# Elevating Agent Decision-Making via Dynamic Graph-Based Strategic Memory

**Anonymous ACL submission**


## Abstract

Large Language Model (LLM)-based agents have shown strong potential in knowledge-intensive and tool-augmented reasoning tasks, where effective reuse of prior decision experiences is crucial for improving agent behavior. However, existing memory methods either encode experience implicitly in model parameters, making it difficult to inspect and selectively reuse, or retrieve explicit memories mainly by semantic similarity, which may select noisy or low-utility experiences. We propose a hierarchical memory graph that enables agents to reuse past experience in a structured and utility-aware manner. Instead of treating past trajectories as isolated demonstrations, our framework maps them into canonical decision paths, making recurring success and failure patterns easier to compare. These patterns are distilled into human-interpretable meta-cognitive strategies and calibrated with downstream reward feedback. For new tasks, the graph retrieves strategies that are both relevant and empirically useful, and injects them as plug-and-play guidance during inference and reinforcement learning. Experiments on single-hop and multi-hop QA benchmarks show notable improvements, with Qwen3-4B achieving a **14.6% relative improvement** in training-free inference.


## 1 Introduction

LLM agents have shown strong potential in knowledge-intensive and tool-augmented reasoning tasks, such as open-domain and multi-hop question answering, where they must decompose queries, invoke external tools, and integrate evidence across multiple steps (Yao et al., 2023; Schick et al., 2023; Chai et al., 2025). To improve beyond isolated problem solving, agents need to accumulate, organize, and reuse prior experience across tasks, a goal that existing methods typically pursue through two memory paradigms. The first is **implicit memory**, typically formed through training procedures such as Reinforcement Learning (RL), where prior experience is encoded into model parameters (Li et al., 2025c; Bai et al., 2022). A more flexible alternative is **explicit memory**, which leverages contextual prompting to inject retrieved experiences or insights directly into the input without modifying model weights (Xu et al., 2025; Chhikara et al., 2025; Zhao et al., 2024).

In tool-augmented agent settings, such experience reuse goes beyond factual recall: memory should also support tool selection, evidence verification, and failure avoidance. However, reliable experience reuse remains difficult under existing memory paradigms. Implicit memory can improve agent behavior through training, but the learned knowledge is hard to inspect, attribute, or selectively reuse. Explicit memory preserves past experiences, but existing methods often represent them as raw trajectories or flat insights, with limited abstraction of decision processes. Retrieval is typically based on semantic similarity rather than downstream utility, making it difficult to select memories that truly help the current task. This gap motivates our central research question: *Can explicit agent memory be organized into interpretable decision-level strategies, and can reward feedback help select strategies that are useful for future reasoning and policy learning?*

To address this gap, we propose a structured and utility-aware memory framework for tool-augmented LLM agents. Rather than storing past experiences as raw trajectories or flat insights, our framework abstracts them into reusable strategic knowledge and learns how useful each strategy is for future tasks. Specifically, we first map agent trajectories into canonical decision paths and distill them into high-level *meta-cognitive strategies*. We then organize these strategies in a trainable graph memory, where edge weights are updated using downstream reward signals. The optimized graph retrieves task-relevant strategies as explicit strate-

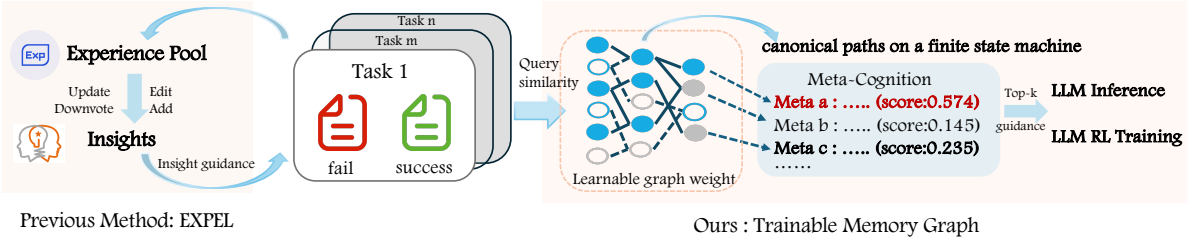


Figure 1: Overview of our framework compared to existing approaches EXPEL (Zhao et al., 2024).

gic priors, which are injected into the agent's context during inference and RL training. Experiments on seven QA benchmarks demonstrate improvements over tool-use, raw-experience, and abstract-memory baselines.

Our main contributions are threefold:

- We propose a hierarchical memory graph that organizes agent trajectories into queries, canonical decision paths, and meta-cognitive strategies, providing a structured alternative to raw trajectories or flat insights.
- We introduce reward-calibrated strategy selection, using downstream reward differences to update graph weights and retrieve empirically useful strategies beyond semantic similarity.
- We instantiate the resulting memory as a plug-and-play strategic prior during both inference and RL training. Experiments across seven QA benchmarks show that structured strategic memory improves average performance over tool-use and static-memory baselines.

Overall, this work presents a unified framework for enabling LLM agents to learn from experiences and reuse them for adaptive strategic reasoning.

## 2 Related Work

### 2.1 Tool-Augmented Agentic Policy Learning

LLM agents increasingly use external tools to extend their reasoning and problem-solving capabilities. Prompt-based approaches such as ReAct (Yao et al., 2023) and WebGPT (Nakano et al., 2022) first showed that agents can interleave reasoning with tool actions, while Search-o1 (Li et al., 2025b) further enables autonomous search invocation during multi-step problem solving. Recent work improves tool-augmented agents through RL-based training (Sun et al., 2025; Zheng et al., 2025; Song et al., 2025), including Search-R1 (Jin et al., 2025), which trains models to alternate between reasoning and retrieval. Other studies explore reward design (Wang et al., 2025; Qian et al., 2025), strategic tool use (Feng et al., 2025), and general RL infrastructure for multi-turn tool learning (Chai et al., 2025). However, RL for tool-augmented agents still faces large and weakly guided exploration spaces (Dong et al., 2026). Recent approaches therefore inject external plans or experiential priors into the input context to guide policy optimization (Lu et al., 2026a; Wang et al., 2026). Our work follows this plug-and-play direction, but retrieves high-utility meta-cognitive strategies from an offline-trained memory graph and prepends them to prompts, providing explicit strategic priors without modifying the underlying Group Relative Policy Optimization (GRPO) algorithm.

### 2.2 Structured Memory and Strategic Learning for LLM Agents

Recent studies have explored how LLM agents can reuse prior experience for long-horizon reasoning and decision-making. Reflective and experiential methods such as Reflexion (Shinn et al., 2023) and EXPEL (Zhao et al., 2024) transform past interactions into verbal feedback or reusable rules. Later systems extend this idea toward persistent memory storage, dynamic updating, and practical memory management, including MEM1 (Zhou et al., 2025), MemAgent (Yu et al., 2025), A-MEM (Xu et al., 2025), Mem0 (Chhikara et al., 2025), and MemoryOS (Kang et al., 2025). To capture richer reasoning dependencies, graph-based memory further organizes experiences through logic-aware, hierarchical, constructivist, multi-graph, or adaptive structures (Liu et al., 2025; Zhang et al., 2025; Li et al., 2025a; Jiang et al., 2026; Lu et al., 2026b; Anonymous, 2026), and recent surveys identify graph memory as a promising paradigm for relational and hierarchical agent experience modeling (Yang et al., 2026). However, most existing methods still rely

on textual memories, static structures, or heuristic retrieval, and rarely learn which memory components are empirically useful for downstream tasks.

Another line of work integrates memory with RL to improve agent adaptation. Prior studies analyze how experiences and memory update strategies affect agent behavior (Pan and Zhao, 2025; Xiong et al., 2025), while recent methods make memory an active part of policy learning. For example, AgeMem (Yu et al., 2026) learns memory operation policies, MIRA (Nourzad and Joe-Wong, 2026) uses structured memory to guide RL, and EMPO[2] (Liu et al., 2026) leverages memory for exploration under hybrid policy optimization.

Prior reflective memory methods often store flat textual insights, whereas our method separates concrete queries, canonical decision paths, and abstract strategies. Compared with graph memory methods based on semantic or heuristic retrieval and RL-only tool-use methods without explicit strategic memory, our framework learns graph weights from downstream reward signals and injects selected strategies as interpretable priors without modifying the underlying RL algorithm.

# 3 Preliminaries

## 3.1 Heterogeneous Graph Structure

Graphs provide a natural formalism for modeling structured dependencies among diverse entities. A heterogeneous graph (Zhang et al., 2019) can be defined as

$$\mathcal{G} = (V, E, \mathcal{O}_V, \mathcal{R}_E, C), \tag{1}$$

where $V$ denotes the set of nodes, $E \subseteq V \times V$ denotes the set of directed edges, $\mathcal{O}_V$ denotes the set of node types, $\mathcal{R}_E$ denotes the set of relation types, and $C$ is the collection of node contents. Each edge $e = (u, v, r) \in E$ specifies a relation of type $r$ from node $u$ to node $v$.

Connectivity in $\mathcal{G}$ is represented by node-type adjacency matrices

$$A^{xy} \in \{0,1\}^{|V_x| \times |V_y|}, \quad (x, y) \in \mathcal{O}_V \times \mathcal{O}_V, \tag{2}$$

where $\mathcal{V}_x$ and $\mathcal{V}_y$ denote the sets of nodes of type $x$ and $y$, respectively. An entry $(A^{xy})_{ij} = 1$ indicates that node $i$ of type $x$ is connected to node $j$ of type $y$. This formulation emphasizes the structural dependencies across different node types.

To enable learning, each $A^{xy}$ is coupled with a weight matrix $W^{xy}$, so that propagation is governed by the weighted operator $A^{xy} \odot W^{xy}$. Thus, structure defines feasible paths, while weights determine effective information flow. Formally,

$$\mathbf{H}_y = \sigma\Big( (A^{xy} \odot W^{xy})^\top \mathbf{H}_x \Big), \tag{3}$$

where $H^x \in R^{|V^x| \times 1}$ is the activation score vector for layer $x$ and $\sigma(\cdot)$ denotes an activation function.

## 3.2 Tool-Augmented LLM Agents

The interaction between an LLM and external tools can be formalized as a structured multi-turn decision process (Chai et al., 2025). At each time step $t$, the agent observes $s_t = (q, h_{1:t-1}), \quad a_t \sim \pi_\theta(a_t \mid s_t)$. where $q$ is the user query and $h_{1:t-1}$ the dialogue or reasoning history, then generates an action $a$ denoting internal reasoning, tool invocation, or answer generation, using a protocol with tags such as `<think>`, `<tool_call>`, and `<answer>`.

The process continues until either the tag `<answer>` is generated or the agent has issued up to a maximum of $K$ tool invocations. A trajectory $\tau = (s_1, a_1, o_1, \ldots, s_T, a_T, o_T)$ yields reward $R(\tau)$ , where $o_t$ denotes the environment's response (e.g., tool outputs if $a_t$ is a tool call), and the policy is optimized via $J(\theta) = E_{\tau \sim \pi_\theta}[R(\tau)]$,

$$\nabla_\theta J(\theta) \approx E_\tau \Big[ \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\, \hat{A}_t \Big]. \tag{4}$$

# 4 Method

As illustrated in Figure 2, our method consists of three phases. First, we construct an Agent-Centric Memory Graph (Sec. 4.1) that organizes trajectories into structured paths and meta-cognitive strategies. Second, we train graph weights (Sec. 4.2) to estimate memory utility and refine retrieval. Finally, we perform Memory-Guided RL (Sec. 4.3), where retrieved strategies are injected as context for policy optimization.

## 4.1 Hierarchical Memory Graph Construction

**Memory Graph Structure.** We instantiate a heterogeneous graph $\mathcal{G} = (V, E)$ with a three-tier reasoning hierarchy consisting of three node types: queries, transition paths, and meta-cognitions:

- **The Query Layer** $\mathcal{Q}$: Each query node $q_i$ stores the input, sampled trajectories, and outcome labels. Since a single query may yield multiple rollouts, $q_i$ can connect to several transition paths via edges $(q_i \to t_j)$.

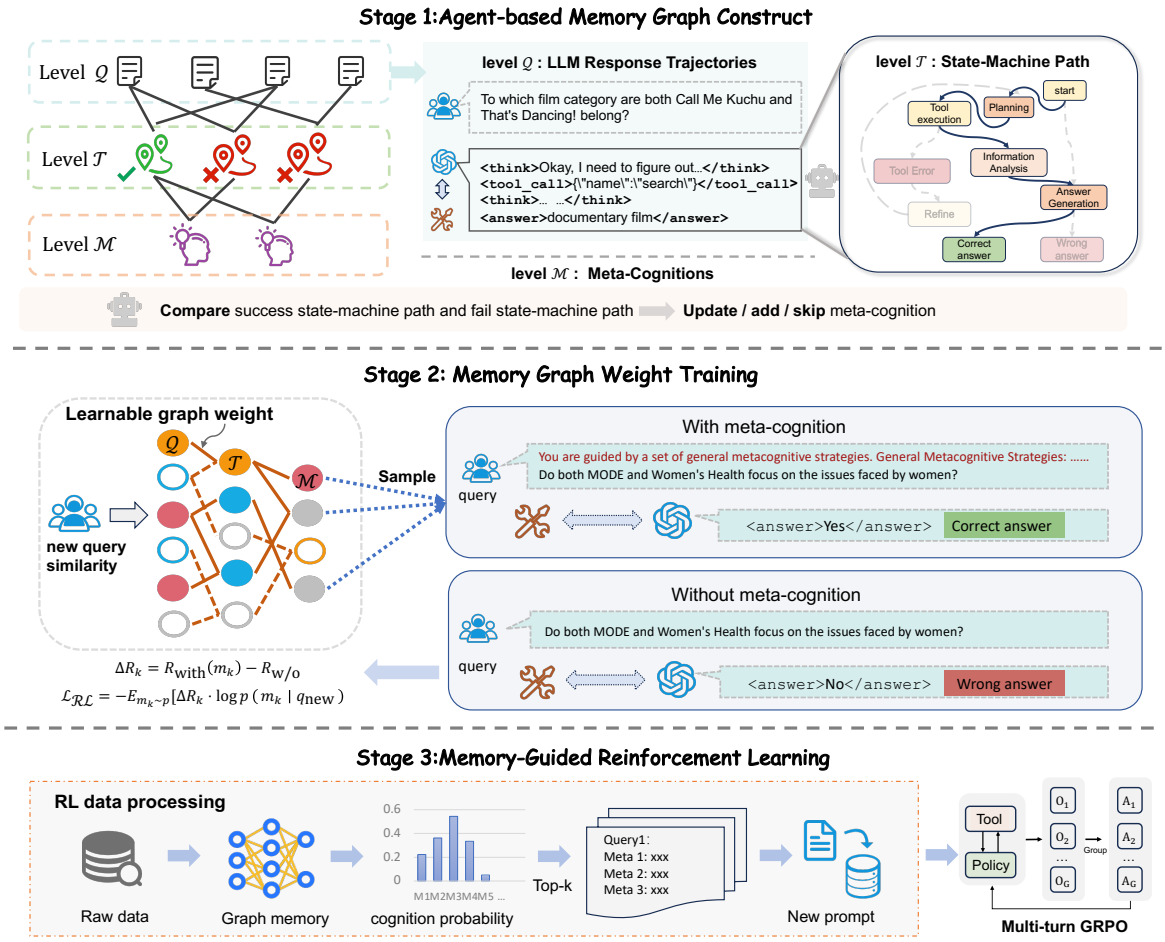


Figure 2: The framework of the trainable memory graph. Stage 1 builds a graph from LLM trajectories, encoding queries, decision paths, and meta-cognition. Stage 2 estimates strategy utility via differential rewards and updates graph weights. Stage 3 injects top-k strategies into RL training for policy optimization.

- **The Transition Path Layer $\mathcal{T}$**: Each path node $t_j$ encodes a canonical trajectory obtained by mapping reasoning traces into Finite State Machine (FSM) paths. This abstraction filters linguistic noise and exposes the decision structure shared across tasks.

- **The Meta-Cognition Layer $\mathcal{M}$**: Each meta-cognition node $m_k$ captures a reusable strategic principle distilled from transition paths. These principles characterize why reasoning patterns succeed or fail and serve as domain-agnostic behavioral heuristics.

The complete construction process is illustrated in Figure 5. Formally, graph connectivity is encoded by two bipartite adjacency matrices, augmented with learnable weights $w_{qt}$ and $w_{tm}$, respectively. This directed hierarchy links concrete task experiences to increasingly abstract strategic knowledge, providing structured candidates for utility-aware retrieval.

$$A^{q\to t} \in \{0,1\}^{|\mathcal{Q}|\times|\mathcal{T}|}, \quad A^{t\to m} \in \{0,1\}^{|\mathcal{T}|\times|\mathcal{M}|}. \tag{5}$$

**Finite State Machine Construction.** Reasoning trajectories are often noisy, redundant, and stylistically diverse. We construct a FSM $\mathcal{S}$ to normalize them into canonical decision paths $t_j$ composed of standardized cognitive primitives. The FSM state space is built in a data-driven and LLM-assisted manner: a teacher LLM inspects example trajectories, identifies recurring decision behaviors, and proposes candidate states, which are then consolidated into a compact schema with states such as `StrategyPlanning` and `InternalConflict`. We keep this FSM fixed across the main evaluation datasets, using it as a shared abstraction rather than learning a new state space for each task. This representation filters surface-level execution noise while preserving macro-level decision dependencies, allowing trajectories from different tasks to be com-

pared through shared cognitive transitions.

**Meta-Cognition Induction and Update.** For each query, we sample $N$ trajectories. When both successful and failed trajectories are available, an LLM compares their FSM paths to identify the key divergence and summarize it into a candidate meta-cognitive strategy. When only failed trajectories are observed, we retrieve semantically similar queries using $\cos(\mathbf{e}_{q_i}, \mathbf{e}_{q_j})$, where $\mathbf{e}_q$ denotes the query embedding, and reuse validated strategies from their successful paths.

To maintain graph sparsity, we employ an LLM-driven consolidation mechanism with a maximum of 30 unique meta-cognition nodes. Before the threshold is reached, the LLM judges whether a new insight should refine an existing node or instantiate a distinct one. Once the threshold is reached, the framework only updates existing node attributes rather than creating new nodes. This compresses redundant patterns into more robust and generalizable principles. Concrete cases and prompt templates are shown in Appendix H.3.

### 4.2 Reward-Calibrated Strategy Selection

The memory graph provides structured strategy candidates, but different meta-cognitions may have different utility for a new task. We therefore learn reward-calibrated graph weights to move beyond semantic-similarity retrieval and select strategies that are empirically useful for downstream tasks.

**Graph-Based Strategy Scoring.** We assign learnable weights $W_{qt}$ and $W_{tm}$ to the query–transition and transition–meta-cognition edges, respectively. Given a new query $q_{\text{new}}$, we initialize the query relevance vector by semantic similarity: $\mathbf{H}_{\mathcal{Q}}[i] = \text{Sim}(q_{\text{new}}, q_i)$. The relevance is then propagated through the weighted graph:

$$\begin{aligned} \mathbf{H}_{\mathcal{T}} &= (A_{qt} \odot W_{qt})^{\top} \mathbf{H}_{\mathcal{Q}}, \\ \mathbf{H}_{\mathcal{M}} &= (A_{tm} \odot W_{tm})^{\top} \mathbf{H}_{\mathcal{T}}. \end{aligned} \tag{6}$$

Here, $\mathbf{H}_{\mathcal{M}}$ gives the relevance scores of meta-cognition nodes. We keep the top-$k$ candidates and sample a meta-cognition $m_k$ from the softmax distribution over their scores. To estimate the utility of $m_k$, we compare two rollouts: one guided by $m_k$ and one without such guidance. The reward gap $\Delta R_k = R_{\text{with}}(m_k) - R_{\text{w/o}}$ serves as a differential utility signal, measuring whether the selected strategy improves task performance.

**Reward-Calibrated Weight Learning.** We optimize the graph weights with REINFORCE, treating the selected meta-cognition as an action and the reward gap as its feedback:

$$\mathcal{L}_{\text{graph}} = -E_{m_k \sim p}\left[\Delta R_k \log p(m_k \mid q_{\text{new}})\right]. \tag{7}$$

A positive $\Delta R_k$ increases the probability of selecting paths that lead to useful meta-cognitions, while a negative $\Delta R_k$ suppresses them. In this way, the graph gradually learns which strategies are empirically useful for downstream tasks.

### 4.3 Memory-Guided RL

Our framework incorporates the optimized memory graph into RL training. For each training instance, the graph retrieves relevant meta-cognitive strategies as high-level priors and injects them into the agent's context to guide policy optimization.

**Strategic Context Retrieval.** Given a training instance $q_{\text{train}}$, we use the optimized graph weights from Section 4.2 to score meta-cognition nodes and select the top-$k$ strategies $\{m_1, \ldots, m_k\}$. These scores reflect both semantic relevance and reward-calibrated utility. The selected strategies are prepended to the original query to form a memory-augmented prompt: $\tilde{q}_{\text{train}} = [\, m_1, m_2, \ldots, m_k \,;\, q_{\text{train}} \,]$.

**Optimization Objective.** The agent policy $\pi_\theta$ is trained to maximize expected reward conditioned on $\tilde{q}_{\text{train}}$. In practice, we adopt GRPO and the objective is:

$$\mathcal{L}_{\text{GRPO}} = -E_t\Big[\min\big(r_t(\theta)\hat{A}_t, \\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\big)\Big] \tag{8}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|\tilde{q}_{\text{train}})}{\pi_{\theta_{\text{old}}}(a_t|\tilde{q}_{\text{train}})}$. Here, $\hat{A}_t$ denotes the group-normalized advantage and $\epsilon$ is the clipping parameter. By conditioning policy optimization on retrieved strategies, the agent reuses useful experience instead of learning only from current rollouts.

## 5 Experiment

### 5.1 Datasets

To evaluate the effectiveness and generalizability of our approach, we conduct experiments on seven widely-used question answering datasets, covering both single-turn and multi-hop reasoning tasks. **(1) General QA Datasets**: We include Natural Questions (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and PopQA (Mallen et al., 2023), which consist of open-domain factoid questions requiring retrieval and basic reasoning capabilities. **(2) Multi-hop QA Datasets**: For

Table 1: Performance comparison across seven QA datasets in inference. †indicates in-domain datasets, while ⋆denotes out-of-domain datasets. Percentages in Avg. column denote relative improvement over TIR.

| Methods | Avg. (↑ / ↓ vs. TIR) | General QA | | | Multi-Hop QA | | | |
|---|---|---|---|---|---|---|---|---|
| | | NQ⋆ | TriviaQA⋆ | PopQA⋆ | HotpotQA† | 2wiki⋆ | Musique⋆ | Bamboogle⋆ |
| **Qwen3-8B** | | | | | | | | |
| TIR | 0.334 (–) | 0.275 | 0.593 | 0.358 | 0.325 | 0.324 | 0.094 | 0.365 |
| Direct Inference | 0.258 (↓22.7%) | 0.200 | 0.519 | 0.191 | 0.230 | 0.275 | 0.058 | 0.334 |
| CoT | 0.252 (↓24.6%) | 0.209 | 0.512 | 0.182 | 0.223 | 0.271 | 0.055 | 0.308 |
| Raw Trajectory | 0.352 (↑5.4%) | **0.317** | 0.604 | 0.380 | 0.329 | **0.363** | 0.105 | 0.364 |
| A-MEM | 0.334 (≈ 0.0%) | 0.286 | 0.590 | 0.366 | 0.339 | 0.332 | 0.112 | 0.313 |
| EXPEL | 0.329 (↓1.5%) | 0.306 | 0.594 | 0.379 | 0.317 | 0.327 | 0.092 | 0.287 |
| Ours | **0.365 (↑9.3%)** | 0.316 | **0.622** | **0.382** | **0.358** | 0.354 | **0.128** | **0.392** |
| **Qwen3-4B** | | | | | | | | |
| TIR | 0.306 (–) | 0.298 | 0.581 | 0.351 | 0.268 | 0.281 | 0.077 | 0.290 |
| Direct Inference | 0.211 (↓31.0%) | 0.158 | 0.413 | 0.157 | 0.183 | 0.240 | 0.033 | 0.290 |
| CoT | 0.181 (↓40.8%) | 0.149 | 0.375 | 0.146 | 0.156 | 0.190 | 0.022 | 0.228 |
| Raw Trajectory | 0.325 (↑6.2%) | 0.310 | 0.558 | 0.379 | 0.282 | 0.344 | 0.076 | 0.327 |
| A-MEM | 0.319 (↑4.2%) | 0.310 | 0.586 | 0.381 | 0.272 | 0.269 | 0.091 | 0.325 |
| EXPEL | 0.321 (↑4.9%) | 0.312 | 0.570 | 0.388 | 0.294 | **0.347** | 0.075 | 0.263 |
| Ours | **0.351 (↑14.6%)** | **0.335** | **0.596** | **0.393** | **0.299** | **0.347** | **0.099** | **0.391** |

more complex reasoning scenarios, we adopt HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), Musique (Trivedi et al., 2022), and Bamboogle (Press et al., 2023), which require integrating information across multiple documents.

### 5.2 Baselines

To comprehensively evaluate the effectiveness of our proposed method, we design experiments from two complementary perspectives: **(1) Inference Impact:** We assess how the integration of our memory workflow influences model performance in training-free settings, i.e., during direct inference. **(2) Training Impact:** We investigate how the memory architecture affects RL training dynamics, focusing on convergence speed and the final performance achieved.

We compare our framework with baselines covering different levels of experience utilization. **Memory-free methods** include **Direct Inference**, which uses the LLM without external reasoning or memory, and **CoT** (Wei et al., 2023), which elicits intermediate reasoning steps. We also include tool-augmented baselines, **TIR** (Chai et al., 2025) and **Search-R1** (Jin et al., 2025), where Search-R1 further learns tool-use behavior through RL; however, these methods do not explicitly reuse prior experience. **Raw experience replay** is represented by **Raw Trajectory**, which retrieves the nearest historical execution trace as a few-shot demonstration, preserving concrete step-by-step guidance but lacking structural abstraction and learnable retrieval. Finally, we compare with **abstract memory methods without utility learning**: **A-MEM** (Xu et al., 2025) builds structured memory notes and links them through semantic checks, while **EXPEL** (Zhao et al., 2024) distills past trajectories into high-level textual insights. Although these methods abstract experience, they do not explicitly calibrate memory utility using downstream reward feedback.

### 5.3 Experimental Settings

We conduct experiments with two model scales, **Qwen3-4B** and **Qwen3-8B**. For the retrieval component, we adopt the 2018 Wikipedia dump (Karpukhin et al., 2020) as the knowledge source and employ the **E5** (Wang et al., 2024) retriever. We report **Exact Match (EM)** as the primary evaluation metric.

We use disjoint HotpotQA subsets for memory construction and graph-weight training, and ensure that neither overlaps with downstream training or evaluation data. The constructed memory graph is then used to retrieve a small number of reward-calibrated meta-cognitive strategies for prompting. For fair comparison, all memory-based baselines are built from the same trajectory pool, use the

Table 2: Performance comparison across QA datasets in training. Avg. column also reports relative improvement (%) compared to Search-R1 as the base. † indicates in-domain datasets, while ⋆ denotes out-of-domain datasets.

| Methods | Avg. (↑ / ↓ vs. Search-R1) | General QA | | | Multi-Hop QA | | | |
|---|---|---|---|---|---|---|---|---|
| | | NQ⋆ | TriviaQA⋆ | PopQA⋆ | HotpotQA† | 2wiki⋆ | Musique⋆ | Bamboogle⋆ |
| **Qwen3-8B** | | | | | | | | |
| Search-R1 | 0.395 (–) | 0.384 | 0.651 | 0.429 | **0.391** | 0.386 | 0.143 | 0.380 |
| Raw Trajectory | 0.400 (↑1.27%) | **0.406** | 0.657 | 0.433 | 0.376 | 0.367 | 0.139 | 0.423 |
| A-MEM | 0.403 (↑2.03%) | 0.398 | 0.656 | **0.436** | 0.389 | **0.409** | 0.138 | 0.398 |
| EXPEL | 0.371 (↓6.08%) | 0.362 | 0.621 | 0.407 | 0.354 | 0.375 | 0.121 | 0.357 |
| **Ours** | **0.408 (↑3.29%)** | 0.386 | **0.662** | 0.434 | 0.387 | 0.403 | **0.152** | **0.435** |
| **Qwen3-4B** | | | | | | | | |
| Search-R1 | 0.375 (–) | 0.357 | 0.625 | 0.426 | 0.354 | 0.402 | 0.115 | 0.348 |
| Raw Trajectory | 0.415 (↑10.67%) | 0.403 | 0.624 | 0.434 | **0.420** | **0.428** | 0.186 | 0.412 |
| A-MEM | 0.388 (↑3.47%) | 0.393 | 0.603 | 0.439 | 0.385 | 0.322 | 0.157 | 0.418 |
| EXPEL | 0.337 (↓10.13%) | 0.322 | 0.577 | 0.399 | 0.311 | 0.363 | 0.081 | 0.305 |
| **Ours** | **0.426 (↑13.60%)** | **0.408** | **0.646** | **0.462** | 0.410 | 0.407 | **0.189** | **0.463** |

same teacher LLM for abstraction when applicable, and share the same retrieval corpus, retriever, prompting framework, and comparable prompt budget. RL-based baselines are trained under the same optimization settings as our method. Detailed hyperparameters and baseline configurations are provided in Appendix A.2.3.

### 5.4 Main results

**Experimental Analysis: Memory-Guided Inference**. The detailed inference results are summarized in Table 1. On the 8B-scale model, our method demonstrates strong competitiveness, achieving an average score of 0.365, which represents a notable **+9.3%** relative improvement over the TIR baseline and ranks first among all contenders. The gains are even more pronounced on the smaller Qwen3-4B model, where our method yields a substantial **+14.6%** relative improvement in average performance. These results suggest that structured strategic guidance is particularly useful for smaller models, which may benefit more from explicit decision-level priors.

Since EXPEL also uses GPT-4o to distill insights from the same trajectory pool, the gains over EXPEL suggest that our improvement is not merely due to the teacher LLM, but also to structured memory organization and reward-calibrated selection. Furthermore, although our memory is constructed only from HotpotQA, its gains on several out-of-domain datasets indicate that the FSM-based abstraction captures reusable strategies beyond the construction data.

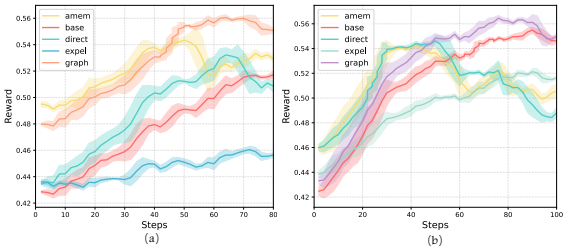


Figure 3: Training curves of Qwen3-4B and Qwen3-8B.

**Experimental Analysis: Memory-Guided RL**. We further evaluated our method by integrating it into the RL training process. The detailed training results are in Table 2.

On the Qwen3-8B model, our method achieves the best average performance (0.408), improving upon Search-R1 baseline by **3.29%**. This shows our method provides additional benefits even after the model is optimized with RL. The gains are most notable on challenging out-of-domain datasets like TriviaQA and Bamboogle, suggesting our memory may help the RL agent acquire strategies that transfer to several out-of-domain datasets.

The improvement is larger on Qwen3-4B, where our method achieves a 13.60% relative gain over Search-R1, surpassing the gain observed in the 8B model. Our trained Qwen3-4B model (0.426) outperforms the Qwen3-8B model baseline (0.395), suggesting that explicit memory can partially compensate for smaller model capacity in this setting.

As shown in Table 2, **EXPEL** underperforms the no-memory Search-R1 baseline in the RL setting, with drops of 6.08% and 10.13% on the 8B and 4B

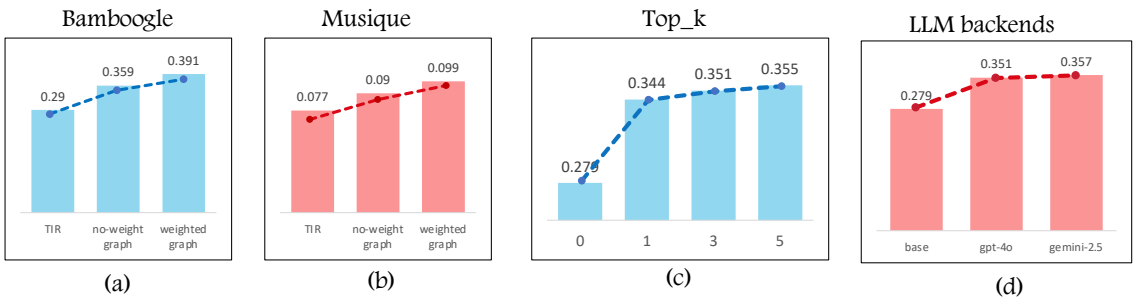


Figure 4: **Ablation studies of our framework.** (a) and (b) show the effect of disabling weight optimization. (c) varying the number of retrieved meta-cognitive strategies. (d) generalization across LLM backends.

models. We hypothesize that static, uncalibrated rule injection may introduce irrelevant or restrictive guidance that interferes with policy optimization. By contrast, **Raw Trajectory** remains a strong baseline and can help when retrieved examples closely match the target query, but its verbosity and lack of abstraction may limit cross-domain transfer. Our method distills trajectories into reusable meta-cognitive strategies and learns their utility through reward-driven graph weights, leading to the best average performance.

Overall, our method improves average performance in both inference and RL training, with larger gains on smaller models. This suggests that structured memory can transfer useful strategies from HotpotQA to several out-of-domain QA datasets and partially narrow the capacity gap.

### 5.5 Ablation Studies

To dissect the contribution of each component within our framework, we conduct ablation studies across three dimensions: (1) the impact of learnable graph weights, (2) the sensitivity to the number of retrieved meta-cognitions, and (3) the cross-model transferability of the constructed memory.

**Impact of Learnable Graph Weights.** We first examine the impact of freezing the graph weights (i.e., no learning of edge confidence). In this setup, we keep all edges at uniform weight and retrieve strategies purely based on structural presence. As shown in Figure 4 (a-b), performance drops particularly on Bamboogle, indicating that learning to prioritize high-utility memory connections is crucial for effective strategy reuse. This supports the usefulness of reward-calibrated graph weighting, which helps distinguish broadly useful meta-cognitions from less effective or overly specific ones.

**Sensitivity to Number of Retrieved Meta-Cognitions.** We evaluate how the number of retrieved meta-cognitive strategies affects model performance. Figure 4 (c) presents the average accuracy of the 4B model on seven benchmarks as a function of the number of meta-cognitions. Increasing $K_{\text{meta}}$ from 0 to 3 yields most of the performance gain, while further increasing it to 5 brings only marginal improvement. We therefore use $K_{\text{meta}} = 3$ in the main experiments as a prompt-efficient trade-off, rather than the numerically best setting. The detailed results are shown in Table 4.

**Cross-Model Generalization.** To test model-agnosticism, we evaluated the GPT-4o-constructed graph using a Gemini-2.5-Pro backend. Table 5 shows consistent performance gains over the baseline despite the change in LLM, suggesting that the constructed memory can remain useful when generated by different teacher LLMs.

## 6 Conclusion

In this paper, we address inefficient decision-making and poor experience reuse in LLM-based agents by introducing a trainable graph memory framework. The framework maps raw trajectories into finite-state transition paths and further distills them into high-level meta-cognitive strategies, which are organized in a structured memory graph. Unlike prior methods that rely on implicit parameter updates or static retrieval, our approach learns graph weights from RL signals to identify high-utility strategies and injects them as memory-augmented context during training. This mechanism promotes strategic transfer from past experiences while reducing the interference of uncalibrated memories. Experiments show that our method improves downstream reasoning performance in both inference and RL training settings, with gains across several in-domain and out-of-domain QA benchmarks.

## Limitations

While our framework shows promising results for utility-aware strategic memory in tool-augmented QA agents, several directions remain open for future work. First, our current memory nodes are represented as textual meta-cognitive strategies for interpretability and easy prompt integration. Future work could explore richer memory representations, such as structured templates, executable plans, or multimodal strategy descriptions, to better support agents operating in more diverse environments.

Second, our memory graph is constructed before downstream use and then applied during inference and RL training. This makes the additional cost largely amortized, while leaving room for future extensions that update the memory graph incrementally as new tasks arrive.

Finally, our utility learning uses downstream task rewards as the calibration signal. Future work could incorporate richer process-level feedback, such as tool-use efficiency or evidence quality, to make strategy selection more adaptive to different deployment goals.

Overall, our framework provides a structured foundation for future research on interpretable and adaptive memory systems for LLM agents.

## References


Anonymous. 2026. Paper title. Under review.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, and 12 others. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *Preprint*, arXiv:2204.05862.

Jiajun Chai, Guojun Yin, Zekun Xu, Chuhuai Yue, Yi Jia, Siyu Xia, Xiaohan Wang, Jiwen Jiang, Xiaoguang Li, Chengqi Dong, Hang He, and Wei Lin. 2025. Rlfactory: A plug-and-play reinforcement learning post-training framework for llm multi-turn tool-use. *Preprint*, arXiv:2509.06980.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. Mem0: Building production-ready ai agents with scalable long-term memory. *Preprint*, arXiv:2504.19413.

Yihong Dong, Xue Jiang, Yongding Tao, Huanyu Liu, Kechi Zhang, Lili Mou, Rongyu Cao, Yingwei Ma, Jue Chen, Binhua Li, Zhi Jin, Fei Huang, Yongbin Li, and Ge Li. 2026. Rl-plus: Countering capability boundary collapse of llms in reinforcement learning with hybrid-policy optimization. *Preprint*, arXiv:2508.00222.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. 2025. Retool: Reinforcement learning for strategic tool use in llms. *Preprint*, arXiv:2504.11536.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Dongming Jiang, Yi Li, Guanpeng Li, and Bingzhe Li. 2026. Magma: A multi-graph based agentic memory architecture for ai agents. *Preprint*, arXiv:2601.03236.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *Preprint*, arXiv:2503.09516.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *Preprint*, arXiv:1705.03551.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. 2025. Memory os of ai agent. *Preprint*, arXiv:2506.06326.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. 2020. Dense passage retrieval for open-domain question answering. *Preprint*, arXiv:2004.04906.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Rui Li, Zeyu Zhang, Xiaohe Bo, Zihang Tian, Xu Chen, Quanyu Dai, Zhenhua Dong, and Ruiming Tang. 2025a. Cam: A constructivist view of agentic memory for llm-based reading comprehension. *Preprint*, arXiv:2510.05520.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. 2025b. Search-o1: Agentic search-enhanced large reasoning models. *Preprint*, arXiv:2501.05366.

Zhiyu Li, Shichao Song, Chenyang Xi, Hanyu Wang, Chen Tang, Simin Niu, Ding Chen, Jiawei Yang, Chunyu Li, Qingchen Yu, Jihao Zhao, Yezhaohui Wang, Peng Liu, Zehao Lin, Pengyuan Wang, Jiahao Huo, Tianyi Chen, Kai Chen, Kehang Li, and 19 others. 2025c. Memos: A memory os for ai system. *Preprint*, arXiv:2507.03724.

Hao Liu, Zhengren Wang, Xi Chen, Zhiyu Li, Feiyu Xiong, Qinhan Yu, and Wentao Zhang. 2025. Hoprag: Multi-hop reasoning for logic-aware retrieval-augmented generation. *Preprint*, arXiv:2502.12442.

Zeyuan Liu, Jeonghye Kim, Xufang Luo, Dongsheng Li, and Yuqing Yang. 2026. Exploratory memory-augmented llm agent via hybrid on- and off-policy optimization. *Preprint*, arXiv:2602.23008.

Keer Lu, Chong Chen, Xili Wang, Bin Cui, Yunhuai Liu, and Wentao Zhang. 2026a. Pilotrl: Training language model agents via global planning-guided progressive reinforcement learning. *Preprint*, arXiv:2508.00344.

Mingfei Lu, Mengjia Wu, Feng Liu, Jiawei Xu, Weikai Li, Haoyang Wang, Zhengdong Hu, Ying Ding, Yizhou Sun, Jie Lu, and Yi Zhang. 2026b. Choosing how to remember: Adaptive memory structures for llm agents. *Preprint*, arXiv:2602.14038.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822, Toronto, Canada. Association for Computational Linguistics.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2022. Webgpt: Browser-assisted question-answering with human feedback. *Preprint*, arXiv:2112.09332.

Narjes Nourzad and Carlee Joe-Wong. 2026. Mira: Memory-integrated reinforcement learning agent with limited llm guidance. *Preprint*, arXiv:2602.17930.

Bo Pan and Liang Zhao. 2025. Can past experience accelerate llm reasoning? *Preprint*, arXiv:2505.20643.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. *Preprint*, arXiv:2210.03350.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. 2025. Toolrl: Reward is all tool learning needs. *Preprint*, arXiv:2504.13958.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Preprint*, arXiv:2302.04761.

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

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. 2025. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *Preprint*, arXiv:2503.05592.

Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Yan Zhang, Fei Huang, and Jingren Zhou. 2025. Zerosearch: Incentivize the search capability of llms without searching. *Preprint*, arXiv:2505.04588.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Hongru Wang, Cheng Qian, Wanjun Zhong, Xiusi Chen, Jiahao Qiu, Shijue Huang, Bowen Jin, Mengdi Wang, Kam-Fai Wong, and Heng Ji. 2025. Acting less is reasoning more! teaching model to act efficiently. *Preprint*, arXiv:2504.14870.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2024. Text embeddings by weakly-supervised contrastive pre-training. *Preprint*, arXiv:2212.03533.

Yingxu Wang, Jiaxin Huang, Mengzhu Wang, and Nan Yin. 2026. Trace: An experiential framework for coherent multi-hop knowledge graph question answering. *Preprint*, arXiv:2604.11193.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Zidi Xiong, Yuping Lin, Wenya Xie, Pengfei He, Jiliang Tang, Himabindu Lakkaraju, and Zhen Xiang. 2025. How memory management impacts llm agents: An empirical study of experience-following behavior. *Preprint*, arXiv:2505.16067.

Wujiang Xu, Kai Mei, Hang Gao, Juntao Tan, Zujie Liang, and Yongfeng Zhang. 2025. A-mem: Agentic memory for llm agents. *Preprint*, arXiv:2502.12110.

Chang Yang, Chuang Zhou, Yilin Xiao, Su Dong, Luyao Zhuang, Yujing Zhang, Zhu Wang, Zijin Hong, Zheng Yuan, Zhishang Xiang, Shengyuan Chen, Huachi Zhou, Qinggang Zhang, Ninghao

Liu, Jinsong Su, Xinrun Wang, Yi Chang, and Xiao Huang. 2026. Graph-based agent memory: Taxonomy, techniques, and applications. *Preprint*, arXiv:2602.05665.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. *Preprint*, arXiv:2210.03629.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, and Hao Zhou. 2025. Memagent: Reshaping long-context llm with multi-conv rl-based memory agent. *Preprint*, arXiv:2507.02259.

Yi Yu, Liuyi Yao, Yuexiang Xie, Qingquan Tan, Jiaqi Feng, Yaliang Li, and Libing Wu. 2026. Agentic memory: Learning unified long-term and short-term memory management for large language model agents. *Preprint*, arXiv:2601.01885.

Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V. Chawla. 2019. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 793–803, New York, NY, USA. Association for Computing Machinery.

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

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. Expel: Llm agents are experiential learners. *Preprint*, arXiv:2308.10144.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. 2025. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. *Preprint*, arXiv:2504.03160.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. 2025. Mem1: Learning to synergize memory and reasoning for efficient long-horizon agents. *Preprint*, arXiv:2506.15841.

# A Experimental Setup Details

## A.1 Multi-turn Tool-integrated QA

When tackling QA benchmarks, we observe that incorporating external knowledge retrieval plays a crucial role in enhancing answer accuracy. To this end, we utilize the 2018 Wikipedia dump as our external knowledge base and adopt the E5 retriever for efficient document retrieval. Within our framework, the entire retrieval process is encapsulated as a modular MCP tool, which the LLM can invoke autonomously based on its reasoning needs.

Specifically, the Qwen3 model follows a structured interaction protocol in which reasoning and planning are carried out between `<think>` and `</think>` tags,tool invocation commands are embedded within `<tool_call>` and `</tool_call>`, and the corresponding outputs are captured between `<tool_response>` and `</tool_response>`. The model engages in multiple rounds of tool interaction until it either generates a final answer enclosed in `<answer></answer>` or reaches the maximum limit of tool invocations, which is set to six in our experiments.

A concrete case study A.1 is presented in the following table to illustrate how this structured retrieval mechanism enables the model to access external knowledge and perform informed reasoning. Table H.3 presents a representative prompt designed to encourage agent to invoke the tool.

## A.2 Experimental Settings

### A.2.1 Artifact Use and Licensing

We use publicly available research artifacts, including QA benchmarks, the 2018 Wikipedia retrieval corpus, pretrained LLM backbones, and the E5 retriever. We have checked that these artifacts are used in accordance with their intended research purposes and original access conditions. All datasets and models used in this work are cited in the main paper. We do not redistribute the original datasets or retrieval corpus. The derived memory graphs and meta-cognitive strategies are used only for research on tool-augmented QA agents.

### A.2.2 Detailed Baseline Specifications

To provide a clearer understanding of the baselines used in our experiments, we offer extended descriptions of each method, including their core mechanisms, how they utilize memory or retrieval, and their inherent limitations. Table 3 presents

**Case 1: Example Response with Tool-Integrated Reasoning**

response: <think> Okay, let's tackle this question. The user is asking which U.S. state was historically dominated by the Republican party from 1860 to 1932 and is represented by State Senator George D. Maziarz.
First, I need to figure out the state in question. The key points here are the time period (1860-1932) and the Republican dominance. I remember that during the 19th century, the Republican Party was a major political force, especially after the Civil War. States like Ohio, New York, and Pennsylvania were known for their Republican strongholds. However, I'm not entirely sure which one fits the specific time frame mentioned.
Next, the mention of State Senator George D. Maziarz. I need to check if this senator is from a state that fits the historical Republican dominance. But I don't recall specific information about George D. Maziarz. Maybe I should look up his state of origin. However, since I can't access external resources right now, I need to rely on my existing knowledge.
... </think>

```
<tool_call>
{"name": "search-query_rag",
"arguments": {"query": "George D. Maziarz state", "topk": 3}}
</tool_call>}
```

<tool_response> Execute the tool search-query_rag succeeded - The args are: 'query': 'George D. Maziarz state', 'topk': 3 - The result is:..... </tool_response>
<answer> New York </answer>

Table 3: Comparison of Agent Memory Architectures

| Method | Memory Structure | Retrieval Mechanism | Trainable? | Key Limitation |
|---|---|---|---|---|
| Direct Inference/ COT/ TIR | None | N/A | No | No experience reuse across tasks |
| Raw Trajectory | Raw Flat Trajectories | Embedding Similarity | No | No abstraction; noisy |
| A-MEM | Semantic Graph | Similarity + Traversal | No | Heuristic; task-agnostic structure |
| EXPEL | Flat Insights | Semantic Similarity | No | No structure; static memories |
| **Ours** | **FSM-based Hierarchical Graph** | **RL-learned Utility Weights** | **Yes** | – |

some of the main differences among several types of memory using baselines.

**Direct Inference** represents the most basic setting, where the model generates answers purely based on its parametric knowledge from pretraining. It does not use any historical experience, external tools, or structured memory, and thus serves as an anchor point for evaluating the benefit of additional components. While simple and efficient, it lacks the ability to recall past mistakes or exploit cross-query information, making it limited for tasks requiring planning or adaptive strategies.

**Chain-of-Thought (CoT)** (Wei et al., 2023) enhances the basic model by encouraging it to articulate intermediate reasoning steps before producing an answer. This improves performance on tasks requiring logical decomposition. However, CoT still does not incorporate any past experience, and the reasoning steps themselves may be unstable or hallucinated, as they are freshly generated per query and not grounded in historical trajectories or external validation.

**Tool-Integrated Reasoning (TIR)** (Chai et al., 2025) equips the agent with external tool-use capabilities, allowing factual queries to be resolved by integrating retrieved evidence. Although this greatly improves factual correctness, the method still lacks any persistent memory, each query is resolved independently, and the agent does not reflect on past successes or failures. Moreover, the tool-use strategy is rule-based and not learned from reward.

**Search-R1** (Jin et al., 2025) adopts RL to optimize tool-use decisions, allowing the agent to learn better search strategies than TIR. Nevertheless, the agent lacks structured experience retention and can-

not build reusable abstractions, making it limited for cumulative learning scenarios.

**Raw Trajectory** stores raw past execution traces and retrieves the most similar trajectories as demonstrations for new queries. This method leverages episodic experience more directly than the previous baselines, but its memory remains unstructured: the system stores full trajectories without abstraction, relevance assessment, or quality control.

**A-MEM** (Xu et al., 2025) constructs a Zettelkasten-style memory graph by generating structured notes for each new experience and linking them to past memories through LLM-based semantic similarity checks.

**EXPEL** (Zhao et al., 2024) abstracts experiences into concise textual insights distilled from past trajectories. This improves interpretability and reduces redundancy relative to raw trajectories. However, its memory is static: once distilled, these insights do not adapt to new tasks or feedback. The system cannot evaluate which memories are more useful, nor update their importance across training, leaving it unable to perform long-term strategic refinement.

#### A.2.3 Other experiment settings

For memory construction and graph-weight optimization, we sample two disjoint subsets from HotpotQA, containing 1,000 and 5,000 examples, respectively. All benchmarks used in our experiments are English QA datasets. These subsets do not overlap with the downstream RL training set or any evaluation split. For each memory-construction query, we sample 3 trajectories to build query–transition–meta-cognition connections in the memory graph.

During retrieval, we first retrieve the top-$K_{\text{query}} = 5$ similar historical queries, propagate their similarity scores through the weighted memory graph, and select the top-$K_{\text{meta}} = 3$ meta-cognitions for prompting. During graph-weight optimization, we sample $N = 2$ meta-cognitive strategies for independent differential evaluation, where each strategy is evaluated by comparing rollouts with and without the selected guidance.

For fair comparison, all memory-based baselines are constructed from the same HotpotQA trajectory pool. A-MEM, EXPEL, and our method use the same GPT-4o teacher for abstraction when applicable. All methods share the same 2018 Wikipedia retrieval corpus, E5 retriever, and tool-integrated prompting/RL framework. We keep the number of retrieved memory items and maximum prompt budget comparable across memory-based methods. For RL-based baselines, we use the same training configuration as our framework, including learning rate, batch size, rollout number, KL coefficient, and maximum tool-call budget.

We report **Exact Match (EM)** as the primary evaluation metric. For memory construction in A-MEM (Xu et al., 2025), EXPEL (Zhao et al., 2024), and our proposed method, where a high-capability large language model is required, we utilized GPT-4o.

We conduct experiments on seven datasets, where HotpotQA is selected as the in-domain test set, while the remaining six datasets are used for out-of-domain evaluation. Unless otherwise specified, reported EM scores are from a single run due to the high computational cost of multi-turn tool-augmented RL.

During the RL training phase, we use the rest of the HotpotQA training set as the training corpus. We adopt a batch size of 512 with a micro batch size of 64, and the rollout sampling is performed with a temperature of 1.0. To accelerate the rollout process of the LLM, we deploy vLLM v1 with a tensor parallel size of 1.

Specifically for the GRPO algorithm, the number of rollout samples (n) is set to 8. All experiments are conducted on a cluster of 8 NVIDIA A100 GPUs.

## B Memory Graph Construction

The detailed descriptions of each type of node in the memory graph are as follows:

- **Query Layer** $\mathcal{Q}$: Each node $q_i \in \mathcal{Q}$ represents a specific task instance, such as a user-issued query. It encapsulates the entirety of an interaction, including the initial input, the agent's generated output, the complete execution trajectory, and a resultant outcome label (e.g., success or failure).

- **Transition Path Layer** $\mathcal{T}$: Each node $t_j \in \mathcal{T}$ corresponds to a standardized decision-making pathway. These pathways are grounded in a predefined FSM $\mathcal{S}$, representing a canonical sequence of the agent's states and actions. This layer abstracts away instance-specific details to reveal underlying behavioral patterns.

- **Meta-Cognition Layer** $\mathcal{M}$: Each node $m_k \in \mathcal{M}$ encodes a high-level, human-readable

strategic principle. These principles are distilled from a comparative analysis of successful and failed transition paths, representing generalized heuristics for effective problem-solving.

The pseudo-code for the overall process of the graph, which is composed of the specific paths of LLM models, is as shown in the algorithm 1. And in order to better explain the concept and workflow of the memory graph, we illustrate how our framework processes a specific HotpotQA query:*"Are the bands The Blakes and Big Supreme from the same country?"* in Figure 5.

### B.1 Mapping Trajectories into the Finite State Machine

The FSM serves as a structured cognitive map that regularizes highly variable CoT traces. Instead of relying on raw natural-language reasoning—which is verbose, stylistically inconsistent, and difficult to compare across tasks—the FSM abstracts an entire reasoning episode into a series of interpretable cognitive states. These states cover the complete lifecycle of decision-making, including intent parsing, uncertainty assessment, planning, tool execution, evidence analysis, and failure diagnosis. This abstraction removes surface-level linguistic noise and allows the system to analyze trajectories purely in terms of decision logic, the overall architecture of which is depicted in Figure 6.

A key requirement for accurate modeling of agent behavior is capturing how the agent reacts to external information. For this reason, the FSM contains an `InformationAnalysis` state that branches into three distinct cognitive responses: `KnowledgeAligned` (external evidence matches internal belief), `KnowledgeGap` (evidence provides new information), and `InternalConflict` (evidence contradicts prior parametric knowledge). Many reasoning failures—such as hallucination or incorrect synthesis—occur exactly at such conflict points. Including these branches allows the memory system to identify and aggregate structurally similar reasoning failures across tasks.

We use a powerful LLM to map the unstructured text into a sequence of predefined cognitive states (prompt in Table H.3). A concrete HotpotQA FSM example is shown in Table B.1.

Overall, the FSM enables the system to preserve the essential decision dynamics of the agent, including how it processes uncertainty and conflicting evidence, while eliminating noisy linguistic variability, providing a reliable basis for downstream learning and strategy induction.

### B.2 Meta-Cognition Induction and Evolutionary Update

We define a **Meta-Cognition Node** ($m_k \in \mathcal{M}$) not as a summary of a specific trajectory, but as a *generalizable heuristic* distilled from FSM transition paths. While a transition path describes *what* the agent did, a meta-cognition captures *why* a behavior led to success or failure and provides explicit guidance for future decisions. For example: "When external evidence contradicts parametric memory, prioritize the external source unless the task demands internal knowledge." Thus, meta-cognitions convert raw trajectory patterns into reusable, high-level behavioral principles.

The meta-cognition layer is built and refined by a high-capacity *Teacher LLM* (e.g., GPT-4o), prompt is given in Appendix H.3. Given a new trajectory pair, the LLM performs an iterative **Knowledge Consolidation** process: it first abstracts and contrasts the FSM paths to extract a candidate insight; then checks whether this insight matches an existing principle; and finally either *refines* that principle by integrating new evidence or *creates* a new node when a previously unseen reasoning pattern is detected. This keeps the memory compact while ensuring that it evolves with accumulated experience.

**Three Meta-Cognition Induction Scenarios** To cover diverse evidence conditions, meta-cognitions are induced under three complementary scenarios.

- **Scenario I: Intra-Query Contrastive Analysis (High Confidence).** When both a successful and a failed trajectory exist for the same query, their FSM paths are aligned to locate the main decision point of divergence. Because the query and evaluation protocol are shared, this setting provides a relatively controlled contrast for strategy induction. However, due to stochastic decoding and tool-use variability, the resulting meta-cognition should be viewed as a high-confidence heuristic rather than a definitive causal rule.

- **Scenario II: Inter-Query Analogical Analysis.** If a query contains only failures, we retrieve semantically similar queries with successful trajectories and compare their FSM

**Algorithm 1** Hierarchical Memory Graph Construction and Update

**Input:** Memory Graph $\mathcal{G}$, new query $q_i$, policy $\pi$, FSM $\mathcal{S}$, sample count $N$, similarity threshold $K$.
**Ensure:** Updated Memory Graph $\mathcal{G}'$.
**procedure** UPDATEMEMORYGRAPH($\mathcal{G}, q_i, \pi, \mathcal{S}, N, K$)
  $T_s \leftarrow \emptyset, T_f \leftarrow \emptyset$ ▷ Initialize sets for successful and failed paths
  $\mathcal{G} \leftarrow$ AddNode($\mathcal{G}, q_i$) ▷ Add current query to the graph
  **for** $n = 1$ **to** $N$ **do** ▷ Sample N trajectories from the policy
    $\tau_n \leftarrow$ SampleRollout($\pi, q_i$)
    $t_n \leftarrow$ GroundTrajectoryToPath($\tau_n, \mathcal{S}$) ▷ Map trajectory to a canonical FSM path
    $\mathcal{G} \leftarrow$ AddNode($\mathcal{G}, t_n$)
    $\mathcal{G} \leftarrow$ AddEdge($\mathcal{G}, q_i, t_n$)
    **if** IsSuccess($\tau_n$) **then**
      $T_s \leftarrow T_s \cup \{t_n\}$
    **else**
      $T_f \leftarrow T_f \cup \{t_n\}$
    **end if**
  **end for**
  $M_{new} \leftarrow$ InduceMetaCognition($q_i, T_s, T_f, \mathcal{G}, K$) ▷ Derive new strategic principles
  **for each** new meta-cognition $m$ **in** $M_{new}$ **do**
    $m_{exist} \leftarrow$ FindMatchingMetaCognition($m, \mathcal{G}$)
    **if** $m_{exist}$ is **null then**
      $m_{final} \leftarrow$ CreateNewMetaCognitionNode($m$)
      $\mathcal{G} \leftarrow$ AddNode($\mathcal{G}, m_{final}$)
    **else**
      UpdateConfidence($m_{exist}$)
      $m_{final} \leftarrow m_{exist}$
    **end if**
    **for each** path $t$ that generated $m$ **do** ▷ Link paths to the principles they support
      $\mathcal{G} \leftarrow$ AddEdge($\mathcal{G}, t, m_{final}$)
    **end for**
  **end for**
  **return** $\mathcal{G}$
**end procedure**
**procedure** INDUCEMETACOGNITION($q_i, T_s, T_f, \mathcal{G}, K$)
  **if** $T_s \neq \emptyset$ **and** $T_f \neq \emptyset$ **then** ▷ **Case 1:** High-confidence induction
    $t_s \leftarrow$ SelectOne($T_s$), $t_f \leftarrow$ SelectOne($T_f$)
    $m \leftarrow$ ContrastPaths($t_s, t_f$) ▷ e.g., find first diverging decision
    **return** $\{m\}$
  **else if** $T_s = \emptyset$ **and** $T_f \neq \emptyset$ **then** ▷ **Case 2:** Speculative induction
    $M_{spec} \leftarrow \emptyset$
    $Q_{sim} \leftarrow$ FindSimilarQueries($q_i, \mathcal{G}, K$) ▷ Based on embedding similarity
    **for each** similar query $q_j$ **in** $Q_{sim}$ **do**
      **for each** successful path $t_j$ of $q_j$ **do**
        $M_{spec} \leftarrow M_{spec} \cup$ GetMetaCognitionsFromPath($t_j, \mathcal{G}$)
      **end for**
    **end for**
    **return** $M_{spec}$
  **else**
    **return** $\emptyset$ ▷ No new insights if only successes or no rollouts
  **end if**
**end procedure**

**Case 2: An example of Finite State Machine**

**Illustrative Decision Path.** The following sequence illustrates a canonical decision path encoded within our framework:

$$\begin{aligned}\text{Start} &\rightarrow \text{CorrectGoalEstablished} \rightarrow \text{KnowledgeUncertainGap}\\ &\rightarrow \text{StrategyPlanning} \rightarrow \text{SequentialDependentPlanning}\\ &\rightarrow \text{ToolExecution} \rightarrow \text{InformationAnalysis}\\ &\rightarrow \text{KnowledgeAligned} \rightarrow \text{DecisionMaking}\\ &\rightarrow \text{InsufficientInformation} \rightarrow \text{AssumptionBasedReasoning}\\ &\rightarrow \text{AnswerGeneration} \rightarrow \text{WrongAnswer}\\ &\rightarrow \text{DiagnosisHub} \rightarrow \text{InternalKnowledgeConflict} \rightarrow \text{End}\end{aligned}$$

This path represents a chain of cognitive states traversed by the agent. It begins with goal establishment, proceeds through planning and execution, encounters a knowledge gap leading to flawed reasoning, and concludes with self-diagnosis. By encoding such trajectories as nodes in the transition path layer, the graph provides a structured and abstract representation of a complex reasoning process, which can be analyzed, compared, and learned from.

patterns. Although less direct, this produces *analogical heuristics* that transfer successful behavioral patterns from related tasks.

- **Scenario III: Positive Behavior Distillation.** When only successful trajectories are available, the system explicitly refrains from extracting meta-cognitive insights. This conservative mechanism prevents the framework from internalizing coincidental successes or spurious correlations, ensuring that the learned priors represent robust strategies derived from meaningful trial-and-error contrasts rather than environmental flukes.

Together, these scenarios allow the memory graph to prioritize high-confidence contrastive heuristics and transferable analogical strategies, while conservatively avoiding spurious insights from success-only evidence.

# C The impact of Meta-Cognition

## C.1 Qualitative Analysis: The Corrective Role of Meta-Cognition

To illustrate how meta-cognitive strategies regulate agent behavior, we present a case study in Figure 7 involving complex multi-hop reasoning.

In the **baseline scenario** (top), the agent relies on erroneous parametric memory, confidently hallucinating "Al Gore" as the 44th Vice President and explicitly refusing to use external tools (*"I don't think I need to use the search tool"*). This typifies intuition-driven failure.

In contrast, the **meta-cognition-augmented agent** (bottom) retrieves a strategy emphasizing *"structured planning and early synthesis verification."* Guided by this prior, the agent exhibits a distinct behavioral shift: it detects internal conflicts during reasoning (*"Wait, maybe I'm mixing up the numbers"*), suspends judgment to structure its thoughts, and strategically invokes the retrieval tool to verify the lineage. This case demonstrates that the induced meta-cognition effectively suppresses overconfident hallucinations and enforces a rigorous, evidence-based reasoning process.

## C.2 Meta-Cognition Activation Analysis Across Query Types

To understand how the proposed memory graph enables task-adaptive reasoning, we analyze the activation probabilities of all 30 meta-cognition nodes under two representative multi-hop query types—*Comparison* and *Bridge*. For each trained model, we record the selection probability of every meta-cognition node during inference and visualize this high-dimensional preference space using a t-SNE projection (Figure 8). Crucially, the purpose of this visualization is **not to perform semantic clustering**, but to characterize the **functional preference patterns** of the agent—specifically, which strategies are prioritized under distinct reasoning demands.

**General Meta-Cognitions.** We observe that a subset of meta-cognitions—such as *InternalKnowledgeCheck* exhibit consistently moderate activation across both Comparison and Bridge tasks. These meta-cognitions implement generic verification behaviors (e.g., quickly checking internal consistency or confirming tool outputs) and therefore serve as broadly applicable reasoning heuristics. Their stable appearance across task types indicates that the agent learns a set of **universally useful verification habits**.

**Task-Specific Meta-Cognitions.** In contrast, certain meta-cognitions—including *SequentialDependentPlanning* show sharply elevated activation in Bridge queries but low usage in Comparison queries. Bridge tasks typically rely on chained dependencies and frequently require the agent to identify missing information and execute follow-up tool calls. The selective activation of these strategies demonstrates that the memory system captures **scenario-specific reasoning preferences** rather than applying uniform heuristics.

Taken together, these activation patterns show that the hierarchical memory graph does not merely store abstract advice—it supports **task-adaptive meta-cognition**. Comparison queries emphasize lightweight verification, while Bridge queries require multi-step planning or targeted gap resolution. These patterns are consistent with the intended role of utility-weighted selection, which encourages different strategies to be preferred under different reasoning demands.

### C.3 Evolution of Meta-Cognition Selection During RL Training

To understand how RL reshapes the agent's strategic preferences, we tracked the selection probability of each meta-cognition node across three distinct training checkpoints (Batch 10, 150, and 250). Figure 9 visualizes this evolutionary process, providing a dynamic view of how the memory graph optimizes its retrieval distribution.

**Suppression of Low-Quality Meta-cognitions.** As illustrated in the top panel of Figure 9, the selection probability of brittle heuristics—such as `IncorrectInformationSynthesis` and `PrematureKnowledgeSufficient`—declines sharply throughout the training process ($0.136 \rightarrow 0.028 \rightarrow 0.017$). These strategies typically encourage over-reliance on parametric memory or premature halting of the reasoning chain. Since trajectories guided by these heuristics frequently lead to incorrect answers (and thus negative rewards), the RL policy naturally suppresses their activation weights to maximize expected return.

**Emergence of High-Quality Meta-cognitions.** In contrast, strategies that encourage structured verification and cautious synthesis become increasingly preferred during training, with their selection probability rising from 0.015 to 0.113. This trend suggests that reward-based training shifts the memory distribution toward strategies associated with more reliable multi-hop reasoning.

This analysis confirms that RL training does not merely tune the model's internal parameters—it gradually **reshapes the distribution over strategic principles**. The system actively filters out hallucination-prone shortcuts while cementing task-robust reasoning habits. This provides additional evidence that our memory graph functions as a **trainable policy prior** that evolves from experience, rather than a static retrieval module.

## D Additional Experimental Analysis

### D.1 Effect of the Number of Meta-Cognitive Strategies

To better understand how the quantity of retrieved meta-cognitive strategies affects agent performance, we evaluate four configurations: using 0 (no memory, denoted as TIR), 1, 3, and 5 strategies as contextual input. Results across seven QA benchmarks are presented in Table 4.

We observe that introducing even a single meta-cognitive strategy leads to a notable improvement over the baseline (TIR), especially on multi-hop tasks such as Bamboogle (+11.6%) and HotpotQA (+2.9%). This suggests that explicit strategic signals can substantially aid reasoning even in limited quantities. As the number of strategies increases, performance generally improves, but the marginal gains become smaller—likely due to redundancy or prompt saturation. The best overall result is achieved at `top_k=5`, while `top_k=3` offers a more efficient trade-off with only a small drop in average performance.

These findings imply that a moderate number of well-curated strategies can enhance generalization and decision quality, without incurring the risks of prompt overload or noise from irrelevant memories.

Table 4: Performance of different numbers of meta-cognition.

| Methods | General QA | | | Multi-Hop QA | | | | Avg. |
|---|---|---|---|---|---|---|---|---|
| | NQ$^\star$ | TriviaQA$^\star$ | PopQA$^\star$ | HotpotQA$^\dagger$ | 2wiki$^\star$ | Musique$^\star$ | Bamboogle$^\star$ | |
| **Qwen3-4B** | | | | | | | | |
| TIR | 0.298 | 0.581 | 0.351 | 0.268 | 0.281 | 0.077 | 0.290 | 0.306 |
| $top_k = 1$ | 0.326 | 0.583 | 0.382 | 0.290 | 0.327 | 0.096 | 0.406 | 0.344 |
| $top_k = 3$ | 0.335 | 0.596 | 0.393 | 0.299 | 0.347 | 0.099 | 0.391 | 0.351 |
| $top_k = 5$ | 0.333 | 0.594 | 0.392 | 0.299 | 0.349 | 0.094 | 0.418 | 0.355 |

### D.2 Cross-API Memory Robustness

To further validate the portability and reliability of our structured memory graph, we construct the memory using two distinct LLM APIs: GPT-4o and Gemini-2.5-pro. These memory graphs are then integrated into the same downstream agent architecture (Qwen3-4B and Qwen3-8B), and evaluated across seven QA datasets. As shown in Table 5, the resulting performance differences are minor, with Gemini-based memory slightly outperforming its 4o counterpart in most cases.

Specifically, on the multi-hop benchmark Bamboogle, the Gemini-constructed memory shows a notable increase (e.g., +0.043 on Qwen3-8B), while maintaining parity or marginal gains in general QA datasets like TriviaQA and PopQA. These results indicate that while different APIs may introduce slight variations in strategy abstraction, our framework is robust to such differences and maintains high effectiveness regardless of the underlying model used to generate the memory.

### D.3 Additional Study: 2Wiki for memory Construction

In our main experiments, we utilized HotpotQA to construct the FSM and memory graph due to its rich coverage of reasoning types (bridge, comparison, etc.). To rigorously examine whether our FSM abstraction relies on HotpotQA-specific structures, we conducted a cross-dataset portability study.

We repeated the entire memory construction pipeline using only trajectories sampled from the **2WikiMultiHopQA** dataset. The FSM definition (states and transitions) remained unchanged. We then evaluated this memory graph on the same seven downstream benchmarks using the Qwen3-4B model.

As shown in the result (Table 6), the memory graph constructed from 2Wiki still yields clear improvements over memory-free inference. This confirms that the FSM abstraction is dataset-agnostic and does not rely on Hotpot-specific structure. Our method is not tied to HotpotQA, and similar benefits are observed when using alternative multi-hop datasets. This experiment validates the generality and robustness of the proposed memory construction pipeline.

## E Computational Overhead Analysis

To assess the practical efficiency of our framework, we provide a granular breakdown of the computational overhead, categorized into offline graph construction (Stage 1 & 2) and online inference latency (Stage 3). All measurements were conducted on the same hardware setup used for the main experiments.

### E.1 Offline Graph Construction (One-Time Cost)

The construction of the memory graph is an offline preprocessing step performed once prior to RL training. As shown in Table 7, the primary costs stem from parsing trajectories into FSM paths and inducing meta-cognition via the powerful LLM.

- **Component-level Breakdown:** FSM parsing is efficient, averaging $4.086 \pm 0.879$ seconds per rollout. The meta-cognition update, which involves calling the powerful LLM (e.g., GPT-4o) to distill insights from trajectory pairs, requires $4.754 \pm 3.418$ seconds per query.
- **Total Preprocessing Time:** For a seed set of 500 queries, the cumulative time for Stage 1 is approximately 2.3 hours. The subsequent utility-weight optimization (Stage 2) is a batch process that takes an additional 2–3 hours.

Crucially, this combined cost of $\sim$5 hours is **fully amortized**. The resulting graph structure and

Table 5: Performance comparison across LLM.

| Methods | General QA | | | Multi-Hop QA | | | | Avg. |
|---|---|---|---|---|---|---|---|---|
| | NQ$^\star$ | TriviaQA$^\star$ | PopQA$^\star$ | HotpotQA$^\dagger$ | 2wiki$^\star$ | Musique$^\star$ | Bamboogle$^\star$ | |
| **Qwen3-8B** | | | | | | | | |
| Ours(4o) | 0.316 | 0.622 | 0.382 | 0.358 | 0.354 | 0.128 | 0.392 | 0.365 |
| Ours(gemini) | 0.318 | 0.621 | 0.385 | 0.362 | 0.336 | 0.123 | 0.434 | 0.369 |
| **Qwen3-4B** | | | | | | | | |
| Ours(4o) | 0.335 | 0.596 | 0.393 | 0.299 | 0.347 | 0.099 | 0.391 | 0.351 |
| Ours(gemini) | 0.337 | 0.598 | 0.396 | 0.314 | 0.360 | 0.093 | 0.397 | 0.357 |

Table 6: Performance comparison across memory dataset.

| Methods | General QA | | | Multi-Hop QA | | | | Avg. |
|---|---|---|---|---|---|---|---|---|
| | NQ$^\star$ | TriviaQA$^\star$ | PopQA$^\star$ | HotpotQA$^\star$ | 2wiki$^\dagger$ | Musique$^\star$ | Bamboogle$^\star$ | |
| **Qwen3-4B** | | | | | | | | |
| TIR | 0.298 | 0.581 | 0.351 | 0.268 | 0.281 | 0.077 | 0.290 | 0.306 |
| 2wiki-memory | 0.317 | 0.591 | 0.394 | 0.290 | 0.308 | 0.091 | 0.322 | 0.330 |

weights are frozen and reused throughout the entire RL training process and final inference, introducing zero recurrent cost during the agent's learning loop.

### E.2 Online Inference Latency

A key concern is whether the longer prompts (augmented with retrieved meta-cognition) significantly slow down inference. To quantify this, we measured the runtime on the **HotpotQA test set** (using a batch size of 8 with the Qwen3-8B backbone).

As detailed in Table 8, the inclusion of meta-cognition increases the average batch processing time from 32.156s to 32.511s. This results in an additional latency of only **0.355 seconds per batch** (or ∼0.044s per query). This represents a marginal overhead of approximately **1.1%**, which is negligible compared to the natural variability of LLM token generation. The minimum and maximum latencies are also nearly identical, confirming that our method does not introduce significant variance or instability to the inference pipeline.

## F Potential Risks and Intended Use

The current experiments use public QA benchmarks and a Wikipedia retrieval corpus, and do not involve private user data. However, if the framework is extended to real user interactions, memory construction may raise privacy concerns because trajectories could contain sensitive or identifying information. Practical deployments should include data filtering, anonymization, access control, and mechanisms for removing outdated or harmful memories.

## G Use of AI Assistants

AI assistants were used for limited writing support, including grammar correction, wording refinement, and improving the clarity and organization of the manuscript. They were not used to generate scientific claims, design experiments, fabricate results, or make authorship-level contributions.

## H Prompt Templates

### H.1 Multi-turn Tool-integrated QA Prompt

When LLMs need to interact with tools multiple times to answer a question, it is necessary to guide the LLM on which tools to use and how to use them. The specific prompt is shown in table H.3.

### H.2 Constructing FSM Path

Given the specific response path of the LLM and the complete structure of the state machine, we employ an LLM (e.g., GPT-4o) to map the generated answer onto one of the predefined paths in the state machine. The following shows the exact prompt used in H.3.

### H.3 Meta-cognition Constructing

With both successful and failed state-machine paths available, we derive high-level meta-cognitions

Table 7: **Offline Construction Cost Breakdown.** Statistics are measured over 500 seed queries. Note that Utility Update is a batch process measured end-to-end.

| Component | Unit | Time (Mean $\pm$ Std) | Variance | Notes |
|---|---|---|---|---|
| FSM Parsing | per rollout | $4.086 \pm 0.879$ s | 0.772 | Depends on trajectory length |
| Meta-cognition Induction | per query | $4.754 \pm 3.418$ s | 11.685 | Triggered once per 3 rollouts |
| **Stage 1 Total** | 500 queries | $\approx 2.3$ hours | – | One-time construction |
| **Stage 2 Total** | End-to-End | $\approx 2.0$–$3.0$ hours | – | One-time weight optimization |

Table 8: **Inference Time Comparison.** Statistics are derived from HotpotQA test dataset with a batch size of 8. The overhead introduced by meta-cognitive prompting is marginal ($\sim$1.1%).

| Setting | Unit | Mean Time (s) | Std (s) | Min (s) | Max (s) | Relative Overhead |
|---|---|---|---|---|---|---|
| w/o Meta-cognition | per batch | 32.156 | 3.759 | 26.715 | 38.675 | – |
| w/ Meta-cognition | per batch | 32.511 | 4.363 | 26.628 | 40.314 | +1.10% |
| **Difference** | **per batch** | **+0.355 s** | – | – | – | – |
| **Difference** | **per query** | **+0.044 s** | – | – | – | – |

by contrasting the two. The following prompt H.3 illustrates how a pair of successful and failed paths under the same query is used to induce meta-cognition.

**Prompt A: System and User Prompt**

**SYSTEM PROMPT:**

```
# Tools
You may call one or more functions to assist with the user query.
You are provided with function signatures within <tools></tools> XML tags:
<tools>
{
  "name": "search-query_rag",
  "description": "MCP RAG Query Tool (Synchronous Version)
  Args:
    query: query text
    topk: The default number of documents returned is 3
  Returns:
    str: The formatted query result",
  "parameters": {
    "type": "object",
    "properties": {
      "query": {"title": "Query", "type": "string"},
      "topk": {"default": 3, "title": "Topk", "type": "integer"}
    },
    "required": ["query"]
  }
}
</tools>

# Tool call format
For each function call, return a JSON object with function name and arguments
    within <tool_call></tool_call> XML tags:
<tool_call>
{
  "name": <function-name>,
  "arguments": <args-json-object>
}
</tool_call>
```

**USER PROMPT:**

```
Answer the given question. After reasoning, if you find you lack some
    knowledge, you can call the search tool.
You may search as many times as you want.
If you find no further external knowledge is needed, you can directly provide
    the answer inside <answer> and </answer>, without detailed illustrations.
For example: <answer> Beijing </answer>.

Question: Which US State, historically dominated by the Republican party from
    1860 to 1932, is represented by State Senator George D. Maziarz?
```

**Query: Are the bands The Blakes and Big Supreme, from the same country?**

***Trajectory (Fail)***

<think>
Okay, I need to figure out if The Blakes and Big Supreme are from the same country. Let me start by recalling what I know about these bands. \n\nFirst, The Blakes. I think they might be a British band. Let me check. The Blakes were indeed a British rock band from the 60s. Big Supreme, I think, is also British. But to be certain, I should use the search tool to verify their origins. Let me call the search function for each band to get accurate information.
</think>
**<tool_call> search-query_rag("The Blakes origin country") </tool_call>**
**<tool_call> search-query_rag("Big Supreme origin country") </tool_call>**
<tool_response>…</tool_response>
<think>
From the search results:
**First, I looked up The Blakes**. The search results mention that The Blakes are a band named after a dream by bassist Snow Keim involving William Blake, an 18th-century English poet. The band originated in Seattle, Washington, USA.
**Alternatively, perhaps the user is referring to the brand as a band, but that's unclear. However, the search results for \"Big Supreme origin country\" point to the brand Supreme being American. But if \"Big Supreme\" is a band, maybe they are associated with the brand.**
</think>
<answer>Yes</answer>

***Trajectory (Success)***

<think>
Okay, I need to figure out if The Blakes and Big Supreme are from the same country.First, The Blakes. I think they might be a British band, but I'm not entirely sure. I should verify that.
Then there's Big Supreme. I'm less familiar with them and not sure about their origin either.Since I'm not certain, I should use the search tool.
</think>
**<tool_call>**
**{"name": "search-query_rag", "arguments": {"query": "The Blakes band country", "topk": 3}}**
**</tool_call>**
**<tool_call>**
**{"name": "search-query_rag", "arguments": {"query": "Big Supreme band country", "topk": 3}}**
**</tool_call>**
<tool_response>…</tool_response>
<think>
First, I searched for The Blakes. The results mention Norman Blake, an American musician from Tennessee and Alabama.
**The search for "Big Supreme" clearly shows they are from the United Kingdom.**
However, given the available information, the answer would be that Big Supreme is from the UK, and The Blakes (if referring to Norman Blake) are from the US.
Therefore, based on available info, they are not from the same country.
</think>
<answer>No</answer>

***Finite State Machine PATH***

Start -> CorrectGoalEstablished -> KnowledgeUncertainGap -> StrategyPlanning -> ParallelStrategy -> ToolExecution -> InformationAnalysis -> **InternalConflict -> DecisionMaking -> ContradictoryInformation** -> AnswerGeneration -> WrongAnswer -> DiagnosisHub -> InadequatePlanning -> End

***Finite State Machine PATH***

Start -> CorrectGoalEstablished -> KnowledgeUncertainGap -> StrategyPlanning ->ParallelStrategy -> ToolExecution -> InformationAnalysi s -> **KnowledgeAligned -> DecisionMaking -> SufficientInformation** -> AnswerGeneration -> Success -> End

***Meta-cognition***

**When processing multiple unknown entities in parallel, one must detect and resolve internal knowledge conflicts arising during the information analysis phase, as they are a direct consequence of inadequate planning and lead to an incorrect final answer.**

Figure 5: An example of graph construction.

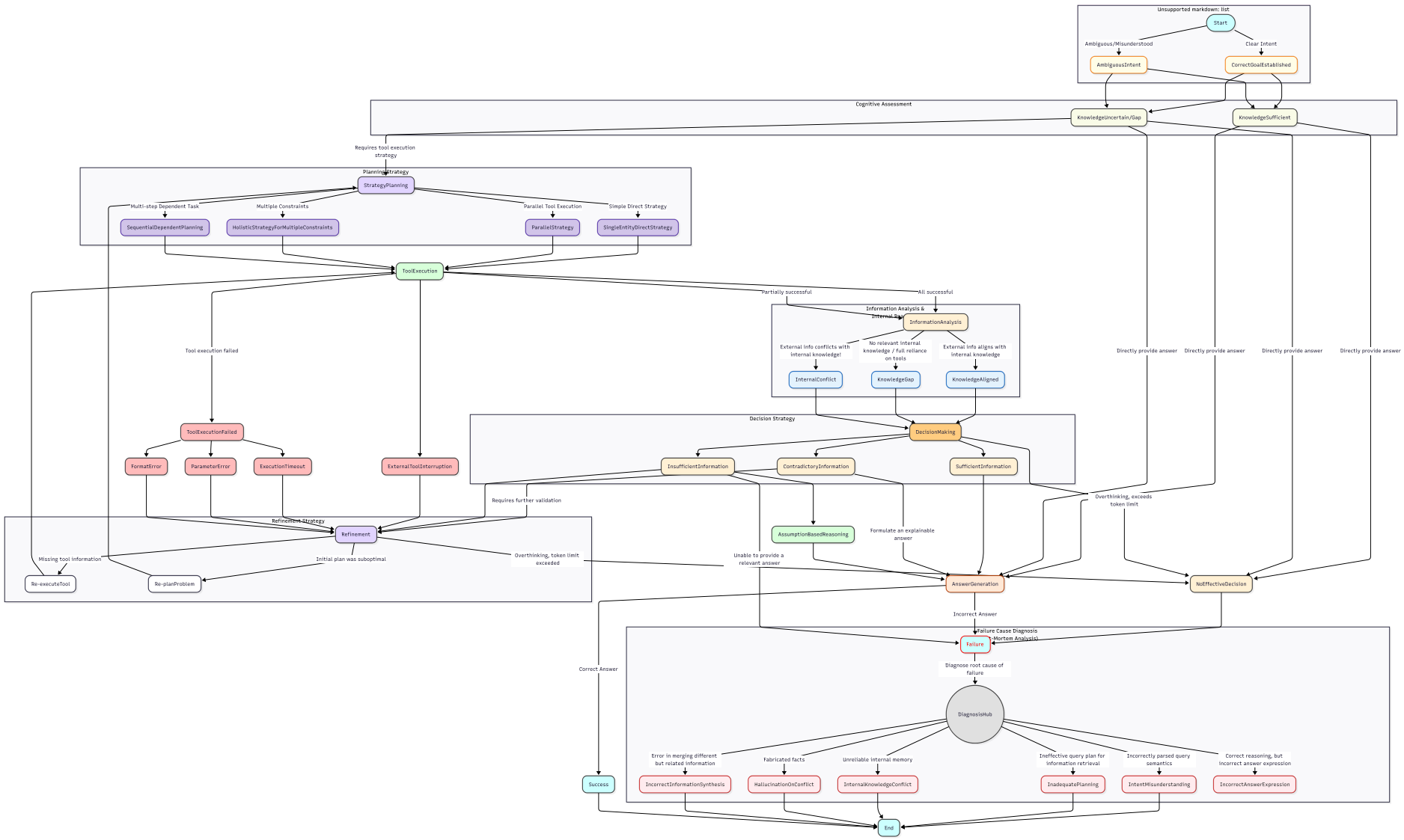


Figure 6: Overview of the finite-state machine. For readability, we group states into cognitive assessment, planning, tool execution, information analysis, decision making, answer generation, and failure diagnosis.

Query: Eugene Collins Pulliam, was an American newspaper publisher and businessman who was the founder and longtime president of Central Newspapers Inc., and was the maternal grandfather of which American politician and lawyer who was the 44th Vice President of the United States from 1989 to 1993?

## No Meta-cognition response

"<think>\nOkay, let's tackle this question. The user is asking about Eugene Collins Pulliam's maternal granddaughter, who became the 44th Vice President of the US from 1989 to 1993. \n\n**First, I know that the 44th Vice President was Al Gore.** But wait, the question specifies the maternal grandfather. So I need to confirm if Al Gore's mother is related to Eugene Pulliam.\n\nEugene Pulliam founded Central Newspapers Inc., and his daughter was Dorothy Pulliam. Dorothy married Albert Gore Sr., who was the father of Al Gore. So Al Gore's maternal grandfather would be Eugene Pulliam. \n\nWait, let me make sure. If Dorothy is the daughter of Eugene Pulliam, and she married Albert Gore Sr., then Al Gore's mother is Dorothy Gore. Therefore, Eugene Pulliam is Al Gore's maternal grandfather. That makes Al Gore the answer. \n **But I should double-check if there's any possibility of another person. The 44th VP was Al Gore, so the answer should be Al Gore. I don't think I need to use the search tool here because this is a well-known historical fact. So the answer is Al Gore.**\n</think>\n\n**<answer>Al Gore</answer>**<|im_end|>",

**Fact-based errors**

**Excessive confidence, lack of structured verification or stable reasoning strategies**

## Meta-cognition

**For factual queries with synthesis challenges due to ambiguous internal and external data, implementing structured sequential dependent planning and early focused synthesis verification tends to enhance accuracy. Recognizing ambiguities and consistently applying verification strategies can help mitigate synthesis errors.**

"<think>\nOkay, let's tackle this question. The user is asking about Eugene Collins Pulliam's maternal granddaughter who became the 44th Vice President of the US from 1989 to 1993. First, I need to recall who that VP was. I know that the 44th VP was Al Gore, but wait, Al Gore was the Vice President under Bill Clinton. Wait, no, actually, the 44th VP was Al Gore, but he was the 45th President. **Wait, maybe I'm mixing up the numbers. Let me think again.\n\nWait, the 44th Vice President was Dan Quayle, right?** No, Dan Quayle was the 44th VP, serving under George H.W. Bush. But the user is asking about the maternal granddaughter of Eugene Pulliam. Wait, Eugene Pulliam was the founder of Central Newspapers Inc. and a newspaper publisher. His daughter was named Dorothy, and Dorothy's husband was Dan Quayle's father? Wait, no. Let me check.\n\nAlternatively, maybe the answer is Al Gore. Wait, Al Gore's mother was Pauline Gore, and his father was Albert Gore Sr. But Eugene Pulliam's daughter was Dorothy, and Dorothy married Dan Quayle's father? **Wait, I'm getting confused here. Let me try to structure this.**\n\nEugene Collins Pulliam's maternal granddaughter would be the daughter of his daughter. Wait, the question says \"maternal grandfather,\" so the maternal side. So, if Eugene Pulliam is the maternal grandfather, then his daughter would be the mother of the VP. So, the VP's mother is Eugene's daughter. Let me think about who that could be.\n\nI recall that Dan Quayle's mother was Dorothy Quayle, and Dorothy Quayle was the daughter of Eugene Pulliam. So, Dan Quayle would be the maternal grandson of Eugene Pulliam. Therefore, the VP in question is Dan Quayle, who was the 44th Vice President under George H.W. Bush. Wait, but Dan Quayle was the 44th VP, serving from 1989 to 1993. Yes, that matches the time frame given. So the answer should be Dan Quayle. But wait, the user is asking for the politician and lawyer who was the 44th VP. Dan Quayle was a lawyer and politician, so that fits. \n**\nBut I'm not 100% sure. Maybe I should verify. Let me check if there's any conflicting information. Alternatively, maybe the answer is Al Gore. But Al Gore's mother was Pauline Gore, and I don't think she was related to Eugene Pulliam. So I think Dan Quayle is correct. However, to be certain, I should use the search tool to confirm the connection between Eugene Pulliam and Dan Quayle**.\n</think>\n\n<tool_call>\n{\"name\": \"search-query_rag\", \"arguments\": {\"query\": \"Eugene Collins Pulliam maternal granddaughter 44th Vice President\", \"topk\": 3}}\n</tool_call><|im_end|><|im_start|>user\n<tool_response>\n# Execute the tool search-query_rag successed\n - The args are: {'query': 'Eugene Collins Pulliam maternal granddaughter 44th Vice President', 'topk': 3}\n - The result is:\n{\n \"result\": [\n [\n {\n \"document\": {\n \"id\": \"17806128\",\n \"contents\": \"\\\"Eugene C. Pulliam\\\"\nEugene C. Pulliam Eugene

……

The Eugene S. Pulliam First Amendment Award is awarded annually by the Society of Professional Journalists in honor of Pulliam's dedication to First Amendment rights and values. The award seeks \\\"\\\"to honor a person or persons who have fought to protect and preserve one or more of the rights guaranteed by the First Amendment.\\\"\\\" Eugene S. Pulliam\"\n },\n \"score\": 0.8502768874168396\n }\n ]\n ]\n}\n</tool_response><|im_end|>\n<|im_start|>assistant\n<think>\nLooking at the search results: The first document mentions that Eugene C. Pulliam was the maternal grandfather of Dan Quayle. The third document also states that Dan Quayle, his half-nephew, was Vice President from 1989 to 1993. The second document talks about Myrta Pulliam, but she's mentioned as a cousin of Dan Quayle, not the granddaughter. \n\nSo, the maternal granddaughter would be Dan Quayle's mother. Wait, but the documents say Eugene Pulliam was the maternal grandfather, meaning his daughter is the mother of Dan Quayle. However, the third document refers to Dan Quayle as a half-nephew, which might imply a different family connection. But the first document clearly states he's the maternal grandfather. \n\nTherefore, the answer should be Dan Quayle. **The search results confirm his role as VP and his relation to Pulliam. No conflicting info here. So the answer is Dan Quayle.\n</think>\n\n<answer> Dan Quayle </answer>**<|im_end|>",

**At the initial stage, promptly identified the conflicts between the timeline and the conditions, avoiding the entire reasoning being based on wrong premise.**

**Without relying on subjective familiarity, use the tool for structural verification**

**Based on external search tools and own reasoning, provide an accurate answer.**

Figure 7: An example that demonstrates how meta-cognition affects reasoning

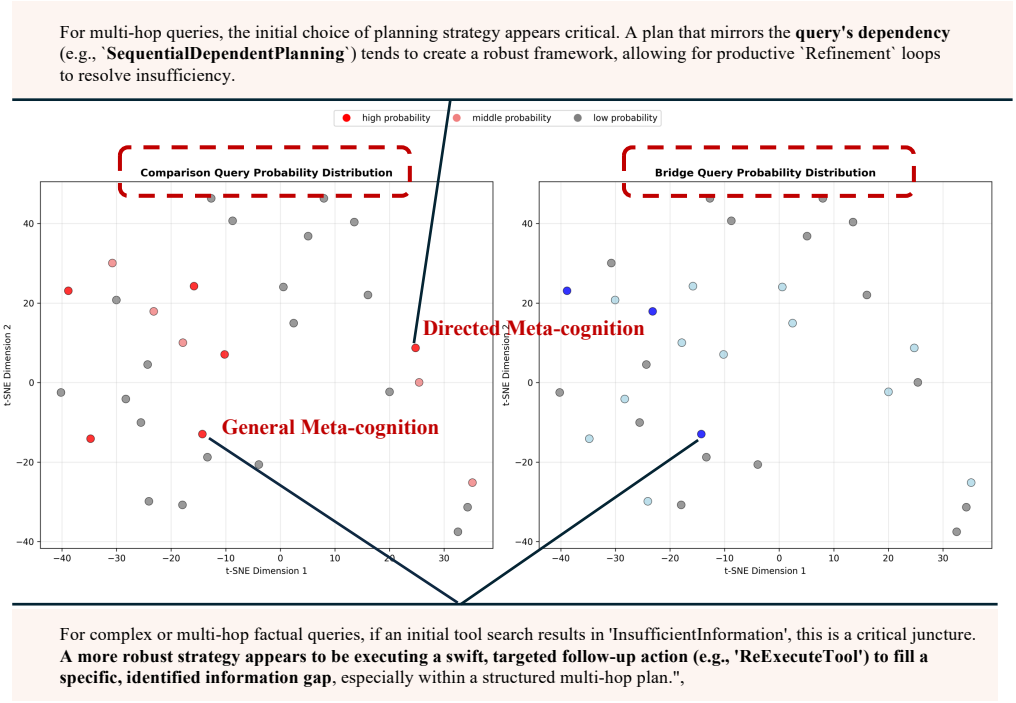


Figure 8: **Visualization of Meta-Cognitive Strategy Preferences.** The t-SNE projection maps the functional space of meta-cognition nodes based on their activation probabilities. This visualization reveals how the agent's preference shifts between *Comparison* and *Bridge* queries.

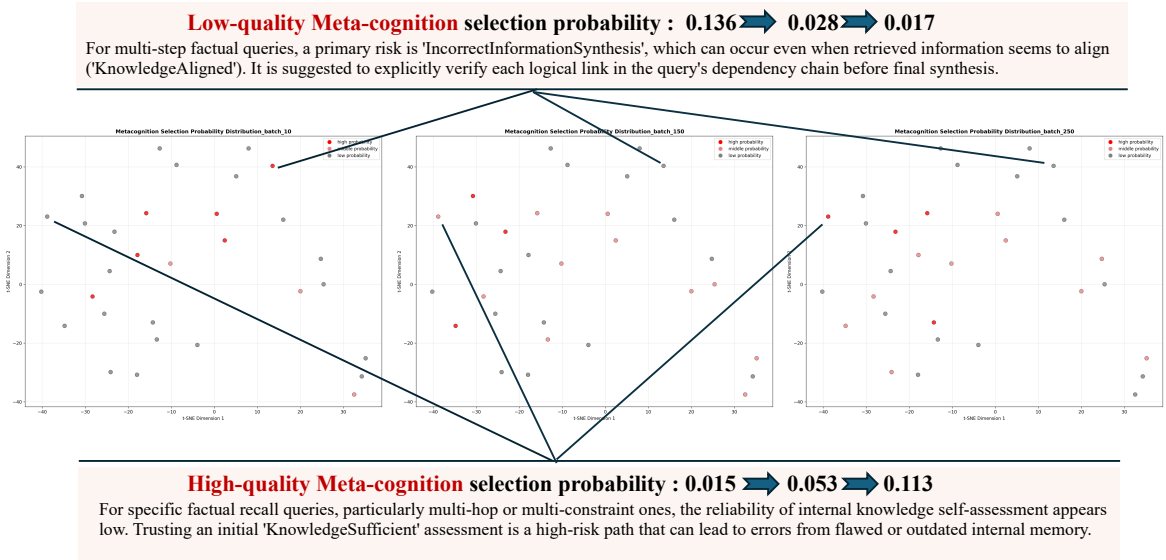


Figure 9: **Evolution of Strategic Preferences during RL Training.** We visualize the selection probability of meta-cognition nodes across three training checkpoints (Batch 10, 150, 250) using t-SNE projections. Each point represents a strategic principle; **color intensity** encodes the selection probability (Red = High, Gray = Low). As training progresses, the agent suppresses hallucination-prone heuristics (Top Panel) and converges toward robust verification strategies (Bottom Panel).

**Algorithm 2** Trainable Graph Weight Optimization via Policy Gradient

**Input:** Memory Graph $\mathcal{G}$ with initial weights $\mathbf{w}$, Training Queries $\mathcal{D}$, Agent model, Reward function $\mathcal{R}$, learning rate $\alpha$.
**Output:** Optimized Memory Graph $\mathcal{G}$ with updated weights $\mathbf{w}^*$.
**procedure** OPTIMIZEGRAPHWEIGHTS($\mathcal{G}, \mathcal{D}, \alpha$)
  **for each** query $q_{\text{new}}$ **in** $\mathcal{D}$ **do**
    ▷ — Step 1: Stochastic Guidance Selection —
    $m_k, p(m_k \mid q_{\text{new}}) \leftarrow$ SelectGuidingMetaCognition($\mathcal{G}, q_{\text{new}}$)
    **if** $m_k$ is **null then** ▷ No relevant guidance found
      **continue**
    **end if**
    ▷ — Step 2: Counterfactual Evaluation —
    $\text{Response}_{\text{with}} \leftarrow$ Agent.generate($q_{\text{new}}$, guidance $= m_k$)
    $R_{\text{with}} \leftarrow \mathcal{R}(\text{Response}_{\text{with}}, q_{\text{new}})$
    $\text{Response}_{\text{w/o}} \leftarrow$ Agent.generate($q_{\text{new}}$, guidance $=$ null)
    $R_{\text{w/o}} \leftarrow \mathcal{R}(\text{Response}_{\text{w/o}}, q_{\text{new}})$
    $\Delta R_k \leftarrow R_{\text{with}} - R_{\text{w/o}}$ ▷ Calculate reward gap (utility signal)
    ▷ — Step 3: Policy Gradient Update —
    $\nabla_{\mathbf{w}} \mathcal{L} \leftarrow -\Delta R_k \cdot \nabla_{\mathbf{w}} \log p(m_k \mid q_{\text{new}})$ ▷ Compute gradient for REINFORCE
    $\mathbf{w} \leftarrow \mathbf{w} - \alpha \cdot \nabla_{\mathbf{w}} \mathcal{L}$ ▷ Update all contributing weights
  **end for**
  **return** $\mathcal{G}$
**end procedure**
**procedure** SELECTGUIDINGMETACOGNITION($\mathcal{G}, q_{\text{new}}$)
  ▷ Activate relevant subgraph based on semantic similarity
  $\mathcal{M}_{\text{act}} \leftarrow$ ActivateSubgraph($q_{\text{new}}, \mathcal{G}$)
  **if** $\mathcal{M}_{\text{act}}$ is empty **then**
    **return null**, 0
  **end if**
  ▷ Compute relevance scores for all activated meta-cognitions
  **for all** $m \in \mathcal{M}_{\text{act}}$ **do**
    $S(m \mid q_{\text{new}}) \leftarrow 0$
    **for all** path $q_i \rightarrow t_j \rightarrow m$ in $\mathcal{G}$ **do**
      **if** $q_i$ is in activated subgraph **then**
        $S(m \mid q_{\text{new}}) \leftarrow S(m \mid q_{\text{new}}) + \text{Sim}(q_{\text{new}}, q_i) \cdot w_{qt}^{(i,j)} \cdot w_{tm}^{(j,m)}$
      **end if**
    **end for**
  **end for**
  ▷ Compute selection probabilities using softmax
  $Z \leftarrow \sum_{m' \in \mathcal{M}_{\text{act}}} \exp(S(m' \mid q_{\text{new}}))$
  **for all** $m \in \mathcal{M}_{\text{act}}$ **do**
    $p(m \mid q_{\text{new}}) \leftarrow \exp(S(m \mid q_{\text{new}}))/Z$
  **end for**
  ▷ Stochastically sample a meta-cognition based on probabilities
  $m_k \leftarrow$ Sample($\mathcal{M}_{\text{act}}$, probabilities $= \{p(m \mid q_{\text{new}})\}$)
  **return** $m_k, p(m_k \mid q_{\text{new}})$
**end procedure**

**Prompt B: Prompt for constructing FSM path**

**Instruction:** You are a metacognition analysis expert specialized in extracting *generalized decision principles and guidance strategies* from state machine execution paths.
**State machine transition rules:** `{transitions_info}`
**Core Requirements:**

1. **Generalizability Focus**: Output strategies and principles must be general, applicable to similar problems, without specific query details.
2. **Direct Usability**: Generated content should be directly usable as guidance principles for new problems.
3. **Principled Expression**: Use cautious guidance terms like "consider", "may help", "tends to" rather than definitive statements.
4. **Concise Effectiveness**: Output only the most core insights, avoid redundancy and complexity.
5. **Quality Control**: Strictly evaluate whether there is sufficient evidence to support new metacognition.
6. **Knowledge Confidence Awareness**: Recognize that LLM's internal knowledge confidence varies across queries — success patterns may be domain-specific.
7. **Uncertainty Acknowledgment**: Express appropriate uncertainty in guidance principles, avoiding overly definitive conclusions.
8. **Quantity Management**: When metacognition count exceeds 30, prioritize updating low-confidence existing metacognitions.

**Output Format (Quantity-Aware):**
Your output must be a JSON object with the following structure:

```
{
  "decision": "update" or "create" or "skip",
  "target_meta_id": <ID of metacognition to update (only when decision is "
      update")>,
  "reasoning": "Brief explanation including quantity management when count >
      30.",
  "meta_cognition": {
    "summary": "Concise general guidance summary (use cautious language).",
    "strategy_principles": [
      {
        "principle": "...",
        "confidence": "high" | "medium" | "low",
        "confidence_score": 30 - 85
      },
      ...
    ],
    "overall_confidence": "high" | "medium" | "low",
    "evidence_paths": <int>,
    "uncertainty_note": "Brief acknowledgment of limitations or knowledge-
        dependency concerns."
  }
}
```

**Prompt C: Metacognition Prompt Specification**

**State machine transition rules:** `{transitions_info}`
**Core Requirements:**

1. **Generalizability Focus:** Output strategies and principles must be general, applicable to similar problems, without specific query details.
2. **Direct Usability:** Generated content should be directly usable as guidance principles for new problems.
3. **Principled Expression:** Use cautious guidance terms like "consider", "may help", "tends to" rather than definitive statements.
4. **Concise Effectiveness:** Output only the most core insights, avoid redundancy and complexity.
5. **Quality Control:** Strictly evaluate whether there is sufficient evidence to support new metacognition.
6. **Knowledge Confidence Awareness: Recognize that LLM's internal knowledge confidence varies across queries—success patterns may be domain-specific.**
7. **Uncertainty Acknowledgment: Express appropriate uncertainty in guidance principles, avoiding overly definitive conclusions.**
8. **Quantity Management:** When metacognition count exceeds 30, prioritize updating low-confidence existing metacognitions.

**Critical Self-Reflection Requirements:**

- **Pattern Validity:** Question whether identified patterns truly represent generalizable principles.
- **Knowledge Dependency: Consider if success stems from strategy effectiveness or the LLM's domain familiarity.**
- **Evidence Sufficiency:** Demand higher evidence standards for strategies that could mislead future queries.
- **Simplicity Over Complexity:** Favor simple, robust principles over complex, brittle ones.

**Metacognition Quantity Control Strategy:**

**When metacognition count $\leq$ 30:**
- Normal decision making: create, update, or skip based on evidence quality.
- Prefer creating new metacognition when patterns are sufficiently distinct.
- **Express appropriate uncertainty in new metacognitions.**

**When metacognition count $>$ 30:**
- **Strongly prefer UPDATE over CREATE**: Prioritize improving existing low-confidence metacognitions.
- Only create new metacognition if the pattern is exceptionally valuable and completely distinct.
- Target metacognitions with confidence levels "low" or "medium" for updates.

**Analysis Focus:**

1. **Success Pattern Identification:** Abstract reusable decision patterns from successful paths.
2. **Failure Cause Summary:** Identify generalizable errors to avoid from failed paths.

3. **State Transition Optimization:** Extract best practice principles for state machine execution.
4. **Knowledge Dependency Assessment: Evaluate whether patterns might be specific to certain knowledge domains.**
5. **Existing Knowledge Enhancement:** When quantity is high, focus on strengthening weak metacognitions.

**Decision Options:**

- **create:** Create new metacognition (when discovering valuable and distinct patterns, or when quantity $\leq 30$).
- **update:** Update existing metacognition (preferred when quantity $> 30$, especially targeting low-confidence ones).
- **skip:** Skip metacognition operation (when evidence is insufficient or has no new value).

**Skip Metacognition Situations:**

- Path data quality is poor, patterns are unclear.
- Existing metacognition already covers the pattern, new evidence shows no significant improvement.
- Success/failure path differences are not obvious, difficult to extract effective strategies.
- **Cannot distinguish whether success stems from strategy effectiveness or knowledge domain familiarity.**
- When quantity $> 30$ and no suitable low-confidence metacognition found for update.

**Output Format (Quantity-Aware):** Your output must be a JSON object containing:

```
{
  "decision": "update" or "create" or "skip",
  "target_meta_id": (when decision is update) ID of metacognition to update,
  "reasoning": "Brief decision analysis, must include quantity management when
      count > 30.",
  "meta_cognition": {
    "summary": "...",
    "strategy_principles": [
      {"principle": "...", "confidence": "high", "confidence_score": 80},
      {"principle": "...", "confidence": "medium", "confidence_score": 60}
    ],
    "overall_confidence": "medium",
    "evidence_paths": 7,
    "uncertainty_note": "..."
  }
}
```