentBench (Hu et al., 2026) contributes 371 requests across four document-memory competencies. ALFWorld (Shridhar et al., 2021) contributes 134 embodied tasks, each evaluated with an offline trajectory bank and with online accumulation, for 268 requests. The resulting pool contains 2,614 requests from 15 types. We use *benchmark* for the three datasets and *task source* for six groups: LoCoMo, the four MemoryAgentBench competencies, and ALFWorld. Appendix D.1 details the retained categories and history construction; Table D.1 there summarizes request and memory-unit counts.

Requests and Memory Units. A request identifies a question or task instance together with its evaluation context. The pool has 25 *memory units*: a LoCoMo dialogue, a MemoryAgentBench

document, or an ALFWorld suite under one memory regime. Question-answering requests in a unit share a fixed history. Online ALFWorld instead starts empty and writes a trajectory after each episode, so its state can differ across episodes and substrate executions. Our offline comparison conditions on each recorded execution state; choosing another substrate does not replay the preceding episodes of an online switching policy. Identical text can likewise denote different requests across task instances, memory regimes, or type annotations. Appendix D.1 describes these cases.

Backbone and Evaluation Setup. The main experiments use *Qwen3.5-9B* with greedy decoding, at most 512 generated tokens, and a 32K effective context; 54% of requests have longer histories. The default auxiliary LLM and encoder are *gpt-4o-mini* and *text-embedding-3-small*. Metadata-based routers receive source and type annotations, and every router receives the profile-eligibility mask. Because ReasoningBank is restricted to ALFWorld, this mask also reveals that source. Appendix A specifies configurations and metrics. Section 6.1 defines the text-grouped and held-out-group protocols.

4 Why Semantic Routing May Fail

Semantically Similar Requests Can Favor Different Memories. Many LLM and agent routers treat routing as a function of query semantics. This is natural when routes correspond to model capabilities, domains, or intents; vLLM Semantic Router, for example, classifies requests from sentence representations (Wang et al., 2025).

To ask whether the same basis is sufficient for memory, we represent each request with normalized *all-MiniLM-L6-v2* sentence embeddings (Reimers & Gurevych, 2019). Semantic proximity, however, does not imply the same task, memory state, or useful memory mechanism. Table 1 makes this mismatch concrete, and Appendix B.1 gives the encoder, the thresholds and the pair counts behind it. Even at cosine similarity at least 0.9, 32% of pairs cross request types and 64% differ in success under at least one framework; among pairs individually solvable by some framework, 19% share no successful framework. These pooled descriptive rates show why semantic similarity alone may be insufficient for choosing memory. Representative examples and full definitions appear in Table B.1.

Table 1: Highly similar request pairs.

	$\cos \geq .8$	$\cos \geq .9$
Request pairs	8,435	1,740
Different request types	54%	32%
Dense retrieval[a]	18%	14%
A-Mem[b]	16%	10%
LightRAG[c]	31%	29%
Summary tree[d]	36%	33%
ReasoningBank[e]	28%	33%
Any framework	66%	64%
No framework succeeds on both	18%	19%

Methods: [a] Packer et al., 2023; Chhikara et al., 2025; [b] Xu et al., 2025; [c] Guo et al., 2025b; [d] Sarthi et al., 2024; [e] Ouyang et al., 2026.

Memory-Substrate-Based Routing. Because query semantics alone may not identify the appropriate memory, routing can also use task information and differences in storage and readout. Following Huang et al. (2026), we call each such form a *memory substrate*, and choosing one per request *memory-substrate-based routing*. Table 2 shows that the candidates differ in both retained state and how it is exposed to the backbone, so routing is broader than choosing a retriever. Appendix D.2 gives the extended discussion.

Table 2: Memory substrates in our evaluation.

Type	Instances	Stored memory	Readout
No memory	M_0	none	none
Flat retrieval	dense (M_1), sparse (M_2)	history chunks	dense or BM25 top-k
Summary index	summarized text (M_3)	chunk summaries	gist-page selection
Evolving notes	A-Mem (M_4)	linked, rewritable notes	agentic search
Structured	LightRAG (M_5), summary tree (M_6)	entity graph or summary tree	graph–vector or hierarchical search
Procedural	ReasoningBank (M_7), skill bundles (M_8)	strategies or skills	strategy or skill retrieval
Parametric	LoRA (M_9)	adapter weights	adapter generation
Context-native	full context (M_{10}), compressed KV (M_{11})	raw history or compressed KV	full context or episode reprefill

5 How Much Headroom Does Memory Routing Offer?

We next ask how much is at stake in choosing among memory substrates. We compare a fixed substrate with a per-request oracle; their difference, which does not establish what a deployed router can reproduce, measures headroom on recorded outcomes.

5.1 Outcome Matrix and Scoring

Let $\mathcal{X} = \{x_i\}_{i=1}^{n}$ be the $n = 2{,}614$ requests and $\mathcal{M} = \{M_0, \ldots, M_{11}\}$ the substrates. The masks $a_{ij}, o_{ij} \in \{0, 1\}$ record profile eligibility and completion, with $o_{ij} \leq a_{ij}$. They define eligible candidates $\mathcal{M}_i = \{M_j : a_{ij} = 1\}$ and candidates with a completed task score $\mathcal{O}_i = \{M_j : o_{ij} = 1\}$. Eligibility includes configuration and budget restrictions; it does not establish intrinsic substrate compatibility (Appendix A). For a completed cell, $q(x_i, M_j) \in [0, 1]$ is task success for ALFWorld, token-level F1 for LoCoMo, or a competency-specific MemoryAgentBench score. Accurate retrieval permits answer-item partial credit; long-range understanding uses adapted substring matching. Appendix A specifies these metrics. The raw matrix $\mathbf{Q} \in ([0, 1] \cup \{\perp\})^{n \times 12}$ retains scores together with status masks. Its 31,368 cells comprise 27,202 completed outcomes, 4,096 inapplicable pairs, and 70 eligible ALFWorld online cells affected when two suite-level runs reached the wall-clock budget. The last group is treated as run-limit failures and receives zero credit. Figures B.4 and B.5 in Appendix B.21 visualize the scores and status masks.

To score offline selections, let $\widetilde{q}(x_i, M_j) = q(x_i, M_j)$ if $o_{ij} = 1$ and zero otherwise, giving matrix entries $\widetilde{Q}_{ij} = \widetilde{q}(x_i, M_j)$. For a run-limit cell, zero is the prescribed failure score under the evaluation budget; inapplicable cells also receive zero in aggregate reporting. A policy receives the eligibility mask, but not held-out completion status, and chooses $\pi(x_i) \in \mathcal{M}_i$. Its score is

$$S(\pi) = 100 \sum_{i=1}^{n} w_i \, \widetilde{q}\big(x_i, \pi(x_i)\big). \tag{1}$$

The weights sum to one, giving equal weight to the three benchmarks and, within MemoryAgentBench, to its four competencies. This prevents LoCoMo's 76% share of requests from dominating the aggregate. All reported scores use the 0–100 scale; display rounding can change the last digit of derived quantities. The score also depends on the recorded execution state s_{ij}. In online ALFWorld, this includes trajectories written by earlier episodes of that substrate execution. Since recombining columns does not generate an online switching policy's histories, our scores evaluate offline selection conditional on the recorded states (Appendix C).

The fixed and oracle references are

$$S(M_j) = 100 \sum_{i} w_i \widetilde{q}(x_i, M_j), \qquad S_{\text{fix}} = \max_{M_j \in \mathcal{M}} S(M_j), \tag{2}$$

$$S_{\text{orc}} = 100 \sum_{i} w_i \max_{M_j \in \mathcal{O}_i} q(x_i, M_j), \tag{3}$$

with an empty maximum defined as zero. Fixed references receive zero credit on excluded pairs or run-limit failures. Maximizing over completed candidates gives the same value as including run-limit failures at score zero. Since every policy is scored with the same rule, $S(\pi) \leq S_{\text{orc}}$ under the stated execution budget and recorded states. A larger budget or a different online rollout defines a different evaluation. The *headroom* is

$$H = S_{\text{orc}} - S_{\text{fix}}. \tag{4}$$

5.2 Headroom Analysis

The recorded outcome matrix contains substantial routing headroom. LightRAG is the best fixed substrate at 36.82, while the observed-outcome oracle reaches 64.74, yielding a 27.93-point gap, or 75.85% relative to the fixed score. Table 3 compares all twelve substrates when each is used without routing and when perfect routing selects among them. For the latter, each request with a positive observed score is assigned to an observed applicable substrate attaining its maximum, with ties split equally. Requests with no positive recorded outcome account for 25.43% of the

evaluation weight and remain unrouted. Every substrate receives between 2.54% and 10.31% of the weight, so the recorded optimum is distributed across the library; the summary tree makes the largest individual contribution, at 9.35 points. Tables B.3 and B.4 in Appendix B.2 give the benchmark-level breakdown.

Table 3: Scores without routing and with perfect routing. *Routed* is the evaluation-weighted assignment share; contributions sum to S_{orc}.

		Without routing	With perfect routing		
	Substrate	$S(M_j)$	Routed (%)	Score	Contribution
M_0	No memory	15.62	3.30	97.20	3.21
M_1	Dense retrieval	30.83	7.74	85.61	6.62
M_2	Sparse retrieval	28.74	6.95	87.73	6.10
M_3	Summarized text	27.84	8.48	80.82	6.86
M_4	A-Mem	17.94	4.24	83.16	3.52
M_5	LightRAG	**36.82**	10.04	86.82	8.71
M_6	Summary tree	36.11	10.31	90.75	9.35
M_7	ReasoningBank	12.94	5.04	100.00	5.04
M_8	Skill bundles	26.24	8.55	83.62	7.15
M_9	LoRA	17.50	4.77	93.17	4.44
M_{10}	Full context	5.74	2.54	62.37	1.58
M_{11}	Compressed KV	12.36	2.63	81.95	2.15
		$S_{\text{fix}} = \mathbf{36.82}$			$S_{\text{orc}} = \mathbf{64.74}$

Most of this headroom remains within individual memories. Even assigning each memory unit its best substrate reaches only 45.88, leaving 18.87 points to request-level selection; 19 of the 25 units retain more than five points of internal headroom (Table B.7 of Appendix B.5). This makes coarse task adaptation useful but incomplete: further gains require distinctions among requests that share the same source and memory, as visualized in Figure 2 in Section 6.2. These are descriptive, in-sample gaps conditional on recorded executions; Appendices B.4, B.13 and F.1 report the uncertainty, scoring sensitivity and remaining granularity details.

6 How Should a Memory Routing Policy Be Structured?

The oracle in Section 5 measures what substrate selection could offer on the recorded outcomes. To realize this gain, a policy must choose before seeing the answer. We therefore compare policies with different information and supervision requirements under two forms of test-time novelty.

6.1 Policies and Protocols

Perfect routing selects $\arg\max_{M_j \in \mathcal{O}_i} Q_{ij}$ from completed, applicable outcomes. A practical policy instead estimates these scores before answering and selects from $\mathcal{M}_i$, the evaluation-profile candidate set defined by eligibility rather than recorded-score availability:

$$\pi(x_i) = \arg\max_{M_j \in \mathcal{M}_i} \hat{q}\big(x_i, M_j\big), \tag{5}$$

where $\hat{q}$ uses information available when the request arrives. Training scores use $\widetilde{\mathbf{Q}}$ under Section 5.1's zero-credit convention; the raw matrix and masks retain completion status. We compare five implementations, detailed in Appendix E.1.

Five Policy Families.

- π_1**: lookup.** Each training-fold metadata key stores per-substrate arithmetic mean scores, using either task source or source and request type. Selection respects the eligibility mask and uses no held-out outcomes; we retain these variants only for new requests under known memory, where the comparison does not require specifying an absent-key fallback.
- π_2**: nearest neighbors.** We encode requests with *all-MiniLM-L6-v2* (Reimers & Gurevych, 2019) and average the recorded scores of the $k = 25$ nearest training requests by cosine similarity. Variants add memory-unit statistics and dense/sparse retrieval probes to query matching.
- π_3**: encoder and classifier.** A logistic-regression classifier predicts request category from the same embeddings, then maps it to a substrate through the lookup, following vLLM Semantic

Router (Wang et al., 2025). We also test the *semantic-router* library, an MLP category classifier and an MLP that predicts the substrate directly.

- π_4**: score regression.** We fit one LightGBM regressor per substrate using query embeddings e, memory statistics u and retrieval probes p, separately or together. The probes summarize dense and sparse retrieval performed before answering; two MLP variants use the same feature blocks.
- π_5**: LLM judgment.** We prompt a model with the request and substrate descriptions to select directly, using greedy decoding. We test the backbone and seven external judges; two additional reasoning models are excluded because the generation budget leaves their responses empty (Appendix B.8).

Together, these families yield 23 variants. Except for π_5, policies use only training-fold cells, from category averages to individual request–substrate outcomes. The six task sources comprise LoCoMo, four MemoryAgentBench competencies and ALFWorld, distinct from the three benchmarks. The lookup, π_2 and π_4 receive explicit metadata, while every policy receives the eligibility mask. Since ReasoningBank is eligible exactly on ALFWorld, this mask identifies the source even without an explicit source feature. Feature-block names therefore do not imply strictly query-only information (Table B.17).

Protocols and Scope. The first protocol evaluates new requests under known memory, keeping identical request texts in one fold. The second holds out clusters from 29 groups: ten LoCoMo memory units, thirteen MemoryAgentBench units and six ALFWorld task families. These groups encode different novelty: unseen memory units for LoCoMo and long-range understanding, an entire competency for the other three MemoryAgentBench sources, and a task family with both memory states still represented for ALFWorld. Neither lookup is retained in this grouped comparison.

Grouped folds also exclude training requests whose text occurs in the test fold. Both protocols score selections from the same recorded matrix; comparison intervals bootstrap the 29 groups rather than the 25 memory units alone. We report recovered headroom as $\frac{S(\pi)-S_{\text{fix}}}{H}$, with 0% for no routing and 100% for perfect routing. Appendix E.2 gives the overlap and fold details. Under ALFWorld online accumulation, earlier episodes and substrate choices affect later memory states. Matrix recombination is conditional on those recorded histories; it does not execute the router as an online policy. Deployment requires separate rollouts with a specified history-update protocol.

6.2 Empirical Analysis

A Task-Source Lookup Captures Most of the Observed Policy Gain. Table E.1 compares the 23 main variants over the full twelve-substrate library, subject to eligibility. The best reaches 44.00 against 36.82 for fixed LightRAG, recovering 25.7% of the single-execution matrix gap. The full oracle remains 20.74 points above the best policy, a gap that does not estimate independently reproducible routing gains. The task source lookup already scores 43.40, accounting for 6.58 of the best policy's 7.18-point gain.

The cross-validated lookup thus remains a strong reference: no main variant demonstrates a positive advantage over it. The best policy differs by $+0.60$ with $[-1.7, +3.1]$; eight comparisons exclude zero, all negatively. Seven of eight LLM judges score below the lookup, in paired intervals that do not establish differences between model generations. Point estimates are non-monotonic across tested models.

Finer partitions expose additional routing headroom. Figure 2(a) illustrates how the maximizing substrate varies across the displayed requests, while panel (b) places the best cross-validated policy on the full-pool granularity ladder. Benchmark- and task-source-level assignments reach 43.66 and 44.40, while memory-unit, memory-unit–type and request-level assignments rise to 45.88, 48.43 and 64.74. The solid curve is selected and evaluated on the same recorded pool, so it provides descriptive in-sample references rather than held-out policy estimates. The growing gains at finer partitions indicate that advancing beyond coarse task adaptation requires distinguishing requests within the same source and memory.

The benchmark-level point assigns ReasoningBank where eligible and LightRAG elsewhere. This full-pool assignment is not an independently selected generalization baseline; using the full library alone does not identify a configuration as one of the 23 variants. Additional configurations also

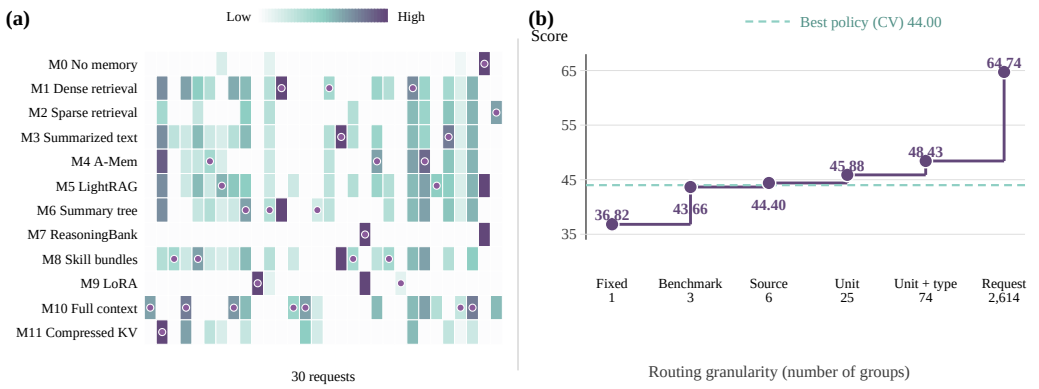


Figure 2: **Request-level variation and routing granularity.** **(a)** Scores of 12 substrates on an illustrative 30-request subset; circles mark a highest-scoring substrate and darker cells indicate higher scores. The subset is drawn from the 100-request sample of Figure B.4 by keeping the requests with no unfinished cell and either a unique maximizing substrate or, where ReasoningBank is eligible, several maximizing substrates, so it is selected for legibility and over-represents requests with a single best substrate relative to the pool. **(b)** Full-pool scores obtained by choosing the best substrate within increasingly fine groups; the dashed line marks the best cross-validated policy (44.00).

lie outside the primary comparison's paired tests: evaluation-weighted training scores 44.47 (Appendix B.18), excluding M_0 scores 44.97 (Table B.13), and the full-pool greedy candidate-set curve peaks at 45.13 (Figure 3(b)).

The Aggregate Gain Is Concentrated in the Task-Source Distinction. Figure 3(a) places the fixed substrates, the best main policy and the recorded oracle on the same scale for all six task sources. ALFWorld contributes $+6.84$ to the lookup's total $+6.58$ gain, while the other sources contribute -0.26 together. For π_4, ALFWorld contributes $+7.21$ to a total of $+7.18$, while the other sources contribute -0.03 together. On the 2,346 question-answering requests alone, the lookup and π_4 score 45.69 and 46.04 against fixed LightRAG's 46.08, similar aggregates that need not reflect similar request-level choices (Appendix E.4). Neither improves that aggregate, supporting a coarse distinction between procedural and question-answering memory.

Held-Out Groups Test More Than Unseen Memory. The best policy in the primary comparison drops from 44.00 to 36.71 under grouped hold-out. This mixed protocol does not isolate a universal failure to transfer between memories. The 7.29-point decrease comprises 0.57 from LoCoMo, 0.27 from accurate retrieval, 5.58 from test-time learning and 0.87 from ALFWorld; long-range understanding and conflict resolution contribute zero. The largest component holds out the entire test-time-learning source, whose score falls from 95.00 to 28.00. Table B.18 gives the decomposition.

ALFWorld's two memory states share descriptions, so holding out one state does not test unseen-task-text transfer. The separate leave-one-memory-unit-out comparison over the twenty LoCoMo and long-range units scores 38.47 for π_4 against 39.22 for the fixed comparator: -0.75 with $[-3.7, +1.8]$. It demonstrates no gain on unseen question-answering units (Table B.19).

Both Ranking and Verification Remain Incomplete. To understand what remains, we distinguish choice agreement from answer quality. Different choices can receive the same recorded score, including all-zero ties where selection offers no scoring benefit. No-memory allocations also include tie-broken assignments, so they do not measure strict preference for answering without memory. Moreover, removing that option changes the complete candidate set; it does not isolate each no-memory decision's causal contribution. Conversely, configurations with the same rounded aggregate can make different per-request choices. Table B.14 examines choice agreement and the decisive subset; Table B.13 gives the no-memory comparison.

The oracle within π_4's top two predictions scores 52.74: perfect verification could recover 8.74 points but would still leave 12.00 to the full oracle at 64.74. The tested verifiers, however, do not

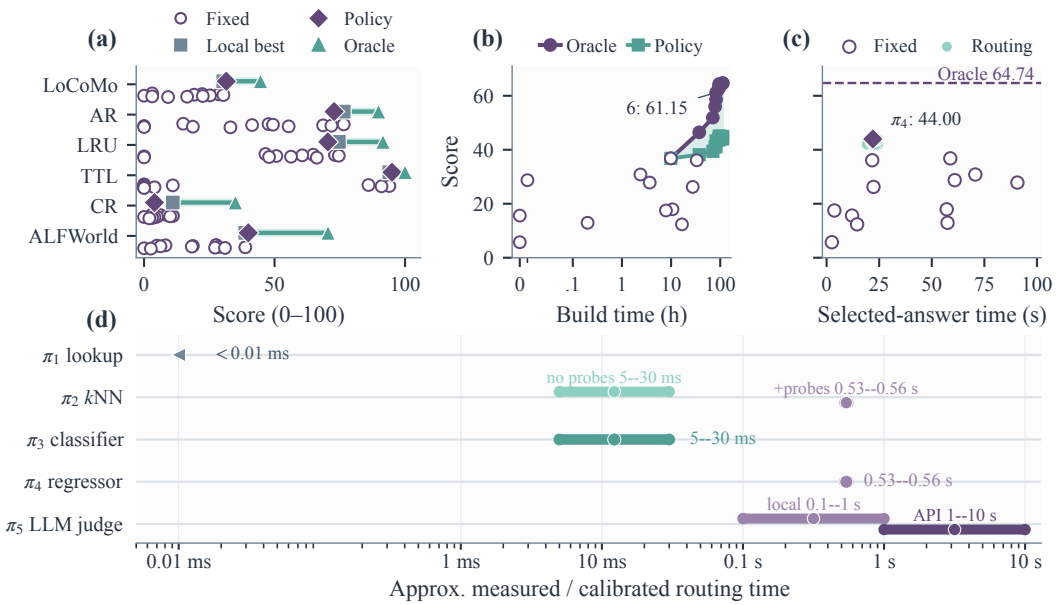


Figure 3: Routing outcomes and costs. (a) Scores by task source, where AR, LRU, TTL and CR are the MemoryAgentBench competencies, i.e., accurate retrieval, long-range understanding, test-time learning and conflict resolution. (b) Substrate construction time. (c) Selected-answer latency. (d) Representative routing time in a warm setup; π_2 separates variants with and without probes, and π_5 separates local and hosted serving.

establish a positive gain (Table B.9). Combining retrieved content offers another possibility, in configurations that vary both evidence and context length and neither isolate a mechanism nor establish deployable oracle performance. On 881 question-answering requests, five-substrate fusion scores 50.88 against the tested policy's 45.79, with the complementary pair chosen in-sample (Table B.20).

6.3 Cost of Routing

Figure 3(b) reports substrate construction rather than router training. Maintaining all twelve substrates across 25 memory units costs 112.1 hours and 2.2 GB; an in-sample subset of six reaches an oracle score of 61.15 at 82.4 hours, while its best policy scores 43.29. Because subset selection and evaluation share the pool, the curve is descriptive.

Panels (c) and (d) separate selected-answer latency from routing-decision time. Under a warm, loaded setup, lookup is below 0.01 ms; embedding-only variants of π_2 and π_3 take approximately 5–30 ms; probe-augmented π_2 and the main π_4 model take approximately 0.53–0.56 s; and π_5 takes approximately 0.1–1 s locally or 1–10 s through a hosted API. These approximately measured and benchmark-calibrated ranges exclude queueing, network transit and cold starts; they are not a matched end-to-end benchmark.

7 Conclusion

We study the value and difficulty of memory routing across 2,614 requests and 12 substrates. Semantically close requests can favor different substrates, making sentence-embedding similarity alone an incomplete routing signal. Under the reported convention, the recorded oracle scores 64.74 against 36.82 for the best fixed substrate, with much of this gap remaining within memory units. Practical routing already improves on a fixed mechanism, and a simple task-source lookup captures most of the best policy's gain. These results support memory selection and coarse task adaptation as useful starting points, while finer selection and transfer to unfamiliar memories and tasks remain open.

AI USE STATEMENT

Generative AI assistants were used to polish the language and presentation of the manuscript and to support code development and organization, data processing, LaTeX editing, and figure preparation. All AI-assisted text, code, analyses, and visualizations were reviewed by the authors, who take responsibility for the final manuscript and reported results.

ETHICS STATEMENT

This study evaluates LLM-agent memory mechanisms using publicly released research benchmarks, including LoCoMo, MemoryAgentBench and ALFWorld. It involves no human participants, no new data collection, and no private or personally identifiable user data. The experiments are limited to controlled research evaluation and do not make decisions affecting individuals. We use the datasets and software in accordance with their stated terms.

REPRODUCIBILITY STATEMENT

The request pool and data sources are described in Section 3 and Appendix A, which also specifies substrate configurations, applicability rules, model settings and metrics. Sections 5.1 and 6.1 define scoring, routing policies and evaluation protocols. Appendix C gives split construction and offline scoring, and Appendix E provides policy implementation and cost details. Together, these sections document the data, settings and procedures used for the reported results. We will release the outcome matrix with its eligibility and observation masks, the evaluation weights, the fold and group definitions, and the code that scores them and implements the policy families of Section 6.1.

REFERENCES

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.11511.

Jiayi Chen, Yingcong Li, and Guiling Wang. MemFlow: Intent-Driven Memory Orchestration for Small Language Model Agents. *arXiv preprint arXiv:2605.03312*, 2026.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance. In *Transactions on Machine Learning Research*, 2024. arXiv:2305.05176.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building Production-Ready AI Agents with Scalable Long-Term Memory. *arXiv preprint arXiv:2504.19413*, 2025.

Zhiqing Cui, Haotong Xie, Jiahao Yuan, Cheng Yang, Hanqing Wang, Yuxin Wu, Yifan Wu, Siru Zhong, Tao Yu, Yifu Guo, Siyu Zhang, Xinlei Yu, Qibing Ren, and Usman Naseem. Uno-Orchestra: Parsimonious Agent Routing via Selective Delegation. *arXiv preprint arXiv:2605.05007*, 2026.

Youwang Deng. Substrate Asymmetry in User-Side Memory: A Diagnostic Framework. *arXiv preprint arXiv:2606.11712*, 2026.

Tao Feng, Haozhen Zhang, Zijie Lei, Peixuan Han, and Jiaxuan You. GraphPlanner: Graph Memory-Augmented Agentic Routing for Multi-Agent LLMs. *arXiv preprint arXiv:2604.23626*, 2026.

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*, 2025a.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. LightRAG: Simple and Fast Retrieval-Augmented Generation. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 10746–10761, 2025b. arXiv:2410.05779.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. HippoRAG: Neurobiologically Inspired Long-Term Memory for Large Language Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2405.14831.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations (ICLR)*, 2022. arXiv:2106.09685.

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.

Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating Memory in LLM Agents via Incremental Multi-Turn Interactions. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=DT7JyQC3MR. arXiv:2507.05257.

Wei-Chieh Huang, Weizhi Zhang, Yuchen Wu, Yankai Chen, Eric Hanchen Jiang, Wooseong Yang, Yiwei Yang, Henry Peng Zou, Hanrong Zhang, Ying Nian Wu, Haolun Wu, Kai-Wei Chang, Philip S. Yu, Xue Liu, and Aylin Caliskan. Harness the Memory: A Holistic Evaluation of Memory Substrates in Memory Agents. *arXiv preprint arXiv:2608.15008*, 2026.

Zhongzhan Huang, Guoming Ling, Yupei Lin, Yandong Chen, Shanshan Zhong, Hefeng Wu, and Liang Lin. RouterEval: A Comprehensive Benchmark for Routing LLMs to Explore Model-level Scaling Up in LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 3860–3887, 2025.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C. Park. Adaptive-RAG: Learning to Adapt Retrieval-Augmented Large Language Models through Question Complexity. In *Proceedings of NAACL*, 2024. arXiv:2403.14403.

Hao Li, Yiqun Zhang, Zhaoyan Guo, Chenxu Wang, Shengji Tang, Qiaosheng Zhang, Yang Chen, Biqing Qi, Peng Ye, Lei Bai, Zhen Wang, and Shuyue Hu. LLMRouterBench: A Massive Benchmark and Unified Framework for LLM Routing. *arXiv preprint arXiv:2601.07206*, 2026.

Jun Liu, Zhenglun Kong, Changdi Yang, Fan Yang, Tianqi Li, Peiyan Dong, Joannah Nanjekye, Hao Tang, Geng Yuan, Wei Niu, Wenbin Zhang, Pu Zhao, Xue Lin, Dong Huang, and Yanzhi Wang. RCR-Router: Efficient Role-Aware Context Routing for Multi-Agent LLM Systems with Structured Memory. *arXiv preprint arXiv:2508.04903*, 2025.

Yifan Lu, Rixin Liu, Jiayi Yuan, Xingqi Cui, Shenrun Zhang, Hongyi Liu, and Jiarong Xing. RouterArena: An Open Platform for Comprehensive Comparison of LLM Routers. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2510.00202.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating Very Long-Term Conversational Memory of LLM Agents. In *Proceedings of ACL*, 2024. arXiv:2402.17753.

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. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2406.18665.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. ReasoningBank: Scaling Agent Self-Evolving with Reasoning Memory. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2509.25140.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as Operating Systems. *arXiv preprint arXiv:2310.08560*, 2023.

Cheng Qian, Zuxin Liu, Shirley Kokane, Akshara Prabhakar, Jielin Qiu, Haolin Chen, Zhiwei Liu, Heng Ji, Weiran Yao, Shelby Heinecke, Silvio Savarese, Caiming Xiong, and Huan Wang. xRouter: Training Cost-Aware LLMs Orchestration System via Reinforcement Learning. *arXiv preprint arXiv:2510.08439*, 2025.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2401.18059.

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 (ICLR)*, 2021. arXiv:2010.03768.

Joshua Tobkin and David Yang. Supra Cognitive Modes: A Routed Architecture for Agent Memory. *arXiv preprint arXiv:2607.19096*, 2026.

Co Tran, Salman Paracha, Adil Hafeez, and Shuguang Chen. Arch-Router: Aligning LLM Routing with Human Preferences. *arXiv preprint arXiv:2506.16655*, 2025.

Chen Wang, Xunzhuo Liu, Yuhan Liu, Yue Zhu, Xiangxi Mo, Junchen Jiang, and Huamin Chen. When to Reason: Semantic Router for vLLM. *arXiv preprint arXiv:2510.08731*, 2025.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking Chat Assistants on Long-Term Interactive Memory. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2410.10813.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-Mem: Agentic Memory for LLM Agents. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2502.12110.

Yanwei Yue, Guibin Zhang, Boyang Liu, Guancheng Wan, Kun Wang, Dawei Cheng, and Yiyan Qi. MasRouter: Learning to Route LLMs for Multi-Agent Systems. *arXiv preprint arXiv:2502.11133*, 2025.

Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. MemSkill: Learning and Evolving Memory Skills for Self-Evolving Agents. *arXiv preprint arXiv:2602.02474*, 2026.

A SETUP DETAILS

Substrates. Table A.1 lists the 12 substrates, their families, read mechanisms and budgets. We follow the families in Huang et al. (2026). We hold the reference implementation's default configurations fixed in the main matrix, without tuning them ourselves, and leave comprehensive tuning and auxiliary-component sensitivity outside scope. **This choice compares memory mechanisms rather than the outcomes of separate hyperparameter searches.**

Table A.1: The 12 substrates, their type as in Table 2 in Section 4, how they are read, and the default read budget.

	Substrate	Type	Read mode	Top-k
M_0	No memory	none	empty	–
M_1	Dense retrieval	flat retrieval	dense top-k	10
M_2	Sparse retrieval	flat retrieval	BM25 top-k	5
M_3	Summarized text	summary index	gist page selection	5
M_4	A-Mem	evolving notes	agentic search	5
M_5	LightRAG	structured	hybrid graph and vector	10
M_6	Summary tree	structured	hierarchical dense top-k	5
M_7	ReasoningBank	procedural	dense strategy top-k	1
M_8	Skill bundles	procedural	dense memory top-k	10
M_9	LoRA	parametric	adapter generation	–
M_{10}	Full context	context-native	full context	–
M_{11}	Compressed KV	context-native	selected episode re-prefill	–

Substrates Without A Single Upstream System. Three substrates require their construction to be stated, since no one published system defines them. M_3 is a gist index built with the auxiliary model, i.e., the history is buffered into 500-word pages, each page is summarized once, and a request selects among at most 200 gists before the selected pages are copied into the prompt. M_8 is a reduced variant of MemSkill, i.e., its released conversational skill bank is used verbatim and prefilled for every fixed-length chunk, whereas the controller, the state encoder and the skill-evolution loop of the original remain inactive, since they operate only during reinforcement learning. M_{11} re-prefills selected episodes rather than compressing attention state, i.e., utterances are buffered, clustered into episodes, and the selected episodes are re-prefilled through the backbone's own attention path, so it is not an implementation of a published key-value compression method. M_8 **and** M_{11} **are accordingly built for this evaluation rather than reproduced**, so a result on either bounds what our configuration achieves rather than what the original design achieves.

Component Configuration. We decode *Qwen3.5-9B* greedily with at most 512 new tokens and an effective context of 32K. Substrates in the main matrix use *gpt-4o-mini-2024-07-18* for any LLM calls needed to build or maintain memory, and *text-embedding-3-small* for embeddings. The routing policies in Section 6 use *all-MiniLM-L6-v2* as their sentence encoder.

Metrics. We use task success for ALFWorld, token-level F1 for LoCoMo, and competency-specific scores for MemoryAgentBench. Conflict resolution uses binary substring match of a gold answer, and test-time learning uses exact match of its numeric label.

For accurate retrieval, we adapt substring matching to allow partial credit through *answer-item coverage*. Each request has $m_i \geq 1$ required answer items fixed before scoring, and each item k has acceptable aliases A_{ik}. We score the single generated answer y_i as

$$q_i = \frac{1}{m_i} \sum_{k=1}^{m_i} \mathbf{1}[\exists a \in A_{ik} : N(a) \text{ is a substring of } N(y_i)] \,,$$

where N lowercases text, removes punctuation and English articles, and collapses whitespace. Aliases are alternatives for one item, while distinct required items receive separate credit. A single answer can therefore score $1/2$ or $2/3$ without repeated execution.

We average the 100 request-level scores and multiply by 100 to obtain the source score. This ordinary mean neither counts fully correct requests nor weights requests by their number of items. The

adaptation differs from the benchmark's binary substring-exact-match metric: it permits fractional source means, but integer aggregate totals remain possible. For this competency, success-status analyses throughout the paper, require full coverage, $q_i = 1$; positive-score coverage uses $q_i > 0$.

The native exact-match metric for long-range understanding gives every substrate zero on its structured answers. We replace it with substring match, explicitly changing the native metric. The weights w_i in Eq. (1) in Section 5.1 assign equal weight to the three benchmarks and, within MemoryAgentBench, to the four competencies.

Evaluation-Profile Eligibility and Run Status. The evaluation profile's restrictions specify the scope of the comparison, rather than establishing a substrate's intrinsic ability to serve a request. The eligibility mask a_{ij} defines the substrate–request pairs admitted by the evaluation profile.

The profile restricts ReasoningBank to ALFWorld, excluding 2,346 cells. Full context requires the history and a generation reserve to fit the 32K window under the backbone tokenizer; the reference profile also excludes it on ALFWorld. Together, these rules exclude 1,550 cells. Compressed KV and A-Mem are excluded on test-time learning, removing 100 cells each. For A-Mem, this is a budget-related restriction on processing the 5,900-example history, not a report of separately observed timeouts or failed answers on those 100 cells. The total is $2{,}346 + 1{,}550 + 100 + 100 = 4{,}096$ profile-excluded cells with $a_{ij} = 0$.

Policies receive the eligibility mask and cannot select excluded cells, on which fixed-substrate references receive zero credit under Section 5.1. In two ALFWorld online-accumulation runs, 70 eligible cells reached the suite-level wall-clock budget before producing a task score ($a_{ij} = 1, o_{ij} = 0$). The evaluation treats them as run-limit failures and assigns zero credit when selected. All comparisons condition on these profile assignments; budget-related exclusions do not establish intrinsic substrate limitations. Thus eligibility determines the candidate set, while completion status records whether an eligible execution produced a task score.

B Additional Results

B.1 Similar Requests and Substrate Outcomes

Section 4 uses sentence-embedding similarity as a motivating probe. We encode requests with normalized *all-MiniLM-L6-v2* embeddings and form all pairs above each cosine-similarity threshold. Among the 8,435 pairs at cosine similarity 0.8 or above, 49% come from different memory units. Table 1 in Section 4 reports the accompanying outcome statistics for five representative memory frameworks.

We mark a pair as *inconsistent* for a framework when one request receives a success score and the other receives a non-success score under the reported convention. Success means task completion for ALFWorld, the competency-specific criterion for MemoryAgentBench, and token-level F1 of at least 0.5 for LoCoMo. The rates are pooled descriptions rather than controlled estimates: they combine sources, memory states, repeated wording, and different applicability sets. In particular, ReasoningBank is applicable only to ALFWorld. The analysis does not establish that all routing-relevant information is absent from the request text. It illustrates why routing based on this sentence-embedding proxy may fail in our setting.

Table B.1 illustrates why aggregate similarity alone can be a weak guide. Two questions can be almost identical in form yet favor different ways of organizing the same history. These examples do not imply that their relevant distinction is absent from the text; they show that the cosine proxy does not by itself determine a reliable substrate choice.

Table B.1: Highly similar requests that receive success scores from different memory frameworks.

Request	Type	Dense	A-Mem	LightRAG	Tree
What did Gina find for her clothing store on 1 February, 2023?	single-hop	×	×	✓	×
When did Gina open her online clothing store?	temporal	×	×	×	✓
Who was Margaret's husband?	retrieval	×	✓	×	✓
Who was Margaret's brother?	retrieval	×	×	✓	✓

B.2 Reference Points of the Headroom

We detail the quantities reported in Section 5.2 using the same outcome record and scoring matrix $\widetilde{\mathbf{Q}}$, with all scores on the 0–100 scale of Eq. (1) in Section 5.1.

Routing Strategies. Table B.2 orders the reference strategies by the information they use. *Without routing* equips an agent with a single substrate and uses no request information. *Routing by benchmark* selects the best substrate separately for each of the three benchmarks from the full evaluation matrix, making it an in-sample reference. The cross-validated lookup policy instead uses six task-source keys and selects their substrates on training folds. Task-source metadata is also supplied to π_2 and π_4, while the applicability mask reveals ALFWorld to every policy.

Perfect routing uses each request's observed applicable scores. It upper-bounds policies under the stated zero-credit convention and is not implementable. The headroom H is the gap between the single-substrate strategy and this oracle.

Table B.2: Reference points of the headroom, ordered by how much information each strategy uses. *Information* is what the strategy knows about a request at routing time, and *Score* is reported on the 0–100 scale.

Strategy	Information	Substrate for a request	Score
Without routing	none	LightRAG (M_5) for all requests	36.82
Routing by benchmark	benchmark identity	the best substrate of that benchmark	43.66
Perfect routing	observed applicable scores	$\arg\max_{M_j \in \mathcal{O}_i} Q_{ij}$	64.74

Breakdown By Benchmark. Tables B.3 and B.4 decompose the same scores by benchmark, with *Average* giving the equal-weight average over the three benchmarks used in the main text. These values also recover the reference points in Table B.2: *routing by benchmark* takes the best entry in each column of Table B.3, yielding an average of 43.66. Relative to that benchmark-specific fixed choice, the remaining oracle gaps are 14.25 on LoCoMo, 17.29 on MemoryAgentBench and 31.71 on ALFWorld. Coarse benchmark adaptation therefore removes the largest cross-benchmark mismatch but leaves substantial variation within each benchmark, especially ALFWorld.

Table B.3: Breakdown by benchmark of the scores *without routing*. Each entry is the score when substrate M_j is applied to all requests, i.e., $S(M_j)$. "–" denotes an inapplicable substrate.

	Substrate	LoCoMo	MAB	ALFWorld	Average
M_0	No memory	3.11	16.12	27.61	15.62
M_1	Dense retrieval	29.28	55.00	8.21	30.83
M_2	Sparse retrieval	22.30	59.07	4.85	28.74
M_3	Summarized text	28.71	36.16	18.66	27.84
M_4	A-Mem	19.23	28.25	6.34	17.94
M_5	LightRAG	**30.36**	**61.81**	18.28	**36.82**
M_6	Summary tree	25.33	54.63	28.36	36.11
M_7	ReasoningBank	–	–	**38.81**	12.94
M_8	Skill bundles	22.54	28.94	27.24	26.24
M_9	LoRA	3.13	18.39	30.97	17.50
M_{10}	Full context	16.47	0.75	–	5.74
M_{11}	Compressed KV	9.15	25.32	2.61	12.36
Global best fixed: LightRAG (S_{fix})		30.36	61.81	18.28	**36.82**
Best fixed per benchmark (routing by benchmark)		30.36	61.81	38.81	**43.66**

Table B.4: Oracle score contributions by substrate and benchmark. Ties are split equally; each benchmark column sums to its oracle score, and Average uses equal benchmark weights.

	Substrate	LoCoMo	MAB	ALFWorld	Average
M_0	No memory	0.15	2.38	7.10	3.21
M_1	Dense retrieval	5.58	12.25	2.04	6.62
M_2	Sparse retrieval	4.25	12.92	1.12	6.10
M_3	Summarized text	**7.76**	6.69	6.12	6.86
M_4	A-Mem	3.30	5.65	1.62	3.52
M_5	LightRAG	6.52	**14.16**	5.47	8.71
M_6	Summary tree	4.46	12.62	10.98	**9.35**
M_7	ReasoningBank	–	–	**15.11**	5.04
M_8	Skill bundles	6.18	5.01	10.25	7.15
M_9	LoRA	0.36	2.76	10.20	4.44
M_{10}	Full context	4.52	0.22	–	1.58
M_{11}	Compressed KV	1.52	4.43	0.51	2.15
Total (S_{orc})		44.61	79.10	70.52	**64.74**

B.3 Where Each Task Source Stands

Figure 3(a) compares the three reference levels of Section 5 within each task source. Headroom varies substantially: it is widest on ALFWorld (31.7) and conflict resolution (24.0), moderate on long-range understanding (16.9) and LoCoMo (14.2), narrower on accurate retrieval (13.3), and almost closed on test-time learning (6.0), where one substrate already answers nearly everything.

The best policy exceeds the best fixed substrate within three of the six sources, never by more than 1.2 points. The differences are $+1.15$ on LoCoMo, $+1.12$ on ALFWorld and $+1.00$ on test-time learning, compared with -3.67 on accurate retrieval, -4.23 on long-range understanding and -7.00 on conflict resolution. Yet the aggregate gain over S_{fix} comes almost entirely from ALFWorld, where the globally best substrate, LightRAG, is far below the locally best one. A gain from coarse source distinctions can therefore coexist with substantial per-request error.

B.4 How Precise the Headroom Is

We assess uncertainty in Table B.6 by resampling single requests, the 29 cross-validation groups, and the 25 memory units, with 2,000 replicates at each level. The interval widens as the cluster grows: resampling 25 units includes two ALFWorld memory states that jointly carry a third of the weight in Eq. (1) in Section 5.1. Nevertheless, **the lower bound of relative headroom remains 27.9% at the most conservative level**, showing a positive pooled recorded-matrix gap across these resampling levels. These intervals hold the cell outcomes fixed and do not estimate or remove the bias from selecting execution-specific maxima.

Sensitivity To The Weighting. Eq. (1) in Section 5.1 gives the three benchmarks equal weight, preventing LoCoMo's 76% share of the requests from determining the result alone. We also compute two alternative weightings: equal weight per request and equal weight per task source across the six sources. Table B.5 reports the headline quantities for all three.

The weighting substantially changes the magnitudes: relative headroom ranges from 45.8% to 75.9%, and the share captured by the best policy ranges from 13.1% to 25.7%. **The headline figures are the most favorable of the three and should be read as one point in that range**. Across these changes in magnitude, the ordering remains unchanged: under every weighting, headroom is large and the best policy captures a minority of it.

Table B.5: The headline quantities under the three weightings we report. *Suite-balanced* is the one used throughout the paper.

Weighting	S_{fix}	S_{orc}	H	Relative	π_4 captures
Suite-balanced, as reported	36.82	64.74	27.93	+75.9%	25.7%
Every request equally	33.44	52.02	18.58	+55.6%	15.8%
Every task source equally	49.31	71.92	22.61	+45.8%	13.1%

Table B.6: Cluster bootstrap of the headline quantities at three levels of resampling, with 2,000 replicates and 95% percentile intervals.

Resampled unit	Clusters	S_{fix}	S_{orc}	H (relative)
Point estimate	—	36.82	64.74	27.93 (75.9%)
A request	2,614	[34.72, 39.57]	[62.33, 67.06]	[25.27, 30.11] ([65.2%, 85.2%])
A cross-validation group	29	[26.39, 51.01]	[54.68, 74.42]	[19.67, 34.10] ([40.4%, 121.3%])
A memory unit	25	[26.24, 54.41]	[52.86, 74.97]	[14.15, 34.57] ([27.9%, 122.8%])

B.5 The Headroom Inside One Memory

A deployed agent usually serves one memory at a time, such as a user's conversation or a document corpus, for which an engineer can tune the substrate. To examine the remaining headroom, Table B.7 fixes each of the 25 memory units, selects its best substrate, and measures the value of per-request routing within that unit.

Under the weights of Eq. (1) in Section 5.1, the best substrate per unit scores 45.88 and the per-request oracle scores 64.74, leaving 18.87 points of headroom within a single memory, or a relative gain of 41.12%. The recorded gap therefore exists within individual memory units, without establishing how much would transfer to an independent execution. The six units with no internal headroom contain 6 to 8 requests, whereas every unit with at least 100 requests retains more than 5 points.

B.6 Repeated Request Text and Fold Construction

ALFWorld contains 134 task instances, represented by 78 distinct task-description strings, in each of two memory states: an offline bank of expert trajectories and the memory accumulated online. The resulting 268 requests can repeat a description across task instances and memory states. Permuting requests within each memory unit can therefore put identical texts in different folds: under that split, 256 of the 268 embodied requests, or 95.5%, have a verbatim twin in their training set. A policy using request text can exploit this repetition without generalizing to a new description.

Table B.8 compares an earlier within-unit request split with the text-grouped split used in the main evaluation. The latter keeps identical texts together and refits every policy on its training folds. The lookup scores change by $+0.17$ for the source key and -0.06 for source + type; the latter also changes from 39.18 to 38.06 on ALFWorld.

Every policy using text scores lower under the text-grouped split, with the largest change for the regressor using only the query-embedding feature block. The best policy changes from 45.96 to 44.00, reducing its advantage over the lookup table from $+2.73$ to $+0.60$. The 44.00 text-grouped result is the main score used throughout the paper; the earlier within-unit split is retained here

Table B.7: Headroom inside each memory unit, ordered by H. *Best* is the substrate with the highest score on that unit alone, and request scores are unweighted within the unit. Accurate-retrieval entries average within-request answer-item coverage (Appendix A).

Source	Requests	Best	Best alone	Perfect	H
ALFWorld	134	M_7	40.30	81.34	41.04
Long-range	6	M_1	66.67	100.00	33.33
Conflict res.	100	M_3	11.00	35.00	24.00
ALFWorld	134	M_7	37.31	59.70	22.39
Long-range	9	M_4	66.67	88.89	22.22
LoCoMo	199	M_{10}	32.13	46.81	14.68
LoCoMo	105	M_{10}	33.54	48.14	14.60
LoCoMo	228	M_5	31.34	44.96	13.62
LoCoMo	190	M_3	30.51	43.85	13.34
Accurate retr.	100	M_2	76.50	89.83	13.33
Long-range	8	M_5	87.50	100.00	12.50
LoCoMo	193	M_5	36.26	48.44	12.18
LoCoMo	158	M_5	31.48	43.48	12.00
LoCoMo	242	M_5	29.33	41.22	11.89
LoCoMo	204	M_{10}	33.27	44.47	11.20
Long-range	10	M_3	90.00	100.00	10.00
LoCoMo	196	M_{10}	36.37	45.56	9.19
LoCoMo	260	M_{10}	33.43	42.15	8.71
Test-time learn.	100	M_5	94.00	100.00	6.00
Long-range	6	M_1	100.00	100.00	0.00
Long-range	6	M_2	83.33	83.33	0.00
Long-range	6	M_1	83.33	83.33	0.00
Long-range	6	M_3	50.00	50.00	0.00
Long-range	6	M_3	100.00	100.00	0.00
Long-range	8	M_2	100.00	100.00	0.00

only to show the effect of fold construction. The grouped hold-out protocol separately keeps both ALFWorld memory-state copies of a task together within its task-family cluster. The 256 figure counts requests with a training-set twin, not distinct duplicated strings in the full pool.

Keeping repeated request text within one fold lowers the apparent routing gain by about two points in this comparison, showing that fold construction matters when benchmark requests share wording.

Table B.8: Known-memory evaluation under two fold constructions: an earlier split that distributes requests across folds within each memory unit, and the text-grouped split used for the main results. *ALFWorld* is the embodied-source score.

	Overall			ALFWorld	
Policy	within unit	text grouped	Δ	within unit	text grouped
π_1, source	43.23	43.40	+0.17	38.81	38.81
π_1, source + type	43.18	43.12	−0.06	39.18	38.06
π_2, kNN (e, u, p)	44.00	43.26	−0.74	38.43	36.19
π_4, $\phi = e$	45.00	41.90	−3.10	46.64	35.45
π_4, $\phi = (e, u)$	45.34	43.30	−2.04	45.52	38.81
π_4, $\phi = (e, u, p)$	45.96	44.00	−1.96	46.27	39.93

B.7 Verifying the Answer Instead of Ranking the Substrate

On the recorded matrix, the oracle over the top two predictions of π_4 scores 52.74 against 44.00 for its first choice, leaving 12.00 points to the full oracle. These score differences avoid treating an all-zero maximum as a successful ranking hit. Answer verification can potentially recover part of the gap, while improved ranking can offer further gains.

To test verification, we present a language model with the question and two candidate answers, randomizing their order per request to prevent position-based selection while withholding both the gold answer and the score. Only the 1,987 requests with differing answers are sent to the verifier.

In Table B.9, *gpt-5.4-mini* gains 2.25 points, the largest point improvement among the three configurations in Table B.9. Its interval includes zero, so this positive estimate remains directional evidence rather than an established gain. It switches on 36.4% of pairs, selecting 278 of the 480 better second answers at the cost of selecting 241 of the 604 worse ones.

The other two verifiers do not improve the score: *deepseek-v3.2* switches at a similar rate but with a worse hit rate, losing 1.48 points. Under the original output budget, *qwen3.5-plus* retains the first candidate almost every time and thus leaves π_4's choice largely unchanged.

These three comparisons do not establish a positive verifier gain, nor do they show that every inexpensive verifier fails. Output-budget effects and candidate ranking remain separate questions for further evaluation.

Table B.9: Choosing between the two answers that π_4 ranks first and second. *Switched* is the share of differing pairs on which the verifier prefers the second substrate.

	Score	vs. π_4	Switched	Caught	Wrongly switched
π_4, top choice only	44.00	—	—	—	—
deepseek-v3.2	42.52	−1.48 [−3.3, +0.1]	38.0%	212/480	313/604
gpt-5.4-mini	46.24	+2.25 [−0.8, +4.9]	36.4%	278/480	241/604
qwen3.5-plus	43.99	−0.00 [−0.0, +0.0]	0.1%	0/480	2/604
An oracle over the same two	52.74	+8.74	—	480/480	0/604

B.8 What the LLM Judges Select

Table B.10 reports each prompted model's score and most frequently selected substrate. Several judges concentrate on one substrate: *gpt-4o-mini* selects dense retrieval on 93% of requests, and *Llama 3.3 70B* selects LightRAG on 87%. Near-exclusive selection is not universal, however: full context is the modal choice for three judges at 39–41%, close to its applicability rate of 40.7%,

and their aggregate scores also reflect choices on the remaining requests. The selection shares are displayed as whole percentages: 41% is compatible with the 40.7% eligibility rate at that precision. A modal choice therefore does not imply the score of always using that substrate. None of the judges produces an oracle-like selection distribution, which spreads across all 12 substrates (Section 5).

We exclude *DeepSeek V4 Pro* and *MiniMax M3*, both of which return a chain of thought in a separate field before emitting an answer. Under the allowed generation budget, 90% of their responses contain no answer. The parser then falls back to the first feasible substrate, which would give scores near 16 and make truncated responses appear to be deliberate abstention. Reporting the exclusion distinguishes this generation-budget failure from a routing decision.

Table B.10: Every model prompted as π_5, ordered by score. *Most frequent* is the substrate it selects most often and its request share, rounded to a whole percentage.

Vendor	Model	Score	vs. π_1	Most frequent
Meta	*Llama 3.3 70B*	43.68	+0.28	LightRAG, 87%
—	the backbone, *Qwen3.5-9B*	40.31	−3.09	—
Moonshot	*Kimi K3*	39.78	−3.62	full context, 39%
Anthropic	*Claude Opus 4.5*	38.97	−4.42	full context, 39%
OpenAI	*gpt-4o-mini*	35.85	−7.55	dense retrieval, 93%
Anthropic	*Claude Opus 5*	34.91	−8.48	full context, 41%
OpenAI	*GPT-5.6-Terra*	34.84	−8.55	sparse retrieval, 66%
OpenAI	*gpt-5.2*	31.29	−12.11	sparse retrieval, 75%

B.9 Cost of the Substrates

Table B.11 pairs each substrate's score with its costs. *Latency* measures selected-answer wall-clock time under the weights of Eq. (1) in Section 5.1, excluding routing probes and decision calls. *Build* totals construction and maintenance time over the 25 memory units, including auxiliary-model calls and local operations such as adapter training and KV processing. *Storage* gives the resulting disk size, with both build time and storage shared by requests on the same memory unit. Because the time totals combine different operations, they represent neither API billing hours nor hardware-normalized compute costs.

Latency averages also differ in coverage: full context is eligible on 1,064 of 2,614 requests and excludes ALFWorld, while no memory and LoRA cover the full pool. Its 2.5-second mean therefore describes a different request set. Coverage does not explain the 3.7-second LoRA mean versus 12.1 seconds without memory. These aggregates cannot isolate adapter overhead or prefill cost without matched input and output lengths, hardware, cache state, batching, concurrency and timer boundaries. We therefore use them descriptively, without treating them as an intrinsic speed ranking of the mechanisms.

The table is a record of selected-answer times under the evaluated configurations. It does not identify which differences arise from the memory mechanism, input and output lengths, or serving conditions.

Maintaining all 12 substrates costs 112.1 construction hours and 2.2 GB. Selected-answer latency conditions on this maintained candidate set and excludes routing probes and decision calls, so it does not establish an end-to-end speedup.

B.10 Per-Request Scores

Requests with no positive recorded outcome under any candidate account for 25.43% of the evaluation weight (Table 3 in Section 5.2). Figure B.1 instead counts completed cells of $\mathbf{Q}$. The distribution is concentrated near its endpoints: 60.4% fall in the lowest score bin and 9.0% in the highest. These cell counts differ from the weighted request summary, and intermediate scores remain possible, including LoCoMo's token-level F1 scores.

Table B.11: Score and recorded cost of each substrate. *Latency* measures the selected answer, excluding routing probes and decision calls. These descriptive means are not hardware-normalized or matched-workload speed measurements. It is averaged over the requests the substrate can serve, and for perfect routing over all requests with ties among equally best substrates split evenly. It is reported per request, whereas *Build* and *Storage* are totals over the 25 memory units.

	Substrate	Score	Latency (s)	Build (h)	Storage (MB)
M_0	No memory	15.62	12.1	0.00	0
M_1	Dense retrieval	30.83	70.7	2.38	97
M_2	Sparse retrieval	28.74	60.9	0.01	26
M_3	Summarized text	27.84	90.7	3.67	40
M_4	A-Mem	17.94	57.1	10.67	12
M_5	LightRAG	36.82	58.8	9.86	1378
M_6	Summary tree	36.11	21.7	33.29	243
M_7	ReasoningBank	12.94	57.5	0.20	4
M_8	Skill bundles	26.24	22.3	27.54	296
M_9	LoRA	17.50	3.7	7.85	125
M_{10}	Full context	5.74	2.5	0.00	1
M_{11}	Compressed KV	12.36	14.5	16.65	16
Without routing (M_5)		36.82	58.8	9.86	1,378
With perfect routing		**64.74**	**23.6**	112.1	2,238

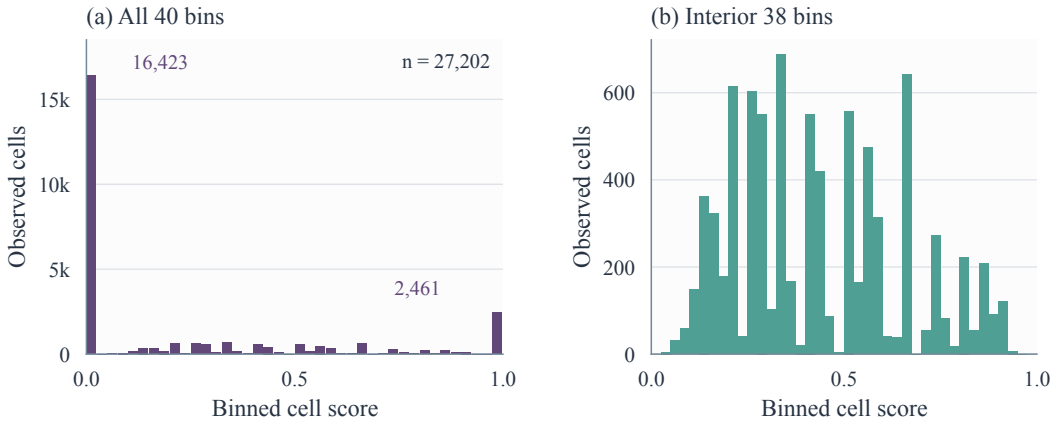


Figure B.1: Distribution of 27,202 completed cell scores over 40 equal-width bins (left), with the 38 interior bins enlarged (right). The endpoint bars aggregate the lowest and highest score intervals.

B.11 Generalization to Unseen Sources

We hold out one source at a time in Table B.12 and apply π_4, $\phi = (e, u)$, trained on the remaining sources. *Routed* gives its score, *Best fixed* gives the best single substrate on that source, and *Perfect routing* gives its oracle.

The router falls below the best fixed substrate selected retrospectively for each target source, whose choice uses held-out source scores. This measures a shortfall from a target-informed reference, not a disadvantage against a deployment baseline selected without target outcomes. Comparison with a deployable no-routing policy would require choosing its fixed substrate from the training sources alone. The aggregate scores do not identify why the router fails on individual requests. Test-time learning has the largest gap in this comparison, where the entire competency is absent from training.

Table B.12: Leave-one-source-out generalization of π_4, $\phi = (e, u)$. Each row holds out one source and routes its requests with the router trained on the remaining ones, where *Best fixed* and *Perfect routing* are the best single substrate on the held-out source and its oracle.

Held-out source	Requests	Routed	Best fixed	Perfect routing
LoCoMo	1,975	27.44	30.36	44.61
Accurate retrieval	100	68.25	76.50	89.83
Test-time learning	100	18.00	94.00	100.00
Long-range understanding	71	67.61	74.65	91.55
Conflict resolution	100	10.00	11.00	35.00
ALFWorld	268	18.28	38.81	70.52

B.12 Routing to No Memory

Table B.13 compares configurations with and without the no-memory option M_0, describing allocation and aggregate performance without establishing a learned preference to avoid memory. *Assigned to* M_0 measures evaluation-weighted allocation, including choices resolved by a tie-breaking rule. It does not distinguish strict predicted preference, $\hat{q}(x_i, M_0) > \max_{j \neq 0 : M_j \in \mathcal{M}_i} \hat{q}(x_i, M_j)$, from a tied maximum. Likewise, agreement with a maximum-score label, $\widetilde{Q}_{i0} = \max_j \widetilde{Q}_{ij}$, can include requests on which every feasible substrate scores zero. Such agreement alone does not establish useful or correct answering without memory.

The regressor scores 44.00 with M_0 and 44.97 without it, allocating 3.67% to M_0. The 44.97 configuration lies outside the 23 full-library variants and their paired tests, so we report no interval for its difference from the task source table. The nearest-neighbor allocation is 16.76%, while its scores with and without M_0 both round to 43.26. These equal rounded scores do not establish identical request-level outcomes: changes may lie below the displayed precision or offset across requests. For kNN, an all-zero neighbor score vector can create tied predicted maxima. Although the aggregate allocation does not identify this case's contribution, a tie rule can then select M_0 without predicting an advantage.

Table B.13: Scores with and without the no-memory option (M_0). Assignment includes tie-broken choices and must not be interpreted as the share with a strict predicted preference for M_0. Scores and allocations use Eq. (1) in Section 5.1.

Policy	With M_0	Without M_0	Assigned to M_0
π_4, $\phi = (e, u, p)$	44.00	44.97	3.67%
π_2, kNN (query + memory + probes)	43.26	43.26	16.76%

B.13 Adversarial Questions and Metric Scope

LoCoMo's adversarial questions account for 446 of its 1,975 requests and score close to zero under every substrate. These questions contain unsupported premises and call for refusal, which token-level F1 against a reference answer may under-credit. Low F1 therefore does not by itself distinguish an appropriate refusal from a failed answer.

The main scores retain these requests under the stated metric. Removing them changes the evaluated task mix; preserving equal benchmark weights would require recomputing each substrate's score on the remaining LoCoMo requests. A near-invariant relative gap after global weight renormalization can follow from rescaling and would not independently establish metric robustness. We make no refusal-credit or alternative-metric headroom claim without direct regrading under an explicit rubric.

B.14 Why Different Implementations Reach the Same Score

Table B.14 distinguishes substrate selection from achieved score. The agreement column gives the share of requests assigned to the same substrate as π_1 with the source key. The last column restricts scoring to *decisive* requests, where at least one substrate succeeds and the feasible substrates are not all equivalent.

Two observations help explain the narrow score band in Table E.1. First, π_2 and π_4 choose a different substrate on roughly three quarters of requests, so similar scores do not reflect identical decisions. Second, selection cannot improve the recorded score on the 25.43% of evaluation weight with no positive observed outcome, or on requests where all feasible candidates score equally. Among decisive requests, an average of 2.96 of 10.35 feasible substrates meets the source-specific success criterion (task success for ALFWorld, full credit for MemoryAgentBench and F1 of at least 0.5 for LoCoMo), allowing different choices to meet that criterion.

These ties do not imply that policies consistently find an adequate substrate. Meeting a success threshold need not maximize a continuous score such as LoCoMo F1. On decisive requests, the oracle scores 86.55 against 57.99 for π_4, leaving a 28.56-point gap. Neither aggregate scores nor tie counts locate this gap exclusively among requests with one uniquely correct substrate: a policy can miss every adequate candidate even when several are available. Locating those errors requires a decomposition by the number of successful candidates and the chosen score.

Table B.14: What each policy selects, and what it scores on the requests where the selection matters. *Same choice as* π_1 is the share of requests on which the policy selects the substrate that the source lookup table selects.

Policy	Same choice as π_1	All requests	Decisive requests
Without routing (M_5)	–	36.82	48.11
π_1, source	100.00%	43.40	57.16
π_2, kNN (query + memory + probes)	26.36%	43.26	56.98
π_3, vLLM-SR style	98.89%	43.31	57.05
π_4, $\phi = (e, u, p)$	22.11%	44.00	57.99
Perfect routing	–	64.74	86.55

B.15 Where the Headroom Sits

Computing the best substrate within each non-adversarial request type in Figure B.2 reveals two main observations. First, headroom is concentrated: embodied task types range from 8.3 points for looking at an object in light to 40.3 for cleaning, and conflict resolution retains 24.0 points. Test-time learning has only 6 because one fixed substrate already scores 94.

Second, **seven different substrates are best for some non-adversarial request type**: M_0, M_1, M_2, M_3, M_5, M_6 and M_7. This describes heterogeneity across types in the evaluated pool, without establishing that type information improves a learned lookup rule. Under cross-validation, the source + type table scores 43.12 against 43.40 for the source-only table. The per-type oracle and the learned table thus answer different questions.

B.16 A Second Backbone

We examine the recorded substrate gap with a larger backbone using a smaller Qwen3.5-27B matrix of 82 requests under 10 substrates. Table B.15 compares these cells with their Qwen3.5-9B counterparts. Both reference scores improve: the best fixed substrate rises from 43.10 to 49.39 and perfect routing from 66.44 to 69.47. LightRAG remains best on average for both backbones, while relative headroom falls from 54.2% to 40.6% but stays positive. **On this matched subset, the**

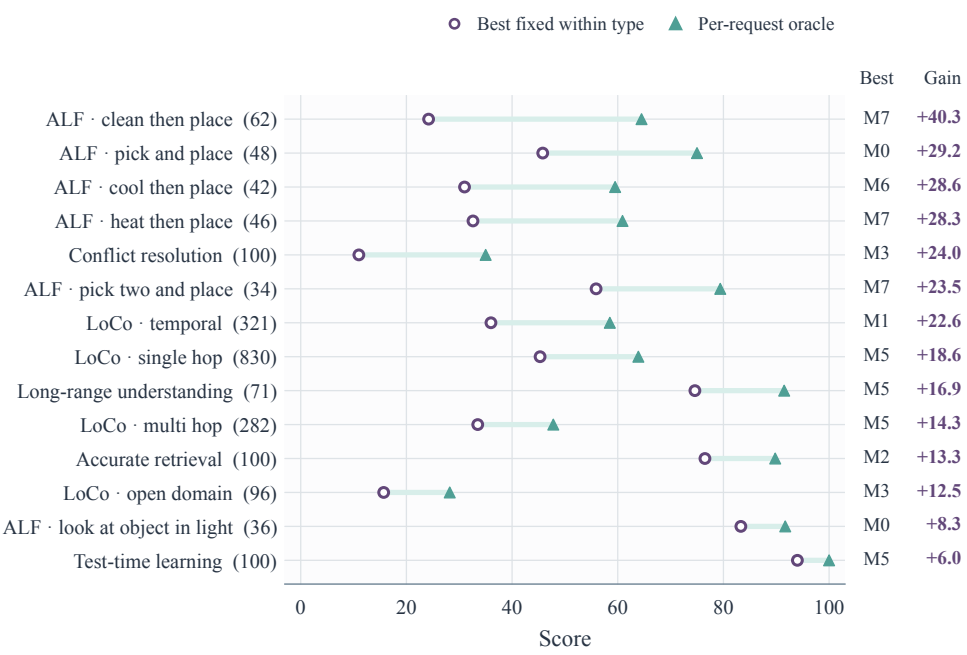


Figure B.2: Scores by request type, excluding LoCoMo adversarial requests. Circles show the best fixed substrate within each type; triangles show the per-request oracle. Counts are in parentheses, and the right columns report the selected substrate and gain.

larger backbone reduces the relative gap while retaining a gap between the fixed choice and the oracle.

The separate transfer comparison selects under the 9B matrix and scores under 27B, reaching 55.88. This outcome-score comparison concerns only the matched sample, not robustness across all sources or execution states. That score exceeds the 27B best fixed score of 49.39 while remaining below its oracle of 69.47, showing a positive transfer gain on this small matched subset.

Because the 27B run stopped after validation, the comparison covers only 82 requests and 10 substrates; we therefore treat it as directional evidence rather than a measurement of the same quality as the main matrix.

Table B.15: The headroom under two backbones, on the 82 requests and 10 substrates that both cover.

Backbone	No memory	Without routing	Perfect routing	H	Relative
Qwen3.5-9B	8.54	43.10	66.44	23.34	+54.2%
Qwen3.5-27B	17.46	49.39	69.47	20.07	+40.6%

B.17 A Cascade That Escalates on a Refusal

An oracle selecting the better of the top two answers would raise the score from 44.00 to 52.74 (Section 6.2). The answer itself is the cheapest such signal, since a substrate with no useful information often says so; we test whether it can identify a zero-score first choice. We flag refusals with surface patterns such as *no mention of*, *not provided*, and *no memory supports*, then read the next-ranked substrate whenever the first refuses.

The refusal flag is associated with zero credit under the stated metric: 93.0% of flagged answers score zero, yet only 12.5% of zero-score answers are flagged. These patterns therefore miss most zero-score answers; they do not establish whether those answers are fluent, confident, or contain an unrecognized refusal. In Table B.16, the cascade escalates on 6.0% of requests and gains 0.35 points, compared with the 8.74-point gain available to the top-two oracle.

The refusal signal captures only a small share of the 8.74-point gain available within the top two candidates. Even perfect verification would leave $64.74 - 52.74 = 12.00$ points to the full oracle. **Both candidate ranking and answer verification retain substantial room for improvement.** This experiment tests one inexpensive signal for zero-score answers; it neither rules out ranking as a bottleneck nor establishes the cost effectiveness of a general verifier.

Table B.16: A cascade over the ranking of π_4 that escalates when the answer of the current substrate is a refusal.

Policy	Score	Substrate reads per request
π_4, top choice only	44.00	1.000
Cascade, up to two substrates	44.35	1.060
Cascade, up to three substrates	44.37	1.081
Best of the top two, which needs an oracle	52.74	2.000
Perfect routing	64.74	1.000

B.18 Which Inputs the Regressor Needs

We remove one feature group at a time from π_4 and rerun cross-validation on the deduplicated folds in Table B.17. Each configuration fits one regressor per substrate, or 12 per training fold. The largest point-estimate decreases are 1.26 points after removing the query representation and 1.02 after removing retrieval overlap. Removing either the dense-specific or sparse-specific probe group gives 44.00 at two-decimal precision, which does not establish identical predictions or choices, or show that the removed features were unused. Because the overlap feature still uses both retrieved sets, neither ablation eliminates a retrieval operation.

Removing the source and request-type one-hots changes the point estimate by -0.04, while the eligibility mask remains available. The $\phi = e$ and $\phi = p$ configurations score 41.90 and 41.91, compared with 43.40 for the lookup. Their feature-block names do not establish strictly query-only or probe-only policies because metadata is a separate input channel. The query-embedding nearest-neighbor configuration and vLLM-SR-style policy score 43.23 and 43.31, respectively; these comparisons do not establish that a text-only policy reaches the same score without source information.

Each removal measures a conditional change relative to the full configuration. The rounded scores and overall-policy confidence intervals cannot establish the significance of a paired ablation difference. We therefore draw no necessity or causal-attribution claim from the 1.02-point overlap decrease. Differences across overlapping feature groups are not additive contributions, and retaining overlap still requires both dense and sparse retrieval.

Table B.17: Feature ablation of π_4. Every row removes one group from $\phi = (e, u, p)$ and re-runs the cross-validation. Scores and changes are rounded to two decimals; equal entries do not imply identical predictions, selections or tree splits.

Input	Score	Change
All of $\phi = (e, u, p)$	44.00	
Without the query representation	42.74	−1.26
Without the retrieval overlap probe	42.98	−1.02
Without the source and request type	43.96	−0.04
Without the dense retrieval probes	44.00	+0.00
Without the sparse retrieval probes	44.00	+0.00

Training Distribution. LoCoMo contributes 76% of requests, whereas Eq. (1) in Section 5.1 weights the three benchmarks equally. The regressors therefore train on a distribution that differs from the evaluation metric. Reweighting each regressor's training loss to match the evaluation weights gains 0.47 points under the first protocol, from 44.00 to 44.47, and 0.22 under the second, from 36.71 to 36.93. Without identifying a single limiting factor or establishing equivalence between the training objectives, these small point-estimate changes leave most of the oracle gap.

B.19 What the Held-Out Protocol Actually Holds Out

The second protocol holds out groups drawn from 29 clusters, so the novelty encountered at test time depends on the source. Table B.18 decomposes the 7.29-point drop of π_4 using the source weights in Eq. (1) in Section 5.1. Its known-task reference uses the deduplicated folds of Appendix B.6, making the embodied contribution much smaller than a comparison against the leaked score would suggest.

The protocol combines three regimes: for LoCoMo and long-range understanding, each cluster is a memory unit, so the corpus is new while the request type remains familiar. Their policy scores change little, contributing 0.57 of the 7.29-point drop. For accurate retrieval, conflict resolution and test-time learning, each competency occupies a single memory unit, so holding it out also removes its task type from training. For ALFWorld, each cluster is a task family and both memory states remain represented in training; the task is new while the memory is familiar.

Test-time learning accounts for most of the drop, contributing 5.58 of the 7.29 points as π_4 falls from 95.00 with the competency represented in training to 28.00 when it is held out. The best fixed substrate on this source scores 94.00. This single held-out competency does not establish that learned routers generally fail on all new memories or task types. It dominates the aggregate and motivates testing this particular shift.

Table B.18: Decomposition of the protocol drop of π_4 by source, with the known side taken from the deduplicated folds. *What is held out* names the cluster that the fold removes.

Source	What is held out	Clusters	Weight	Known	Held out	Contrib.
LoCoMo	a memory unit	10	0.333	31.51	29.81	+0.57
Long-range underst.	a memory unit	10	0.083	70.42	70.42	+0.00
Accurate retrieval	whole competency	1	0.083	72.83	69.58	+0.27
Conflict resolution	whole competency	1	0.083	4.00	4.00	+0.00
Test-time learning	whole competency	1	0.083	95.00	28.00	+5.58
ALFWorld	a task family	6	0.333	39.93	37.31	+0.87
Total						+7.29

Holding Out One Whole Memory Unit. To test transfer across memories directly, we use sources with multiple units: LoCoMo has 10 conversations, long-range understanding has 10 corpora, and ALFWorld has 2 memory states. We hold out one unit at a time and train π_4 on every request outside it. This differs from the grouped protocol in Table B.18: the LoCoMo and long-range results in Table B.19 are not repeated estimates from identical splits. Grouped hold-out removes clusters together; leave-one-unit-out removes exactly one memory unit. Their training sets and generalization settings therefore differ even for LoCoMo and long-range understanding.

We exclude the embodied pair from the unseen-description comparison because its two memory states contain the same 134 task instances and 78 distinct task-description strings. Holding out one state leaves every task text in training and therefore does not test transfer to unseen descriptions. Table B.19 reports the remaining sources separately, with a cluster bootstrap over held-out units.

The transfer-gain interval includes zero. On LoCoMo, the only source with ten request-rich memory units, the policy exceeds the best fixed substrate by $+0.47$, with an interval of $[-0.76, +1.80]$. On long-range understanding it falls below the fixed reference, although that comparison covers only 71 requests across units of six to ten and its interval $[-15.3, +6.3]$ spans 21.6 points, including zero. Pooling the 20 units gives a difference of -0.75 with $[-3.71, +1.76]$.

Including the two embodied units changes both the comparison pool and its weights. Their shared task descriptions remain represented in training when one memory state is held out, so the resulting pooled gain cannot establish transfer to unseen task descriptions. We keep this pooled configuration separate from the 20-unit question-answering comparison.

Granularity Of The Grouping. The second protocol uses 29 cross-validation groups, splitting ALFWorld by task family, and gives 36.71 for $\pi_4(e, u, p)$. These groups differ from the 25 memory units: grouping by `memory_unit_id` instead merges embodied task families into two memory

Table B.19: Leave-one-memory-unit-out. The upper block excludes ALFWorld, whose two memory units share all 78 task descriptions. Intervals are a cluster bootstrap over the held-out units.

Sources	Units	Best fixed	π_4, unseen unit	Difference	Perfect
LoCoMo	10	30.36	30.83	+0.47 [−0.8, +1.8]	44.61
Long-range underst.	10	74.65	69.01	−5.63 [−15.3, +6.3]	91.55
Both	20	39.22	38.47	−0.75 [−3.7, +1.8]	54.00
+ the two ALFWorld units	22	29.91	38.29	+8.38	61.34

states, changing what is held out. The grouping choice therefore changes the generalization target, not just the number of groups.

B.20 Fusing Several Substrates Instead of Selecting One

We compare selecting one substrate with concatenating the retrieved content of several substrates in Table B.20 and Figure B.3. The comparison uses the question-answering sources, LoCoMo and MemoryAgentBench: stored retrieval traces can be assembled into a fused context and re-answered, whereas ALFWorld requires a rollout rather than one generation. We pair the best single substrate on this subset, LightRAG (M_5), with the summary index (M_3), its strongest complement by pair-oracle score on these same requests. Pair selection is therefore in-sample and does not establish which complement would be chosen on unseen requests. All conditions use the matrix's benchmark prompt and scoring metric, but retrieved content and context length vary.

Equal context budget denotes constrained fusion, with a realized mean context of 6,842 tokens versus 6,713 for M_5 alone; it still requires reading both substrates. The multi-substrate settings change both retrieved content and context length. The comparison does not isolate prompt assembly, truncation, and evidence combination. Condition-level lengths are neither additive retrieval volumes nor evidence of an untruncated union. The two-substrate mean of 10,830 tokens exceeds the sum of standalone means, $3{,}976 + 6{,}713 = 10{,}689$, while the five-substrate mean is only 416 tokens longer.

Table B.20: Fusing substrates against selecting one, on a subset of 881 of the 2,346 question-answering requests. The five fused substrates are dense retrieval, sparse retrieval, the summary index, LightRAG and the summary tree, i.e., M_1, M_2, M_3, M_5 and M_6. *Reads* is the number of substrates consulted per request, and the two oracle rows are references rather than implementable policies.

	Reads	Context (tok.)	Score	vs. M_5
Summary index M_3 alone	1	3,976	32.79	−13.40
LightRAG M_5 alone	1	6,713	46.19	—
$M_3 \oplus M_5$, equal context budget	2	6,842	46.48	+0.29 [−1.1, +1.6]
$M_3 \oplus M_5$, combined context	2	10,830	48.28	+2.09 [+0.4, +3.9]
Five substrates, combined context	5	11,246	50.88	+4.69 [+2.1, +6.8]
An oracle over M_3 and M_5	1	—	52.27	+6.08
$\pi_4(e, u, p)$ on this subset	1	—	45.79	−0.40
Perfect routing over all 12	1	—	61.68	+15.49

The subset contains 881 question-answering requests, comprising 540 from LoCoMo and 341 from MemoryAgentBench; these counts describe composition, separately from evaluation weights. Single-substrate scores differ from the corresponding matrix scores by at most 0.06 points: 32.79 versus 32.80 and 46.19 versus 46.13.

Constrained fusion differs from M_5 by $+0.29$, with an interval including zero. Multi-substrate fusion produces higher point estimates while changing both context lengths and content mixtures, so it does not isolate the value of extra context from that of combining substrates. The two-substrate oracle scores 52.27, above five-substrate fusion at 50.88, but assumes that the better answer can be identified. Accordingly, it remains an offline selection reference rather than an implemented policy with a measured one-read decision cost.

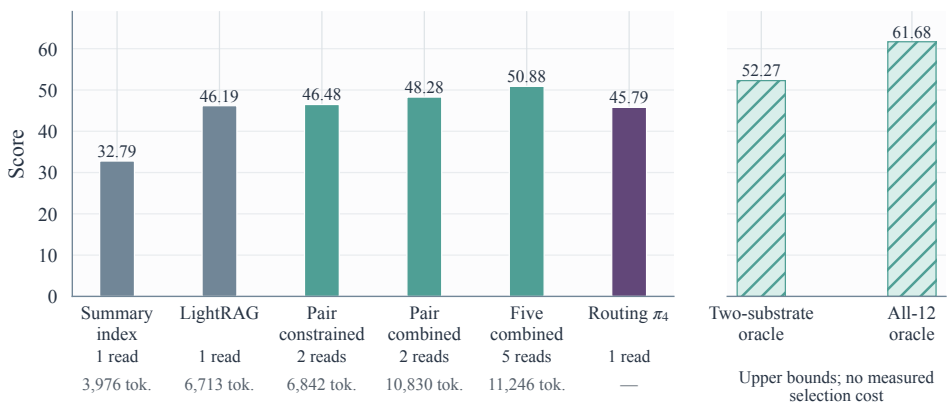


Figure B.3: Fusion and selection on 881 question-answering requests. Labels below the bars give substrate reads and mean context length when available; hatched bars are oracle upper bounds with unmeasured selection cost. Table B.20 reports paired intervals versus LightRAG.

B.21 The Outcome Matrix and Its Coverage

Figure B.4 shows the 100-request sample. Its oracle markers move across many substrate rows, and columns where fixed LightRAG is weak often contain a stronger alternative, making the complementarity behind per-request selection visible. Figure B.5 extends this view to the full pool and separates score variation from substrate-specific coverage and run-limit failures.



Figure B.4: Scores shown at the visualization resolution for a 100-request sample. Substrates are rows and requests are columns; hollow circles mark a maximizing substrate for each request. Gray marks inapplicable cells and rose marks run-limit failures; zero on the color scale is a completed score. The lower strips show fixed LightRAG and the recorded oracle.

Figure B.5 displays the full request pool in one panel per substrate. Each panel unwraps the request sequence into six contiguous rows; horizontal ticks are offsets from the request index at the start of that row. Darker completed cells indicate higher scores, gray marks inapplicable entries, and rose marks the 70 run-limit failures. These failures receive zero credit under the evaluation rule. Sorting by request identifiers mixes all sources without grouping them.

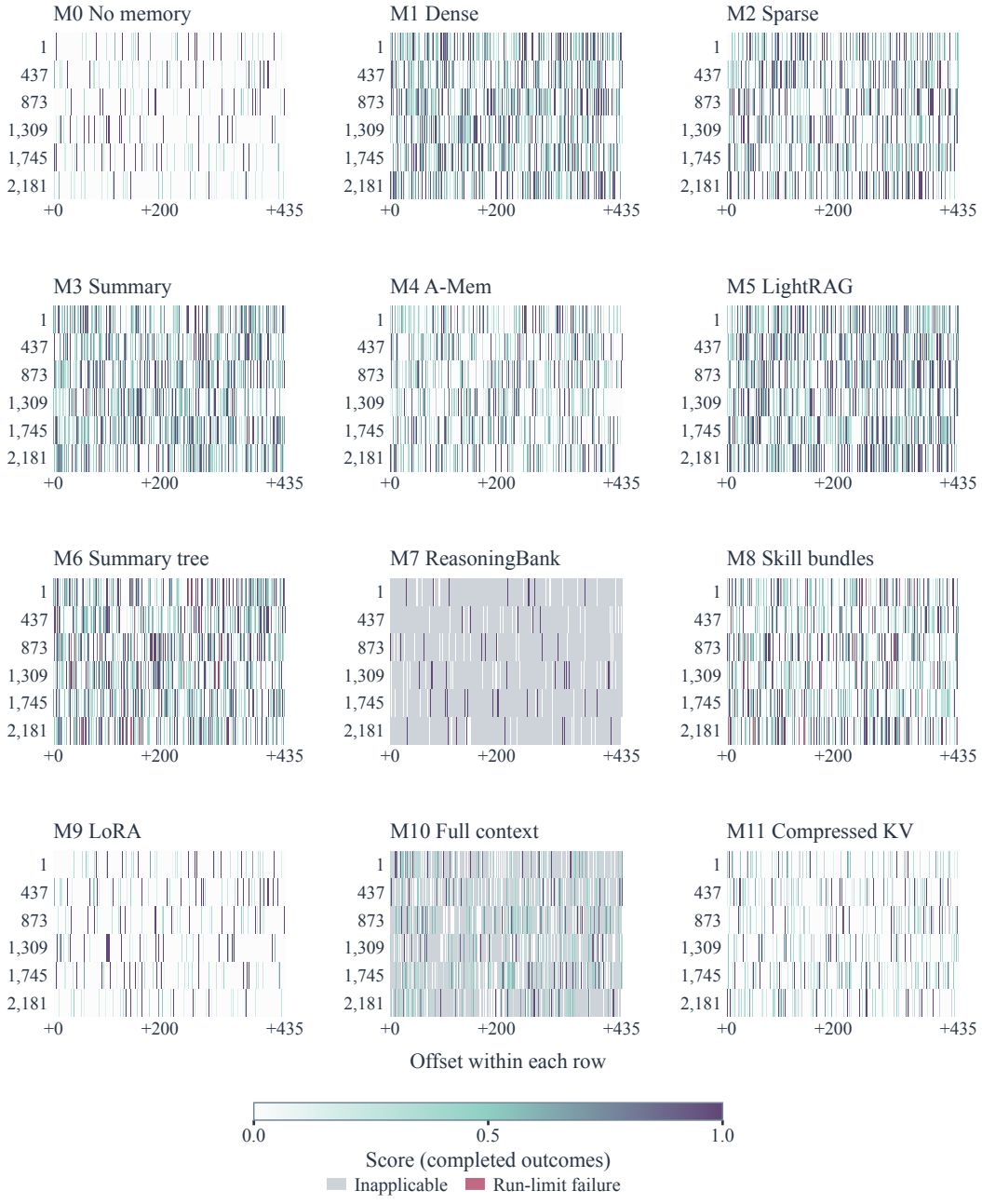


Figure B.5: Scores shown at the visualization resolution for all 2,614 requests, with one panel per substrate. Rows unwrap contiguous request blocks: left ticks mark row starts, and horizontal ticks are offsets. Gray marks 4,096 inapplicable cells; rose marks 70 run-limit failures; zero is a completed score. The final two white slots are layout padding. Substrate names are defined in Table A.1.

C REPRODUCIBILITY

Experimental Configuration. We fix the backbone, decoding settings and auxiliary models across substrates. Appendix A specifies these configurations, applicability rules and benchmark metrics; Section 6.1 defines the policies and evaluation protocols.

Evaluation Splits. Under known memory, we construct five text-grouped folds of 513–533 requests. Each contains every task source, but seven small long-range memory units are absent from at least one fold. Grouped hold-out uses 29 groups: ten LoCoMo memory units, thirteen MemoryAgentBench memory units and six ALFWorld task families. These groups are assigned to five folds of 425 to 609 requests, so each fold holds out five to seven whole groups. Its transfer target depends on the source: a new memory unit, an entire held-out competency or an ALFWorld task family.

Training excludes requests whose text appears in the held-out fold. Lookup means use training outcomes only. Both lookup variants are retained only for the first protocol. Router-comparison intervals bootstrap the 29 groups above and retain every request within each sampled group.

Offline Policy Evaluation and Online Memory. A policy selects an eligible substrate before observing its outcome. We score it from $\widetilde{\mathbf{Q}}$ using the weights in Eq. (1) in Section 5.1, without another backbone call or rollout. Each outcome is conditional on the recorded execution state s_{ij}, and the raw matrix and masks distinguish completed scores, inapplicable pairs, and run-limit failures. Fixed references and the oracle use this same matrix and weighting. For ALFWorld online accumulation, earlier substrate choices can affect actions, trajectories and subsequent memory. Recombining recorded cells neither replays those changes nor establishes a shared fixed history across columns. The oracle bounds matrix scores under this convention, not deployed online performance. Evaluating deployment requires executing the router and its memory-update rule through the episode sequence.

Execution Accounting. Of the matrix's pairs, 27,202 have completed outcomes, 4,096 are inapplicable, and 70 are run-limit failures, for a total of 31,368. Each completed cell is scored from one execution. Constructing the substrates takes 112.1 construction hours and produces 2.2 GB of state (Table B.11); routing probes and selected-answer latency are separate costs.

Run-Limit Failures. The 70 cells belong to two substrates under ALFWorld online accumulation, where the suite-level executions reached the wall-clock budget before producing task scores. They remain eligible ($a_{ij} = 1, o_{ij} = 0$), but the evaluation rule treats reaching this limit as a failure and assigns zero credit whenever one is selected. The oracle maximizes over completed applicable outcomes in $\mathcal{O}_i$; including these failures at score zero would give the same value. The reported comparison therefore measures performance under the stated execution budget. Rerunning with a larger budget would be a different evaluation.

Sensitivity to the Run Limit. As a larger-budget sensitivity check, suppose the 70 run-limit cells instead received arbitrary scores in $[0, 1]$, while all other scores and execution states remained fixed. With equal within-benchmark weight over the 268 ALFWorld requests, any fixed substrate could gain at most $100 \times 70/(3 \times 268) = 8.71$ points. Since the oracle cannot decrease, headroom would remain at least approximately $27.93 - 8.71 = 19.2$ points. This arithmetic bound does not cover reruns that change later online-memory states.

D EXTENDED EVALUATION SETTING

This appendix supplements Section 3 with the complete request-pool construction, substrate taxonomy, and preliminary backbone comparison.

D.1 REQUEST POOL AND EXECUTION STATES

Three Kinds of Experience. A long-horizon agent accumulates conversational facts, document knowledge, and procedures from executed tasks. The following details define the retained requests and histories; Table D.1 summarizes their composition.

LoCoMo (Maharana et al., 2024) provides conversational facts from 10 multi-session dialogues of about 18K words each. From its 1,986 questions, we remove 11 repeated question–category entries within the same dialogue, leaving 1,975 requests across five types. The retained categories are multi-hop (category 1, 282 requests), temporal (2, 321), open-domain (3, 96), single-hop (4, 830), and adversarial (5, 446). Adversarial questions concern events that never occurred and should be declined. We retain identical wording when the category annotations differ.

MemoryAgentBench (Hu et al., 2026) covers document knowledge through accurate retrieval, test-time learning, long-range understanding, and conflict resolution. These four competencies contribute 371 requests over 13 histories of 4K to 155K words.

ALFWorld (Shridhar et al., 2021) provides 134 unseen embodied task instances spanning six types. They use 78 distinct request texts, so the same text can identify different instances. Its history consists of the agent's past trajectories, rather than a supplied text. We evaluate two memory regimes: an *offline bank*, constructed beforehand from 204 expert trajectories (about 93K words) with no writes during evaluation, and *online accumulation*, which starts empty, executes tasks in a fixed random order, and writes one trajectory after each finished episode. Online memory evolves within an execution, so its state at an episode can depend on that execution's earlier actions and written trajectories. Each task appears once in each regime, giving 268 requests.

Table D.1: Request pool. A unit is a dialogue, document, or ALFWorld suite under one memory regime.

Benchmark	Experience	Requests	Units
LoCoMo	conversational facts	1,975	10
MemoryAgentBench	document knowledge	371	13
ALFWorld	task procedures	268	2
Total		2,614	25

History sizes above are whitespace word counts. Under the backbone tokenizer, a LoCoMo dialogue amounts to about 28K tokens. Full-context eligibility uses tokenizer counts rather than word counts; its generation reserve and profile exclusions are specified in Appendix A.

Throughout the paper, a *benchmark* is one of these three datasets. A *task source*, or simply *source*, is one of six groups: LoCoMo, each of the four MemoryAgentBench competencies, and ALFWorld.

Requests and Memory Units. A request identifies a question or task instance together with its source context. We group requests by *memory unit*: one LoCoMo dialogue, one MemoryAgentBench document, or one ALFWorld task suite under a memory regime. Question-answering requests in a unit share a source history. For online ALFWorld, unit membership does not imply identical histories across episodes or substrate executions.

Let s_{ij} denote the execution state under which substrate M_j produced the recorded outcome for request x_i. This state includes the memory contents and, where relevant, prior trajectories. Offline matrix scoring conditions on these recorded states. Selecting another column neither replays earlier episodes nor constructs the state that an online switching policy would induce (Appendix C).

The pool contains 2,614 requests of 15 types across 25 units. For 54% of requests, the history exceeds the backbone's 32K context window. Task-source and type annotations support our analysis and are explicitly available to the metadata-based policies in Section 6.1. All policies receive the evaluation-profile eligibility mask (Appendix A), which defines their candidate set, including profile-specific exclusions. Restricting ReasoningBank to ALFWorld also makes that benchmark identifiable from the mask. The evaluation therefore includes both explicit and implicit access to source information.

Identical request texts can occur in different contexts. Every ALFWorld task text appears under both memory regimes, and LoCoMo can retain identical wording with different question-type annotations within a dialogue.

Backbone and Evaluation Setup. Unless otherwise stated, we use *Qwen3.5-9B* with greedy decoding, at most 512 generated tokens, and a 32K effective context. The default auxiliary LLM and

text encoder are *gpt-4o-mini* and *text-embedding-3-small*, respectively. We hold both fixed across substrates in the main matrix to avoid confounding substrate and auxiliary-model choices. To examine model scale, we also evaluate *Qwen3.5-4B* without memory on the whole pool and *Qwen3.5-27B* on 82 requests under 10 substrates (Appendix B.16). Appendix A specifies the component configurations and metrics.

Routers requiring training follow the text-grouped and held-out-group protocols in Section 6.1. Intervals for the main router comparisons bootstrap the 29 cross-validation groups. Appendix B.4 separately compares request-level, cross-validation-group and 25-memory-unit resampling for headroom.

D.2 Substrate Families

Table 2 in Section 4 groups the action space by how experience is stored and read back. The types differ in whether they impose explicit structure, retain knowledge or procedures, and place experience in the context or the weights. Both query text and memory-derived features can inform selection. This definition fixes the object being selected without prescribing the representation or signals used to choose it.

D.3 Preliminary Backbone and Substrate Comparison

As a preliminary comparison, we examine a coarse rule that assigns structured memory to conversational facts and document knowledge, and procedural memory to task procedures. This follows the guidance of Huang et al. (2026): broad retrieval benefits long-context factual question answering, whereas excessive retrieval harms sequential decision-making. That guidance motivates the rule but does not establish a prespecified baseline. Assigning LightRAG to the question-answering benchmarks and ReasoningBank to ALFWorld gives the full-pool benchmark-level optimum shown in Figure 2(b) in Section 6.2. We therefore treat its score of 43.66 as an in-sample reference, rather than an independently selected baseline. Although this rule requires no per-request outcomes at inference time, our main learned-policy comparison uses the cross-validated lookup table over six task sources at 43.40 (Section 6.2).

We compare the rule with (1) routing among backbones of different scales without memory and (2) using one fixed memory for every request. To enable the first comparison, we additionally execute the whole pool under *Qwen3.5-4B*, allowing a router to select a backbone per request. We report the average pool score on the 0–100 scale defined in Section 5.1.

Table D.2 shows that the coarse substrate rule exceeds the best fixed memory by 18.58%. A semantic router between the two backbones scores 17.88, matching the fixed 4B backbone at the displayed precision. This aggregate equality does not establish identical per-request choices. The fixed 4B backbone also exceeds the 9B backbone without memory (17.88 versus 15.62), so the tested configurations are not ordered by parameter count. The comparison does not isolate model size from reasoning mode, generation limits or serving configuration. Even an oracle over the two backbones reaches only 22.95, below the 36.82 of a single fixed memory.

For these backbones and this pool, selecting a model without memory does not recover the score available from a fixed memory substrate. The comparison motivates memory selection as an additional routing decision. Section 5 measures its observed-matrix headroom using the available outcomes and their status masks.

E Policy Implementation and Extended Analyses

This appendix gives the implementation details and extended analyses for Section 6. The reported comparisons retain their original request pools, candidate sets and scoring conventions.

E.1 Policy Implementations

All five policy families implement Eq. (5) in Section 6.1 using information available before the selected substrate answers. Training scores use $\widetilde{\mathbf{Q}}$; the raw matrix and masks retain whether an

Table D.2: Preliminary comparison. The rule assigns LightRAG to LoCoMo and MemoryAgentBench, and ReasoningBank to ALFWorld. The assignment coincides with the full-pool per-benchmark oracle and is an in-sample reference. The relative improvement is computed over the best fixed substrate.

Strategy	Score
Semantic routing between *Qwen3.5-4B* and *9B*, no memory	17.88
an oracle choice between the two, no memory	22.95
Qwen3.5-9B only, no memory	15.62
Qwen3.5-4B only, no memory	17.88
Fixed dense retrieval	30.83
Fixed LightRAG (the best fixed substrate)	36.82
Substrate-type rule (in-sample reference)	**43.66** (+18.58%)

outcome was observed. The following specifications distinguish the feature blocks, supervision and model variants.

- π_1 **as a lookup table.** We partition training requests by metadata $c(x_i)$, either task source or task source and request type. Each key stores the per-substrate arithmetic means over requests with that exact key, giving $\hat{q} = \bar{q}\big(c(x_i), M_j\big)$ without fitting a model. We report both lookup variants only under the text-grouped protocol; their behavior when an exact key is absent is not part of the retained comparison. Eq. (5) in Section 6.1 restricts selection to eligible substrates. Zero-credit entries remain in the training means, and the lookup uses no held-out outcomes.

  The *task source* and *task source + type* variants have six and fifteen possible keys, respectively. The six task sources are LoCoMo, the four MemoryAgentBench competencies, and ALFWorld; benchmark instead refers to the three parent datasets. This table is the cheapest implementation and can be written by hand. We refit it on each training split, so a source can map to different substrates across folds: conflict resolution, for example, maps to the summary index in some and A-Mem in others.

  We grant metadata to the lookup to measure how far this simple policy can go, and permissively supply it to π_2 and π_4 as well. Appendix B.18 reports a small point-estimate change when explicit metadata is removed from π_4. Every policy receives the evaluation-profile eligibility mask. Because ReasoningBank is eligible exactly on ALFWorld, the mask provides this task-source indicator even when it is omitted as an explicit feature.

- π_2 **as nearest-neighbor retrieval.** We encode requests with *all-MiniLM-L6-v2* and retrieve training requests by cosine similarity. Averaging the recorded scores of the $k = 25$ nearest neighbors gives $\hat{q} = \frac{1}{k}\sum_{x' \in \mathcal{N}_k(x_i)} \tilde{q}(x', M_j)$ under the convention of Section 5. Run-limit failures carry zero under this rule. This variant is *kNN (query)*. The *kNN (query + memory)* variant also matches memory-unit statistics, and *kNN (query + memory + probes)* adds the retrieval probes described below. Like retrieval-augmented generation, it uses similar training requests to inform a new decision and requires only indexing rather than model training.

- π_3 **as an encoder plus a classifier.** We fit a logistic-regression classifier on the same sentence embeddings to predict the request category, then map that category to a substrate through the lookup table: $\pi(x_i) = \arg\max_{M_j \in \mathcal{M}_i} \bar{q}\big(\hat{c}(e(x_i)), M_j\big)$. This is the architecture of deployed semantic routers. We follow vLLM Semantic Router (Wang et al., 2025), which classifies request intent with an encoder, and also run the *semantic-router* library, which matches requests against example utterances for each route in the same embedding space. We call these *vLLM-SR style* and *semantic-router library*. To cover the neural classifiers often fine-tuned in deployed systems, *MLP, category* replaces logistic regression with a two-layer MLP. The *MLP, substrate directly* variant instead skips the table and predicts the substrate.

- π_4 **as a gradient-boosted regressor.** We fit one LightGBM regressor per substrate and select the highest predicted score, $\hat{q} = f_{\theta_j}\big(\phi(x_i)\big)$, where θ_j is fitted to the training-fold cells in column j of $\widetilde{\mathbf{Q}}$. Replacing the trees with a two-layer MLP using the same inputs gives *MLP,* $\phi = e$ and *MLP,* $\phi = (e, u, p)$.

  This implementation consumes the most supervision and is the easiest to extend with additional inputs. We vary ϕ to obtain $\pi_4(\phi)$: within the named feature blocks, $\phi = e$ uses the query embedding, $\phi = (e, u)$ adds memory-unit statistics u, $\phi = p$ uses *retrieval probes*, and $\phi = (e, u, p)$

includes all three. These feature-block names do not establish strictly query-only inputs: metadata and the eligibility mask are additional information channels, and only the explicit metadata ablation isolates removal of the former. We obtain the probes by running dense and sparse retrieval once before answering. Their summaries describe how well a request matches stored experience, including the top-1 and mean similarity, their gap, and the overlap between retrieved sets.

- π_5 **as an LLM judgment.** We prompt a language model with the request and a one-line description of each substrate, asking it to return a choice directly: $\pi(x_i) = \mathrm{LLM}(x_i)$, with no explicit $\hat{q}$. We test *the backbone itself*, *Qwen3.5-9B*, and seven external models from four vendors: *gpt-4o-mini*, *gpt-5.2*, *GPT-5.6-Terra*, *Claude Opus 4.5*, *Claude Opus 5*, *Kimi K3* and *Llama 3.3 70B*. All use greedy decoding. Including Qwen gives eight models from five vendors. Several OpenAI and Anthropic models allow us to compare routing behavior across configurations within those families. We exclude two additional reasoning models because the generation budget truncates their chain of thought before they emit an answer, leaving empty responses (Appendix B.8).

Instantiating these five implementations with different keys, features and model classes gives 23 variants. For example, the table uses two keys, and the regressor has six feature-set or model-class variants. Except for π_5, every policy uses only cells of $\widetilde{\mathbf{Q}}$ from the training folds. Their supervision requirements differ: π_1 and π_3 need one aggregate per category, π_2 needs training-request rows, and π_4 needs every training cell. These cells are read offline; no policy accesses a substrate's outcome on the request it is routing or reads held-out cells to make that decision. We also report random selection and the best substrate on the training folds, the no-routing reference of Section 5.

Reading the Main Comparison. Table E.1 below reports a cross-validated fixed-substrate reference of 36.21 rather than $S_{\mathrm{fix}} = 36.82$, because the substrate is selected on the training folds. LLM judgment and random selection require no fitting to the training folds. The random reference has reported scores of 26.14 and 26.10 under the two protocols. The scores are rounded to two decimals and confidence-interval endpoints to one decimal; a displayed zero endpoint does not reveal its unrounded value. Significance follows the reported marker. Paired comparisons use the cross-validated task source table at 43.40, whereas the separate one-bit rule at 43.66 is an in-sample full-pool reference.

E.2 Extended Protocols and Evaluation Scope

We distinguish two forms of novelty that a deployed policy can encounter:

- **New requests under a known memory.** Training includes the memory units, but the test requests are new. This represents an agent receiving an unseen request from a user or document it has already served.
- **New requests under a held-out group.** The grouped comparison holds out clusters drawn from 29 groups: LoCoMo memory units (10), MemoryAgentBench memory units (13), and ALFWorld task families (6). For LoCoMo and one MemoryAgentBench competency, a held-out cluster is an unseen memory unit, the transfer setting of interest. For the other three competencies, it is the entire competency, so training also excludes that task type. For ALFWorld, it is a task family while both memory states remain represented in training.

  We interpret these regimes separately because they test different forms of novelty. In particular, holding out a complete source or task family can remove an exact lookup key; both lookup variants are therefore outside our retained grouped comparison.

For variants retained under both protocols, the policy implementation and outcome matrix are unchanged; the protocols differ in fold construction. Under the first, identical request texts stay in one fold. Under the grouped protocol, clusters remain intact and we remove any training request whose text occurs in the held-out fold. This preserves the group partition while preventing text overlap, including the 78 ALFWorld descriptions repeated under both memory states. By contrast, permuting requests within each memory unit leaves 256 of the 268 embodied requests with a verbatim training-set twin, allowing text-based policies to exploit leakage (Appendix B.6).

We evaluate policies requiring training by cross-validation. Router-comparison intervals bootstrap the 29 groups above, keeping requests within a group together: these are the 23 question-answering

Table E.1: Cross-validated scores for 23 policy variants and three references. The first protocol evaluates new requests under known memory; grouped hold-out mixes memory, competency and task-family shifts. The final column reports paired differences from the task-source lookup at 43.40, with 95% intervals over 29 groups; * marks intervals excluding zero.

	Policy	New requests	Held-out group	vs. lookup
	Random	26.14	26.10	-17.26^{*} [−24.4, −10.8]
	Best fixed substrate, cross-validated	36.21	35.82	-7.19^{*} [−12.9, −2.6]
π_1	task source	43.40	–	–
	task source + type	43.12	–	−0.28 [−1.3, +0.9]
π_2	kNN (query)	43.23	37.83	−0.16 [−1.9, +1.7]
	kNN (query + memory)	43.35	36.51	−0.05 [−2.7, +2.5]
	kNN (query + memory + probes)	43.26	38.74	−0.14 [−2.5, +2.2]
π_3	vLLM-SR style	43.31	40.20	−0.08 [−0.5, +0.2]
	semantic-router library	43.17	43.05	−0.22 [−1.0, +0.4]
	MLP, category	43.17	40.05	−0.23 [−0.9, +0.2]
	MLP, substrate directly	29.88	18.43	-13.52^{*} [−19.7, −8.0]
π_4	$\phi = e$	41.90	35.90	−1.50 [−5.2, +2.4]
	$\phi = (e, u)$	43.30	33.55	−0.10 [−2.3, +2.4]
	$\phi = p$	41.91	36.94	−1.49 [−3.7, +0.8]
	$\boldsymbol{\phi = (e, u, p)}$	**44.00**	36.71	+0.60 [−1.7, +3.1]
	MLP, $\phi = e$	42.69	36.48	−0.71 [−3.2, +1.5]
	MLP, $\phi = (e, u, p)$	41.71	37.47	−1.69 [−4.9, +1.3]
π_5	*gpt-4o-mini*	35.85	35.85	-7.55^{*} [−12.3, −2.8]
	gpt-5.2	31.29	31.29	-12.11^{*} [−22.5, −4.7]
	GPT-5.6-Terra	34.84	34.84	-8.55^{*} [−12.2, −5.0]
	Claude Opus 4.5	38.97	38.97	-4.42^{*} [−9.5, −0.1]
	Claude Opus 5	34.91	34.91	-8.48^{*} [−13.3, −4.0]
	Kimi K3	39.78	39.78	−3.62 [−8.0, +0.0]
	Llama 3.3 70B	43.68	43.68	+0.28 [−1.4, +1.8]
	the backbone itself	40.31	40.31	−3.09 [−8.6, +1.1]
	Perfect routing	64.74	64.74	–

memory units and six ALFWorld task families, not the 25 memory units alone. We also report recovered headroom, $\frac{S(\pi)-S_{\text{fix}}}{H}$, which is 0% without routing and 100% for perfect routing.

Scope of Matrix Scoring. We score routers by selecting recorded outcome-matrix cells. Under ALFWorld online accumulation, earlier episodes and substrate choices can affect later memory states. The recombined score is therefore conditional on the recorded histories; it is not the result of executing the router as an online policy. Deployment claims require separate rollouts that validate the policy together with its history-update protocol.

E.3 Construction, Supervision and Request Costs

Deployment requires maintaining the candidate substrates and obtaining affordable supervision. We consider three resulting costs: the supervision used to build the policy, the substrates kept available, and the time spent per request.

Cost Of The Supervision. All policies except π_5 require prior outcomes from requests executed under several substrates. Their candidate memories must also be constructed; building the full set for this pool takes 112.1 construction hours, including model calls and local computation (Appendix B.9). This cost is separate from answering a new request.

To examine supervision requirements, we train π_4 on random fractions of the training cells. It scores 38.97, 42.35, 43.25 and 44.00 at 10%, 25%, 50% and 100%, respectively, averaged over three subsamples. Half of the matrix yields 98.3% of the full-matrix score. However, the 25% and 50% configurations remain below the task source table at 43.40, by 1.05 and 0.15 points. More supervision improves this regressor, but its full-matrix score remains statistically indistinguishable from the lookup's.

How Many Substrates To Maintain. Maintaining all 12 substrates costs 112.1 construction hours and 2.2 GB (Appendix B.9), which may be unaffordable for an agent. We therefore grow a subset greedily, adding the substrate that most increases perfect routing, and evaluate both the oracle and the best router within each subset. In Figure 3(b), the first three substrates raise perfect routing from 36.82 to 51.92, while the last six add only 3.59. Six substrates reach an oracle score of 61.15 at 82.4 construction hours: 94% of the score with all 12 substrates at approximately 73.5% of its build cost ($82.4/112.1$). The best policy within that subset scores 43.29, against 44.00 with all 12.

The policy curve flattens earlier than the oracle curve, gaining 6.47 points over the first six substrates and 0.71 over the last six. Thus the policy exploits fewer substrates than contribute oracle headroom. Its curve is also non-monotone: it peaks at 45.13 with ten substrates before reaching 44.00 with all 12. We select the greedy subsets on the full pool, so this is an in-sample candidate-set comparison. The 45.13 configuration is outside the 23 full-library variants and their primary-comparison paired tests. As in the no-memory comparison, each point describes a complete candidate-set configuration.

A small subset exposes much of the observed oracle score at a lower reported construction cost. Because both subset selection and evaluation use the full pool, this curve does not estimate performance on unseen memory units.

Cost Per Request. Each request incurs both selected-substrate answering time and routing-decision time. Figure 3(c) displays the recorded selected-answer component, excluding the dense and sparse probes required by π_4 and the construction and storage of the candidate memories. Its timings describe the evaluated workloads rather than an end-to-end deployment comparison.

Figure 3(d) summarizes approximately measured local routing components together with benchmark-calibrated ranges under a warm setup with models already loaded. For π_1, we round the recorded lookup arithmetic of less than 0.001 ms upward and report less than 0.01 ms, allowing for ordinary bookkeeping. For the embedding-only variants of π_2 and for π_3, the same *all-MiniLM-L6-v2* encoder dominates the recorded 0.124-ms nearest-neighbor arithmetic and the lightweight classifier. Public Sentence Transformers documentation reports 18,000 GPU and 750 CPU queries per second for a comparable six-layer MiniLM search encoder, while its inference guide cautions that tuned-batch throughput need not equal single-request latency.[1] We report an approximate batch-one range of 5–30 ms for these configurations.

For configurations with retrieval probes, the reported dense and sparse medians are 0.52 and 0.003 seconds. Adding the same 5–30-ms encoder range and, for π_4, 0.032 ms of LightGBM scoring yields 0.528–0.553 seconds, reported as approximately 0.53–0.56 seconds for both the probe-augmented π_2 variant and the main π_4 model. Each π_5 decision instead requires one model completion, but this online decision may be served locally or through a hosted API. NVIDIA NIM reports concurrency-one time to first token near 81 ms for an 8B model on one L40S and 82 ms for Llama 3.3 70B on two H100s with a 1,000-token input; their subsequent inter-token latencies are about 22 and 19 ms.[2] Based on these serving measurements, we report approximately 0.1–1 second for a short constrained choice from a warm local accelerator. A hosted request also incurs service and round-trip variability, so we report a separate approximate range of 1–10 seconds.[3] Both ranges are sensitive to hardware, prompt length, generated tokens and load. Queueing, network transit and cold starts are excluded. They are approximate operating ranges assembled from local components and public measurements rather than timings collected for every policy in one common runtime environment.

The plotted timings are descriptive aggregates, not a controlled runtime reduction. Coverage differences and unresolved serving-cost components prevent an intrinsic speed ranking (Appendix B.9). End-to-end cost includes routing probes and model calls as well as the amortized 112.1 construction hours for the full candidate set.

E.4 Selection, Agreement, Verification and Fusion

Observations. Table E.1 compares the 23 main variants over the full library of 12 substrates, subject to the evaluation-profile eligibility mask. Separate candidate-set and training-objective com-

[1] Sentence Transformers model table and inference guide.
[2] NVIDIA NIM 8B benchmark and Llama 3.3 70B benchmark.
[3] OpenAI latency guide.

parisons fall outside its paired tests, e.g., evaluation-weighted training reaches 44.47 with the full library (Appendix B.18). We draw four observations from the table that its rows do not state individually.

- **The implementations differ far less from each other than from not routing.** The best of the 23 variants improves on fixed LightRAG by a relative 19.51%, whereas the 14 variants that are neither an LLM judge nor the degenerate classifier range only from 41.71 to 44.00, with 10 inside a single point.
- **Eight comparisons exclude zero, all of them negatively**, i.e., random selection, the cross-validated best fixed substrate, the direct-substrate MLP, and five LLM judges. No variant excludes zero in the positive direction.
- **The gains concentrate in the coarse source distinction.** This aggregate does not determine the sign of each source contribution, and nearby aggregate scores can arise from different per-request choices (Appendix B.3).
- **Seven of the eight judges score below the six-entry table, and four score below not routing.** In the OpenAI panel, *gpt-4o-mini*, *gpt-5.2* and *GPT-5.6-Terra* score 35.85, 31.29 and 34.84, respectively, so the last improves on *gpt-5.2* rather than continuing the decline. These point estimates compare each judge with the lookup table rather than with another model generation.

Discussion. About three quarters of the single-execution matrix gap remains after the primary full-library comparison, and its transferable component has not been isolated. Four threads bear on that shortfall, each developed in the appendix subsection that owns its table. *Task shift* decomposes the grouped-protocol drop by source and reports the leave-one-memory-unit-out comparison (Appendix B.19). *The no-memory option* separates a predicted preference for answering without memory from a tie-broken assignment to M_0 (Appendix B.12). *Choice agreement* shows that policies disagree on roughly three quarters of requests while scoring within a point of each other, and reports the decisive subset on which that overlap is measured (Appendix B.14). *Verification* bounds what a later commitment could recover, i.e., the oracle over the top two predictions scores 52.74 and the top three reaches 56.26, recovering at most 12.26 of the 20.74-point gap, whereas the refusal-triggered cascade scores 44.35 at 1.06 substrate reads per request (Appendices B.7 and B.17).

Fusing several substrates outperforms the tested policy on the subset where we can measure it, which is why selecting exactly one substrate is a scope decision rather than a finding. Combining the best single substrate with its strongest complement reaches 48.28 against 46.19, using about 1.61 times as much context, and five-substrate fusion adds a further 3.8% of context for 50.88. However, an oracle over just those two substrates scores 52.27, above five-substrate fusion, so the comparison motivates better selection rather than more reading (Appendix B.20).

F Additional Headroom Analysis and Limitations

F.1 Granularity and Conditional Uncertainty

For comparison, the separate full-pool request-type partition has 15 groups, uses seven substrates and scores 45.69. Because request types cross memory units, this partition is not a refinement of the memory-unit partition and is omitted from the granularity chain. The memory-unit–type partition of Figure 2(b) comprises 74 nonempty groups, i.e., 49 LoCoMo groups, 13 MemoryAgentBench groups and 12 ALFWorld groups, since one LoCoMo dialogue contains four rather than five request types. These are full-pool descriptive partitions, which do not separate repeatable request-level differences from execution noise. Nineteen of the 25 units retain more than five points of internal headroom, whereas the remaining six each contain only six to eight requests (Table B.7). The two ALFWorld units carry one third of the aggregate weight, so the memory-unit intervals of Table B.6 are the widest of the three resampling levels.

F.2 Scope and Further Evaluation

Limitations. *One backbone.* We construct the main matrix with Qwen3.5-9B. On the smaller comparison of 82 requests and 10 substrates shared with Qwen3.5-27B, relative headroom decreases

from 54.2% to 40.6%. Transferring the 9B-selected choice to the 27B backbone scores 55.88, compared with 49.39 for the 27B best fixed substrate and 69.47 for its own oracle (Appendix B.16), leaving 13.59 points to that oracle. The comparison is limited to this small matched subset and does not isolate parameter count from reasoning mode or generation settings. A second full matrix would extend its scope.

One measurement per cell. Each observed main-matrix cell is scored from one execution with greedy decoding, while auxiliary-model construction, maintenance and online trajectories can vary between executions. Maximizing over these outcomes may inflate the oracle. Resampling the recorded matrix quantifies sampling variation conditional on those outcomes and cannot correct this execution-noise bias. Neither the 75.9% headline difference nor the remaining three quarters of the matrix gap estimates a transferable selection gain. Independent repeated executions are needed to produce that estimate.

Offline scoring of online memory. We score routers by combining recorded outcomes. In ALFWorld online accumulation, however, routing can change the trajectories written to memory and thus the histories available to later episodes. The offline score is conditional on those recorded histories. Evaluating a deployed router along its own history requires new executions with an explicit history-update protocol.

Limited auxiliary-model coverage. The main matrix uses *gpt-4o-mini* for memory construction and *text-embedding-3-small* for embeddings. We do not establish robustness to replacing these components or to changing the memory-building protocol.

Configuration coverage. Results depend on the fixed evaluation-profile eligibility mask and substrate configurations. Each substrate uses its reference implementation's read budget, and these budgets differ by an order of magnitude. Substrate-specific settings are not exhaustively tuned, and direct calibration against each original paper's results remains incomplete. Underperformance may therefore reflect the evaluation setup as well as the substrate design. Reasoning mode, token-cap truncation and precise serving behavior are not isolated by the reported aggregate comparisons.

Selecting exactly one substrate. We restrict the action space to one substrate per request. Fusion offers another option on the 881-request question-answering sample (Appendix B.20). The equal-context-budget pair scores $+0.29$ above the best single substrate, with an interval including zero. The combined pair uses 10,830 tokens against 6,713 for the single substrate and gains 2.09 with $[+0.4, +3.9]$; five-substrate fusion gains 4.69 with $[+2.1, +6.8]$. These comparisons change both evidence and final context length, and the token means do not isolate formatting, deduplication, or truncation effects. **The two-substrate oracle scores 52.27, above five-substrate fusion at 50.88, while the tested policy scores 45.79.** This oracle requires outcome scores to select its substrate and is an upper bound. Jointly deciding how many substrates to read and which ones to use would extend the evaluated action space.

Latency and maintenance. Selected-answer timings exclude routing probes and assume that candidate substrates are already built. Constructing the full set required 112.1 construction hours and 2.2 GB of state. The timing summaries do not separate workload coverage, generated length and serving costs, so they establish neither an intrinsic speed ranking nor a controlled runtime reduction. Deployment cost must also include probes, decision calls and memory maintenance.

Metric dependence. LoCoMo token-level F1 can under-credit correct refusals. The adapted MemoryAgentBench metrics also differ from the native exact-match criteria, and substring matching can reward answers that contain a reference item without resolving the task cleanly. The reported gaps condition on these metrics; testing their robustness requires direct regrading with a prespecified rubric.

Scope of the pool. We evaluate 2,614 requests over 25 memory units from three benchmarks. Twenty-five units provide few independent clusters, resulting in wide intervals under the unseen-memory protocol. Code-generation memory, e.g., BigCodeBench-Hard, requires containerized grading and is outside the scope of this evaluation.

Simple policies only. We compare five implementations of a one-shot decision and one cascade that escalates on refusal. We evaluate two-answer verifiers separately, but do not study a learned policy that uses the first answer to decide whether to pay for a second read. The top-candidate oracle gaps motivate this extension.

What We Would Measure Next. Candidate ranking and answer verification offer two directions for further evaluation. An oracle over the top two predictions scores 52.74, recovering 8.74 points over the first choice at 44.00 while leaving 12.00 points to the full oracle. Expanding to the top three reaches 56.26, recovering at most 12.26 points while still leaving 8.48. These gaps separate the potential value of verification from the candidates a policy makes available.

The *gpt-5.4-mini* verifier comparison gains $+2.25$, with $[-0.8, +4.9]$ including zero (Appendix B.7); this result does not establish a positive gain. We would next compare output budgets and decision rules across independent units, including whether a verifier can identify when another read is worth its cost. The simpler refusal-triggered cascade flags only 12.5% of zero-score answers.