DID YOU CHECK THE RIGHT POCKET? COST-SENSITIVE STORE ROUTING FOR MEMORY-AUGMENTED AGENTS

Madhava Gaikwad
Independent Researcher
gaikwad.madhav@gmail.com
ICLR 2026 Workshop on Memory and State in LLM-Based Agents

ABSTRACT

Memory-augmented agents maintain multiple specialized stores, yet most systems retrieve from all stores for every query, increasing cost and introducing irrelevant context. We formulate memory retrieval as a store-routing problem and evaluate it using coverage, exact match, and token efficiency metrics. On downstream question answering, an oracle router achieves higher accuracy while using substantially fewer context tokens compared to uniform retrieval, demonstrating that selective retrieval improves both efficiency and performance. Our results show that routing decisions are a first-class component of memory-augmented agent design and motivate learned routing mechanisms for scalable multi-store systems. We additionally formalize store selection as a cost-sensitive decision problem that trades answer accuracy against retrieval cost, providing a principled interpretation of routing policies.

1 INTRODUCTION

Consider a user asking an agent: "What was my weight before I started the diet?" This query depends on historical information from a previous chat session and therefore should draw from long-term memory. It does not require the current-session context, and it typically does not require raw transcripts. Yet many memory-augmented agent frameworks retrieve from multiple stores or large aggregated memory contexts regardless of query type, relying on the language model to filter relevant information during generation (Lewis et al., 2020; Liu et al., 2023a; Chase, 2023).

Uniform retrieval carries two costs. First, it creates **computational waste** by querying memory stores that cannot contain the answer. Second, it can cause **accuracy degradation**: irrelevant or noisy retrieved context can hinder the model's ability to identify answer-bearing information, particularly in long-context settings where additional tokens reduce the signal-to-noise ratio (Liu et al., 2023b; Yu et al., 2024; Zhang et al., 2026).

We formalize store selection as a **routing problem**. Given a query, the router chooses which stores to search *before* retrieval. This decouples store selection from within-store ranking and makes the accuracy–cost tradeoff explicit.

Our evaluation separates two questions. First, do routing policies choose the stores that should be searched for a given query? We test this with *synthetic routing labels* derived from query taxonomies (Section 5.1). Second, if the router chooses the right stores, does this improve downstream QA accuracy on *real LLMs*? We test this with an oracle router and fixed store subsets (Section 5.2). This setup isolates the value of store selection from model capability and from within-store retrieval quality.

Contributions. (1) We define routing metrics (coverage, exact match, waste) that capture complementary aspects of store selection quality. (2) We introduce simple routing baselines, including a conservative hybrid heuristic that emphasizes coverage. (3) We evaluate routing policies using both synthetic labels and real LLM-based QA, showing that selective store choice can reduce tokens while improving accuracy, and that uniform retrieval can underperform despite using more context. (4) We formalize store routing as a cost-sensitive subset-selection problem, providing a decision-theoretic framework that explains when selective retrieval improves both efficiency and accuracy.

2 RELATED WORK

Adaptive Retrieval. Prior work studies *when* to retrieve (Self-RAG Asai et al. (2023), FLARE Jiang et al. (2023)) and *how much* to retrieve (Adaptive-RAG Jeong et al. (2024)). Recent methods explore *which retriever* to use through learning-to-rank formulations Kim & Diaz (2025) or self-reflection mechanisms Wu et al. (2026). SmartRAG Gao

et al. (2025) uses reinforcement learning for joint retrieval and generation optimization. We consider a complementary question: *where* retrieval should occur when memory is partitioned across heterogeneous stores with distinct semantic roles (e.g., episodic vs. semantic memory).

Memory Systems. MemGPT Packer et al. (2024) implements hierarchical memory with explicit management operations. Generative Agents Park et al. (2023) rely on reflection and importance scoring to consolidate information over time. Recent architectures organize memory hierarchically for multi-agent reasoning Zhang et al. (2025). These systems focus primarily on memory *organization*; we instead focus on memory *access*, specifically cost-sensitive routing across stores prior to retrieval. Our store taxonomy draws on cognitive distinctions between episodic and semantic memory Tulving et al. (1972), and translates these distinctions into operational routing decisions.

Multi-Index and Federated Retrieval. The information retrieval literature has long examined federated search across heterogeneous collections Callan (2002); Si & Callan (2003); Shokouhi et al. (2011), where resource selection algorithms estimate relevance distributions for each collection Lu & Callan (2006). These methods route queries across independent search engines; in our setting, the same principles apply to agent memory stores that differ semantically (e.g., current dialogue vs. historical summaries) rather than by document source.

Retrieval Routing and Selection. Recent RAG work shows that routing queries across specialized retrievers can improve both efficiency and accuracy Kim & Diaz (2025); Wu et al. (2026); Guo et al. (2025). ExpertRAG Gumaan (2025) applies mixture-of-experts routing to context selection, while RAP-RAG Ji et al. (2025) plans adaptive retrieval sequences for multi-hop reasoning. Unlike retriever routing, our setting routes across persistent memory stores whose contents reflect distinct temporal or semantic roles (e.g., STM vs. LTM), requiring store-level rather than passage-level decisions.

Context Noise in Long Documents. Long-context modeling faces signal-to-noise challenges when irrelevant tokens distract attention Li et al. (2024b). Recent analyses show that long-context LLMs often underperform targeted retrieval despite larger windows Li et al. (2024a), and that retrieval decisions themselves benefit from uncertainty-guided policies Chen et al. (2026). Our work complements these findings by showing that *store-level* noise, retrieving information from irrelevant memory types, can further degrade performance, especially when each store contributes hundreds of tokens.

Memory Benchmarks. LoCoMo Maharana et al. (2024) evaluates multi-session conversational memory. LongMemEval Wu et al. (2025) benchmarks knowledge updates and temporal ordering. We use their question taxonomies to derive store-labeling protocols.

3 Problem Formulation

3.1 Memory Architecture

A **memory store** is an independent index containing semantically related information. Each store can be queried separately, and the retrieved content is concatenated into the LLM context. Following MemGPT, we consider four stores:

Short-Term Memory (STM) holds the current conversation, typically the last N turns. Queries such as "what did I just mention" or "today's meeting" require STM.

Summary Store contains compressed user facts, including preferences, biographical details, and ongoing projects. Queries such as "what is my phone number" target this store.

Long-Term Memory (LTM) stores summaries of past conversations. Queries about earlier discussions ("what did we talk about last week") rely on LTM.

Episodic Memory preserves raw transcripts. Queries that require exact wording or precise timestamps may need episodic memory.

3.2 The Routing Problem

Let $\mathcal{S} = \{\text{STM}, \text{Sum}, \text{LTM}, \text{Epi}\}$ denote the set of stores, where Sum is the Summary store and Epi is episodic memory. Given a query q, a routing policy π selects stores $\hat{G} = \pi(q) \subseteq \mathcal{S}$. The system retrieves content from the selected stores and prompts the LLM.

Let G denote the **ground-truth stores**, the stores containing the information needed to answer q. In our synthetic routing evaluation, we derive G from query type: a "single-hop fact" query has $G = \{\text{Sum}\}$; a "temporal comparison" query has $G = \{\text{LTM}, \text{Epi}\}$. See Section 9 for the full mapping.

3.3 Evaluation Metrics

We evaluate routing with three metrics:

Coverage measures whether all necessary stores were included:

$$\text{Coverage} = \frac{1}{N} \sum_i \mathbf{1}[G_i \subseteq \hat{G}_i] \tag{1}$$

Under our evaluation protocol (full-store concatenation), a coverage failure means the answer is not retrievable from the provided context.

Exact Match (EM) measures whether the policy selects precisely the required stores:

$$\text{EM} = \frac{1}{N} \sum_i \mathbf{1}[G_i = \hat{G}_i] \tag{2}$$

High EM corresponds to efficient routing without over-retrieval.

Waste counts unnecessary stores retrieved:

$$\text{Waste} = \frac{1}{N} \sum_i |\hat{G}_i \setminus G_i| \tag{3}$$

Waste is a store-level proxy for token cost because each additional store contributes retrieved tokens, and it can also reduce accuracy by introducing contextual noise.

Cost. We measure cost as **context tokens**, the number of tokens inserted into the prompt (counted via tiktoken). This serves as a direct proxy for inference cost.[1]

3.4 Cost-Sensitive Store Routing: A Decision Framework

Store selection can be viewed as a cost-sensitive subset-selection problem. Let $\mathcal{S}$ denote the set of available memory stores, and let c_s represent the retrieval cost associated with store $s \in \mathcal{S}$ (e.g., context tokens or infrastructure access cost). Given a query q, a routing policy selects a subset of stores $G \subseteq \mathcal{S}$ before retrieval.

The objective is to balance answer quality against retrieval cost. Let $\text{Acc}(q, G)$ denote the expected probability that the downstream LLM produces a correct answer when stores G are retrieved. A cost-sensitive routing policy can therefore be defined as

$$\pi^*(q) = \arg\max_{G \subseteq \mathcal{S}} \left[\mathbb{E}\big[\text{Acc}(q, G)\big] - \lambda \sum_{s \in G} c_s \right], \tag{4}$$

where λ controls the tradeoff between accuracy and retrieval cost.

This formulation highlights several useful interpretations. Uniform retrieval corresponds to the special case $\lambda = 0$, where all stores are retrieved regardless of cost. Oracle routing approximates the optimal solution when the relevant stores for each query are known. Heuristic routing policies attempt to approximate π^* using semantic signals extracted from the query.

Importantly, store routing differs from retriever routing. Retriever routing selects which search system or index to query, typically assuming a homogeneous document collection. Store routing operates at the memory-architecture level, where each store contains information with a distinct semantic role (e.g., short-term context, persistent user facts, or historical sessions). The routing decision therefore determines not only which documents are retrieved but also the effective signal-to-noise ratio of the context presented to the language model.

This perspective also clarifies the empirical findings in our evaluation. When irrelevant stores are retrieved, the effective retrieval cost increases while the probability of extracting the correct evidence can decrease due to contextual

[1] We also considered store-access latency. Results are similar; see Section 11.

noise. Conversely, selecting a smaller but semantically appropriate subset of stores improves both efficiency and answer reliability. The oracle–heuristic gap observed in Section 6 suggests that future systems should learn routing policies that optimize downstream QA objectives directly, rather than relying solely on rule-based heuristics.

4 METHOD

We compare policies that span a simple-to-strong spectrum. The key difference lies in what information the router uses and how conservative it is about missing required stores.

Uniform Baseline. Always retrieve from all stores. This guarantees perfect coverage but results in the highest cost and waste. Many deployed systems use this default behavior.

Oracle Upper Bound. Use ground-truth store labels. This achieves perfect coverage and EM and serves as a cost-efficient upper bound under our labeling protocol. The oracle is not deployable, but it quantifies the headroom available from store selection alone.

Fixed Subset Policies. Retrieve from a fixed subset such as STM+Sum+LTM. These policies are deployable and provide a strong baseline when adaptive routing is unavailable.

Hybrid Heuristic (baseline). We also test a simple deployable heuristic that combines semantic pattern matching with a conservative fallback. Algorithm 1 shows the core rule-based logic. In practice, we also use query-store embedding similarity as a tiebreaker when no pattern fires, which contributes an additional 4% coverage (see ablation in Section 6). We present the hybrid as a baseline rather than a final router, because its downstream QA performance leaves substantial room for learned routing.

Algorithm 1 Hybrid Store Routing (core rules; embedding similarity used as tiebreaker when no pattern matches)

Input: Query q
Extract semantic signals from q
if quantity signal ("list all", "every") **then**
    return {LTM, Epi} {Exhaustive recall}
else if temporal signal ("before", "changed") **then**
    return {LTM, Epi} {Historical comparison}
else if multi-hop signal ("compare", "relate") **then**
    return {Sum, LTM} {Cross-reference}
else if current session ("just said", "today") **then**
    return {STM} {Recent context}
else if fact lookup ("what is my", "who is my") **then**
    return {Sum} {User profile}
else
    return {Sum, LTM} {Safe fallback}
end if

Fallback Choice. We tested all six two-store combinations. Sum+LTM yields the highest coverage (89%), making it the safest default.

Design Rationale. We optimize first for coverage, since missing a required store makes the question effectively unanswerable. When semantic signals are clear, we route narrowly; when they are ambiguous, we fall back to the combination that recovers the most cases.

5 EXPERIMENTS

5.1 SYNTHETIC ROUTING EVALUATION

Before testing routing policies on LLM-based question answering, we first verify whether the policies select the appropriate memory stores under controlled conditions using synthetic labels. This preliminary step allows us to isolate the routing decision itself, independent of retrieval quality or model reasoning, and ensures that downstream performance differences can be interpreted more clearly.

Store-Labeling Protocol. We derive ground-truth store labels from the query taxonomies used in LoCoMo and LongMemEval. Each query type is associated with the memory stores that contain the information required to answer

it. For example, single-hop fact queries typically depend on the Summary store, while temporal or comparison queries often require information from both Long-Term Memory and Episodic Memory. The complete mapping of query categories to store requirements is provided in Section 9.

These labels are generated automatically using semantic rules derived from the benchmark taxonomies rather than manual annotation. As a result, the labeling process remains scalable and reproducible across datasets. Because the labels are based on query semantics rather than observed retrieval outcomes, they provide a consistent reference for evaluating routing decisions even when retrieval pipelines or underlying models change.

Dataset. 1,000 synthetic queries across 7 types, 70/30 train/test split.

Table 1: Synthetic routing evaluation. Metrics measure store selection quality (not QA accuracy).

Policy	Coverage	Exact Match	Waste
Uniform	100%	8%	2.9
Rule-based (linguistic only)	57%	35%	0.5
Hybrid (ours)	**94%**	**58%**	1.2
Oracle	100%	100%	0.0

Findings. The uniform policy achieves perfect coverage because it always retrieves from every store. However, its exact match rate is only 8%, reflecting substantial over-retrieval. Since all stores are included regardless of query requirements, the policy rarely selects the minimal set of stores needed to answer a question.

The linguistic rule-based baseline performs better in terms of precision but suffers from limited coverage, reaching only 57%. Its performance declines on query types that lack explicit surface cues, such as temporal comparisons or multi-hop reasoning tasks where the required stores cannot be inferred from simple keyword patterns.

The hybrid heuristic improves coverage substantially, reaching 94%, while maintaining a moderate exact match rate of 58%. This gain is primarily driven by combining semantic pattern detection with a conservative fallback strategy. When the heuristic detects clear signals, it routes narrowly; when signals are weak or ambiguous, it selects a broader but still constrained set of stores, which helps recover otherwise missed cases.

5.2 LLM Evaluation

Setup. We evaluate 12 store-selection policies on a dataset of 150 questions using GPT-3.5-turbo and GPT-4o-mini. The questions span seven query types and are tested under two context regimes. The *short* condition includes 100 questions with approximately 200 tokens retrieved per store, while the *long* condition includes 50 questions with approximately 1000 tokens retrieved per store. This design allows us to study both moderate-context and high-context retrieval settings.

All evaluations use temperature 0 to reduce generation variance and substring-based answer matching for scoring.[2]

Table 2: LLM evaluation (150 questions). Accuracy = QA correctness. Tokens = context size.

Model	Policy	Overall	Short	Long	Tokens
GPT-3.5	oracle	85.3%	93%	70%	299
	stm+sum+ltm	85.3%	92%	72%	591
	uniform	83.3%	91%	68%	787
	hybrid	69.3%	79%	50%	379
GPT-4o-mini	oracle	86.7%	94%	72%	299
	stm+sum+ltm	84.7%	92%	70%	591
	uniform	81.3%	92%	60%	787
	hybrid	70.7%	80%	52%	379

[2]Accuracy differences greater than 4% are statistically significant at $p < 0.05$ via bootstrap resampling (1,000 iterations).

Finding 1: Store selection can improve accuracy and reduce tokens. Oracle routing outperforms uniform retrieval on both efficiency and answer quality. It achieves higher accuracy (86.7% vs 81.3%) while using 62% fewer context tokens (299 vs 787). This result highlights that providing more context does not necessarily improve performance. When the router selects only the stores that are likely to contain the answer, the model receives a smaller but cleaner context, which can lead to more reliable extraction.

Finding 2: Long context amplifies the penalty of over-retrieval. The difference between routing policies becomes more pronounced in the long-context setting. On long-context questions, oracle routing reaches 72% accuracy compared with 60% for uniform retrieval. When each store contributes roughly 1000 tokens, adding irrelevant stores significantly increases the amount of distracting text, making it harder for the model to identify the correct information.

Finding 3: A strong fixed policy is competitive. A fixed routing policy such as STM+Sum+LTM approaches oracle-level accuracy while maintaining moderate cost. This suggests that even simple routing strategies can capture much of the benefit of selective retrieval when the memory architecture is well structured. Episodic retrieval, in contrast, is required only in a small subset of cases and can sometimes degrade performance by introducing unnecessary context.

Finding 4: Heuristic routing is not yet sufficient. Although the hybrid heuristic achieves strong coverage on the synthetic routing benchmark, it performs less well on downstream QA tasks. This gap indicates that selecting the correct stores in principle does not always translate into better end-to-end performance. Downstream accuracy depends not only on store selection but also on the model's ability to retrieve and use the relevant evidence within the provided context.

5.3 Why Does Uniform Underperform?

Uniform retrieval provides more information than oracle routing, yet it consistently yields lower accuracy. Two factors help explain this behavior.

Needle in a haystack. With approximately 787 tokens in the prompt, the model must locate a small set of relevant facts embedded within a larger amount of unrelated text. This difficulty becomes more pronounced in long-context queries, where each additional store can contribute roughly 1000 tokens, further diluting the signal.

Conflicting information. Different stores may contain inconsistent or outdated facts. For example, long-term memory may preserve earlier information that conflicts with updated user summaries. When all stores are retrieved together, the model must decide which source to trust, and it may occasionally select the wrong one.

Example. Consider the query "Who is my current manager?" The Summary store contains the entry "Manager: Jennifer Williams." The LTM store contains a previous session note: "Before the reorg, user reported to Michael Torres. Now reports to Jennifer Williams." Under uniform retrieval, both passages appear in context, and GPT-4o-mini sometimes extracts "Michael Torres" from the more detailed historical passage. Under oracle routing, which retrieves only the Summary store, the model consistently returns "Jennifer Williams." This example illustrates how additional context can sometimes mislead the model rather than improve performance.

5.4 Understanding the Coverage-Accuracy Gap

The hybrid heuristic achieves 94% routing coverage (Section 5.1) but only 70% QA accuracy. Several factors contribute to this difference.

Coverage is not accuracy. Coverage measures whether the router included the stores that contain the required information. QA accuracy measures whether the model successfully identifies and uses that information in the retrieved context. Even when the correct store is present, extraction can fail if the context is long or contains distracting material.

Missing stores remain decisive. When the hybrid heuristic fails to select a required store, which occurs in 6% of queries, the question effectively becomes unanswerable under the retrieval setup. These cases directly contribute to the overall error rate.

Over-retrieval can also hurt. The hybrid policy achieves 58% exact match, meaning that 42% of queries retrieve additional stores beyond those required. The resulting extra context can distract the model and reduce answer accuracy even when the correct stores are included.

Overall, the coverage-accuracy gap reflects two distinct failure modes. The first is routing error, where the heuristic misses a necessary store (12% of QA errors). The second is extraction error, where the model fails to locate or correctly

use the answer despite correct store selection (18% of QA errors), often because over-retrieval introduces additional contextual noise.

6 ANALYSIS

Our results highlight two distinct phenomena. First, selecting the appropriate stores before retrieval can simultaneously reduce context tokens and improve downstream QA accuracy. Second, simple heuristic routing, although effective, does not fully close the performance gap relative to oracle routing. To better understand which signals drive routing quality, we analyze feature contributions and computational overhead.

6.1 FEATURE ABLATION

We evaluate different feature groups on the synthetic routing benchmark to understand which signals are most useful for identifying the correct stores.

Table 3: Feature ablation: routing coverage by feature type.

Features	Coverage	Δ
Linguistic (pronouns, tense)	57%	baseline
+ Semantic (quantity, temporal, multi-hop)	90%	+33%
+ Embedding similarity	94%	+4%

Semantic signals dominate. Adding semantic indicators such as quantity ("list all"), temporal references ("before"), and multi-hop reasoning cues ("compare") increases coverage by 33 percentage points. These signals capture query patterns that simple linguistic features, such as pronouns or tense, often fail to identify. As a result, routing policies that rely only on shallow linguistic patterns tend to miss a substantial fraction of queries requiring historical or multi-store reasoning.

Embedding similarity provides an additional 4 percentage point improvement. Its main benefit appears on queries that do not contain clear surface cues but still have a semantic relationship to specific stores. In these cases, similarity scores help guide the router toward likely stores even when explicit rule-based triggers are absent.

6.2 COMPUTATIONAL OVERHEAD

Routing introduces minimal additional latency. Rule-based routing policies add less than 1 ms of processing time, while embedding-based signals add approximately 5 ms. Both overheads are small relative to typical LLM inference times, which range from 500 to 2000 ms.

Because routing reduces the amount of retrieved context, the resulting 62% token reduction directly translates into lower inference cost. In practice, this reduction also decreases prompt processing time and can improve system responsiveness without requiring any modification to the underlying language model.

7 LIMITATIONS

Synthetic labels. Ground-truth store labels are derived from query taxonomies rather than human annotation. This protocol allows controlled and reproducible evaluation of routing behavior, but it does not fully capture the variability present in real-world deployments. In practice, the necessity of a store may depend on how information is written, summarized, or updated over time. Human validation of store requirements would therefore provide a stronger assessment of routing accuracy in production settings.

Heuristic router gap. The hybrid heuristic achieves high coverage on the synthetic routing benchmark but underperforms on downstream QA compared with oracle routing and strong fixed policies. This difference suggests that routing decisions interact closely with within-store retrieval quality and answer extraction. Closing the gap will likely require learned routing approaches that jointly optimize store selection and evidence retrieval rather than relying solely on rule-based heuristics.

Two model families. Our evaluation includes GPT-3.5-turbo and GPT-4o-mini. While these models represent commonly used systems, testing additional architectures and training paradigms would strengthen claims about generalization. In particular, models with different context handling strategies or retrieval sensitivities may respond differently to routing policies.

Full store retrieval. We concatenate full store contents instead of performing top-k retrieval within each store. This design simplifies the analysis by isolating store-selection effects, but production systems often apply ranking or filtering within each memory store. The interaction between routing decisions and within-store retrieval strategies remains an important direction for future study.

8 Conclusion

We formalized memory store selection as a routing problem and evaluated it in two stages: first using synthetic labels to validate store-selection behavior, and then using real LLMs to measure downstream question answering performance. This two-stage evaluation separates the quality of routing decisions from the effects of retrieval and generation, allowing us to analyze the role of store selection more directly.

Our results show several consistent patterns. Selective store choice can improve both efficiency and accuracy, reducing context tokens by 62% while increasing QA accuracy (86% vs 81%). Uniform retrieval, in contrast, can introduce unnecessary context that reduces performance, particularly in long-context settings where additional stores contribute large amounts of irrelevant text. Semantic signals substantially improve routing quality, increasing coverage by 33 percentage points compared with linguistic features alone. A simple fixed policy such as STM+Sum+LTM also provides a strong and deployable default when adaptive routing is not available.

We additionally introduced a decision-theoretic formulation of store routing that treats memory access as a cost-sensitive optimization problem. This framework explains the observed accuracy–efficiency tradeoffs and clarifies why routing decisions can influence answer quality even when retrieval and language models remain unchanged.

The remaining 16-point gap between heuristic routing (70%) and oracle routing (86%) indicates that further gains are possible. Closing this gap will likely require routing policies that are learned end-to-end and optimized directly for downstream QA outcomes rather than relying solely on hand-designed heuristics. We hope this work encourages further research on learned routing methods and highlights the importance of store selection as a core component of memory-augmented agent systems.

9 Query Type to Store Mapping

To evaluate routing behavior under controlled conditions, we assign each query type a set of stores that contain the information required to answer it. The mapping reflects how information is distributed across the memory architecture rather than how any particular retrieval system performs. For example, factual user attributes are stored in the Summary store, while historical comparisons typically require information from both Long-Term Memory and Episodic Memory.

The goal of this mapping is to provide a consistent reference for measuring routing quality across policies. Because the mapping is derived from benchmark query taxonomies, it remains reproducible across datasets and does not depend on model outputs or manual labeling decisions. Table 9 lists the resulting query-type to store assignments used throughout our synthetic routing evaluation.

Table 4: Ground-truth store labels by query type.

Query Type	Stores	Rationale
single_hop	Sum	Simple fact lookup
single_session	STM	References current conversation
recent_session	LTM	References past conversations
multi_hop	Sum + LTM	Combines facts with history
memory_capacity	LTM + Epi	Exhaustive recall (“list all”)
temporal	LTM + Epi	Historical comparison (“before X”)
knowledge_update	Sum + LTM	Current vs historical state

10 Full Policy Comparison

Table 5: All 12 policies (GPT-4o-mini, 150 questions).

Policy	Accuracy	Short	Long	Tokens
oracle	86.7%	94%	72%	299
stm+sum+ltm	84.7%	92%	70%	591
uniform	81.3%	92%	60%	787
summary+ltm	74.0%	82%	58%	406
hybrid	70.7%	80%	52%	379
ltm	49.3%	58%	32%	212
ltm+episodic	49.3%	57%	34%	408
stm+summary	45.3%	48%	40%	379
summary	30.7%	36%	20%	195
episodic	30.7%	43%	6%	196
stm	14.7%	14%	16%	184
none	14.0%	2%	38%	0

11 Store-Access Cost Analysis

In addition to token cost, we examined the relative infrastructure cost associated with accessing different memory stores. In practical deployments, stores may reside on different storage tiers or require different retrieval pipelines, which can lead to varying access latency and compute overhead. To approximate these differences, we assign relative store-access costs: STM=1, Summary=1, LTM=3, and Episodic=5. These values represent normalized relative effort rather than exact system measurements.

Under this model, uniform retrieval incurs the highest cost (10), since it accesses all stores for every query. Oracle routing averages 3.9, reflecting its ability to select only the stores required for a given question. The hybrid heuristic reduces the average store-access cost by 29% while maintaining 94% coverage, demonstrating that routing policies can meaningfully reduce system overhead even when implemented with simple decision rules.

Although the main paper focuses on token cost because it directly influences LLM inference expense, the store-access analysis shows that similar efficiency gains appear at the infrastructure level. As routing policies become more adaptive, reductions in store-access overhead may translate into lower latency and improved scalability in production memory systems.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection, 2023. URL https://arxiv.org/abs/2310.11511.

Jamie Callan. Distributed information retrieval. In W. Bruce Croft (ed.), *Advances in Information Retrieval: Recent Research from the Center for Intelligent Information Retrieval*, pp. 127–150. Springer, 2002.

Harrison Chase. Langchain: Building applications with llms through composability, 2023. https://www.langchain.com.

Wang Chen, Guanqiang Qi, Weikang Li, Yang Li, Deguo Xia, and Jizhou Huang. Decide then retrieve: A training-free framework with uncertainty-guided triggering and dual-path retrieval, 2026. URL https://arxiv.org/abs/2601.03908.

Jingsheng Gao, Linxu Li, Weiyuan Li, Yuzhuo Fu, and Bin Dai. Smartrag: Jointly learn rag-related tasks from the environment feedback, 2025. URL https://arxiv.org/abs/2410.18141.

Esmail Gumaan. Expertrag: Efficient rag with mixture of experts – optimizing context retrieval for adaptive llm responses, 2025. URL https://arxiv.org/abs/2504.08744.

Yucan Guo, Miao Su, Saiping Guan, Zihao Sun, Xiaolong Jin, Jiafeng Guo, and Xueqi Cheng. Routerag: Efficient retrieval-augmented generation from text and graph via reinforcement learning, 2025. URL https://arxiv.org/abs/2512.09487.

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, 2024. URL https://arxiv.org/abs/2403.14403.

Xu Ji, Luo Xu, Landi Gu, Junjie Ma, Zichao Zhang, and Wei Jiang. Rap-rag: A retrieval-augmented generation framework with adaptive retrieval task planning. *Electronics*, 14(21):4269, 2025.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7969–7992, 2023.

To Eun Kim and Fernando Diaz. Ltrr: Learning to rank retrievers for llms, 2025. URL https://arxiv.org/abs/2506.13743.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

Zhuowan Li, Cheng Li, Mingyang Zhang, Qiaozhu Mei, and Michael Bendersky. Retrieval augmented generation or long-context llms? a comprehensive study and hybrid approach, 2024a. URL https://arxiv.org/abs/2407.16833.

Zixuan Li, Jing Xiong, Fanghua Ye, Chuanyang Zheng, Xun Wu, Jianqiao Lu, Zhongwei Wan, Xiaodan Liang, Chengming Li, Zhenan Sun, Lingpeng Kong, and Ngai Wong. Uncertaintyrag: Span-level uncertainty enhanced long-context modeling for retrieval-augmented generation, 2024b. URL https://arxiv.org/abs/2410.02719.

Jerry Liu et al. Llamaindex: A data framework for llm applications, 2023a. https://www.llamaindex.ai.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts, 2023b. URL https://arxiv.org/abs/2307.03172.

Jie Lu and Jamie Callan. Full-text federated search of text-based digital libraries in peer-to-peer networks. *Information Retrieval*, 9(4):477–498, 2006.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents. *arXiv preprint arXiv:2402.17753*, 2024.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems, 2024. URL https://arxiv.org/abs/2310.08560.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pp. 1–22, 2023.

Milad Shokouhi, Luo Si, et al. Federated search. *Foundations and trends® in information retrieval*, 5(1):1–102, 2011.

Luo Si and Jamie Callan. Relevant document distribution estimation method for resource selection. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pp. 298–305, 2003.

Endel Tulving et al. Episodic and semantic memory. *Organization of memory*, 1(381-403):1, 1972.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory, 2025. URL https://arxiv.org/abs/2410.10813.

Di Wu, Jia-Chen Gu, Kai-Wei Chang, and Nanyun Peng. Self-routing rag: Binding selective retrieval with knowledge verbalization, 2026. URL https://arxiv.org/abs/2504.01018.

Yue Yu, Wei Ping, Zihan Liu, Boxin Wang, Jiaxuan You, Chao Zhang, Mohammad Shoeybi, and Bryan Catanzaro. Rankrag: Unifying context ranking with retrieval-augmented generation in llms. *Advances in Neural Information Processing Systems*, 37:121156–121184, 2024.

Guibin Zhang, Muxin Fu, Guancheng Wan, Miao Yu, Kun Wang, and Shuicheng Yan. G-memory: Tracing hierarchical memory for multi-agent systems. *arXiv preprint arXiv:2506.07398*, 2025.

Qianchi Zhang, Hainan Zhang, Liang Pang, Yongxin Tong, Hongwei Zheng, and Zhiming Zheng. Less is more: Compact clue selection for efficient retrieval-augmented generation reasoning, 2026. URL https://arxiv.org/abs/2502.11811.

A Memory Content Examples

To illustrate how information is distributed across stores, we present representative examples drawn from the synthetic memory construction used in our experiments. These examples demonstrate the semantic roles of each store rather than the exact evaluation instances.

Short-Term Memory (STM). STM contains recent conversational context, such as scheduling or clarification requests:

> "User just mentioned they have a meeting at 3pm today with the marketing team."
> "User asked to schedule a follow-up call for tomorrow morning."

Queries such as "What did I just say about today's meeting?" require STM.

Summary Store. The Summary store contains stable user facts and preferences:

> "User Profile: Name is Alex Chen. Works at TechCorp as Senior Software Engineer."
> "Contact: Phone number is 555-867-5309. Email is alex.chen@techcorp.com."
> "Manager: Jennifer Williams."

Fact lookup queries such as "Who is my manager?" primarily depend on this store.

Long-Term Memory (LTM). LTM stores summaries of past conversations:

> "Session 9 (yesterday): Before the recent reorg, user's manager was Michael Torres. Now reports to Jennifer Williams."
> "Session 10 (last week): User mentioned weight was 185 lbs before starting a diet."

Queries involving historical comparisons or past discussions typically require LTM.

Episodic Memory. Episodic memory preserves raw conversation turns:

> "Session 10, Turn 2: User said 'Back in January I was 185 pounds. With the diet I started, I'm hoping to get down to 170 by summer.' "

Queries requiring exact wording or timestamped references may depend on episodic retrieval.

B Code Availability

All code used to generate synthetic routing labels, run routing-policy evaluations, and reproduce the LLM-based QA experiments is available in following repository:

`https://github.com/krimler/memroute`

The repository includes scripts for synthetic memory generation, routing-policy evaluation, ablation studies, and end-to-end QA benchmarking. Instructions for reproducing the experimental results are provided in the project README. All experiments can be executed using the configuration files included in the repository.