Reasoning with Memory: Adaptive Information Management for Retrieval-Augmented Generation

Hieu Man[1*], **Ro-ee Tal**[2], **Abhishek Kumar**[2†], **JJ Cho**[2], **Benjamin Hsu**[2]

[1]University of Oregon, [2]AWS AI
hieum@uoregon.edu, {rttal,akmarou,chojaeji,benhsu}@amazon.com

Abstract

Multi-hop reasoning remains a fundamental challenge for Retrieval-Augmented Generation (RAG) systems. Recent approaches-from adaptive retrieval to agentic pipelines-struggle to maintain coherent intermediate reasoning states as chains grow longer. We introduce **State-Aware RAG**[1], a framework that addresses this limitation through an explicit *working memory* that serves as a dynamic cognitive workspace for reasoning. Our modular architecture features a lightweight, trainable *extractor* that learns to actively filter, consolidate, and update this working memory via a novel *Path-Outcome Dual Reward* paradigm, which balances local coherence with global strategy. The retriever and generator remain frozen, enabling plug-and-play flexibility. Experiments on eight QA benchmarks demonstrate state-of-the-art results, on average achieving +8.6% over the best memory-augmented baseline and +9.3% over the best RL-enhanced baseline. Our architecture generalizes seamlessly to stronger generators and retrievers without retraining, establishing dynamic memory management as a critical yet underexplored dimension for advancing RAG systems.

1 Introduction

While large language models (LLMs) have revolutionized natural language processing, limitations of their fixed parametric knowledge can lead to critical failures on tasks requiring external information. Retrieval-Augmented Generation (RAG) (Lewis et al., 2021) addresses this by grounding outputs in retrieved documents. While effective for simpler tasks, standard RAG struggles with complex multi-hop reasoning that requires synthesizing evidence across multiple retrieval steps (Liu et al., 2024; Hong et al., 2025). The core issue is that retrieved information is treated statically-simply appended to context-causing irrelevant, redundant, or conflicting information to accumulate and compound across reasoning steps.

Recent advances focus almost exclusively on **information acquisition**-improving what enters the system, through query reformulation (Asai et al., 2023; Mao et al., 2024), iterative retrieval (Trivedi et al., 2023; Jiang et al., 2023b), query decomposition (Hu et al., 2025b; Jiang et al., 2025a), and component fine-tuning (Jin et al., 2025; Song et al., 2025). Yet these approaches neglect **information management**-how retrieved content should be actively curated, consolidated, and maintained throughout multi-step reasoning. Static reranking (Glass et al., 2022) or one-time context compression (Xu et al., 2023; Jiang et al., 2023a) are not resilient to context noise and bloat where each retrieval step may introduce irrelevant or contradictory information (Hong et al., 2025; Wu et al., 2025; Pan et al., 2025).

Memory-augmented architectures offer partial remedies. General-purpose memory systems (Yang et al., 2024; Chhikara et al., 2025; Li et al., 2025c) employ coarse-grained summarization for dialogue continuity, lacking the fine-grained filtering required for multi-hop inference. RAG-specific systems face fundamental constraints: HippoRAG (Gutierrez et al., 2024; Gutiérrez et al., 2025) constructs static knowledge graphs that cannot adapt to evolving trajectories, while MemoRAG (Qian et al., 2025) applies global compression that sacrifices turn-level coherence. None provide per-step memory curation that actively filters and consolidates information as reasoning progresses—instead relying on static structures, one-time compression, or coarse summarization that cannot adapt to intermediate reasoning states.

We introduce **State-Aware RAG**, a framework that addresses this gap through an explicit *working memory* that serves as a global, dynamic cogni-

[*]Work conducted during internship at Amazon.
[†]Corresponding author.
[1]https://github.com/amazon-science/state-aware-rag

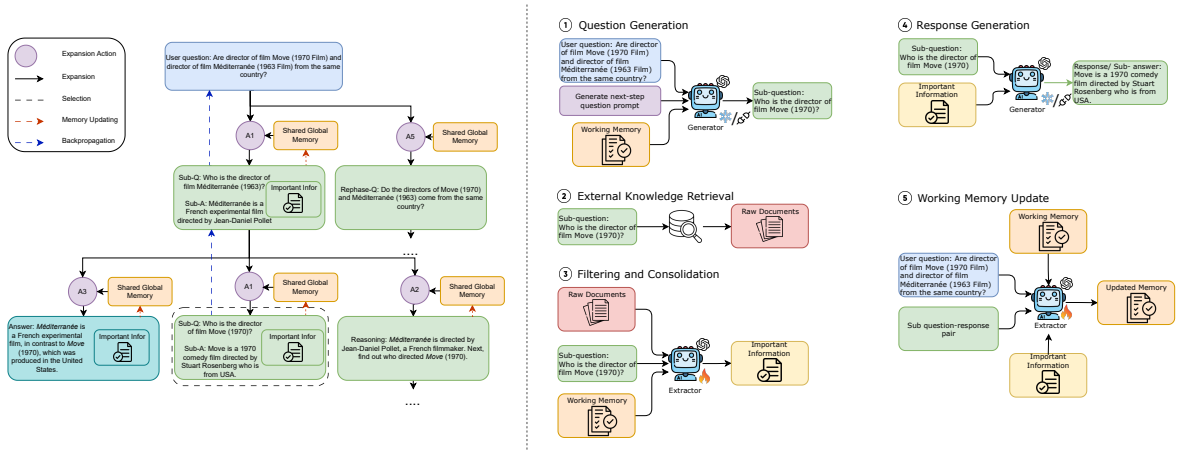

Figure 1: **State-Aware RAG Framework.** Left: overall inference process with dynamic, globally-shared working memory in MCTS. Right: node expansion, where each type of expansion actions (A1-5) executes one or more primitive operations: sub-question generation, document retrieval, content filtering, response generation, and memory update.

tive workspace. Drawing on cognitive models of human reasoning (Baddeley and Hitch, 1974), we design memory that: (1) *persists* across retrieval-generation cycles, (2) is *continuously updated* after each step, and (3) is *actively managed* by a specialized, lightweight component. This component (the extractor) learns effective information management strategies via *Path-Outcome Dual Reward*-a reinforcement learning paradigm that balances local coherence with global reasoning success. Our framework supports two inference modes: Socratic planning for efficient iterative reasoning, and Monte Carlo Tree Search (MCTS) for complex tasks requiring systematic exploration.

We evaluate State-Aware RAG on eight QA benchmarks against strong baselines spanning information-acquisition and memory-augmented approaches. Our method achieves state-of-the-art results: 58.0% average accuracy on multi-hop QA (+8.6% over HippoRAG 2) and 72.7% on single-hop QA (+9.3% over S3). Our key contributions:

1. *Working Memory Paradigm*: A dynamic working memory system that persists and evolves across reasoning steps, transforming RAG from stateless retrieval into goal-directed knowledge traversal.

2. *Path-Outcome Dual Reward*: A reinforcement learning framework that jointly optimizes for local coherence and global reasoning success, enabling effective memory curation.

3. *Plug-and-Play Architecture*: A modular design that keeps retriever and generator frozen, enabling seamless generalization to stronger components without retraining.

2 Related Work

Multi-Hop Retrieval-Augmented Generation. Multi-hop reasoning requires synthesizing information across multiple documents and retrieval steps (Khattab et al., 2023). Early approaches such as IR-CoT (Trivedi et al., 2023) interleave retrieval with chain-of-thought reasoning iteratively. Recent advances employ more sophisticated strategies: FLARE (Jiang et al., 2023b) triggers retrieval based on generation uncertainty; RaFe (Mao et al., 2024) decomposes queries for enhanced retrieval; Self-RAG (Asai et al., 2023) learns critique tokens for retrieval decisions; MCTS-RAG (Hu et al., 2025b) and RAG-Star (Jiang et al., 2025a) apply tree search over reasoning paths. These systems optimize *what* to retrieve rather than *how* to manage retrieved content, leading to context degradation as chains extend (Liu et al., 2024; Hong et al., 2025). The retrieval backbone underpinning these systems has itself advanced through LLM-based embedding models that better align pre-training objectives with dense passage retrieval (Wang et al., 2024b; Muennighoff et al., 2025; BehnamGhader et al., 2024; Man et al., 2024, 2025).

Memory-Augmented RAG. Memory mechanisms have been explored to address context limitations. General-purpose systems (Hu et al., 2025a; Anonymous, 2025; Zhang et al., 2025a) focus on dialogue continuity through coarse summa-

rization. RAG-specific approaches include HippoRAG (Gutierrez et al., 2024; Gutiérrez et al., 2025), which constructs static knowledge graphs, and MemoRAG (Qian et al., 2025), which applies one-time global compression. While effective for long-context retrieval, these systems struggle to adapt dynamically to evolving reasoning trajectories. Memory-R1 (Yan et al., 2025) trains agents for discrete memory operations (ADD, UPDATE, DELETE, NOOP) via outcome-driven RL, demonstrating that learned memory management outperforms heuristic pipelines. However, it targets episodic memory continuity across dialogue sessions rather than within-trajectory information curation during multi-hop reasoning-its discrete operations cannot perform fine-grained contradiction resolution or relevance-weighted consolidation from heterogeneous retrieved documents, and its working memory is not shared across concurrent reasoning branches. State-Aware RAG addresses these limitations through a unified extractor that maintains persistent, globally-shared working memory with continuous filtering and consolidation across reasoning steps.

Reinforcement Learning (RL) for RAG. RL has emerged as a key tool for optimizing RAG components beyond supervised fine-tuning. To address the temporal credit assignment problem in multi-step reasoning, recent works propose process-supervised rewards (Zhang et al., 2025b), turn-level advantage estimation (Wang et al., 2024a), and multi-agent coordination frameworks (Chen et al., 2025; Liu et al., 2025). Search-R1 (Jin et al., 2025) and S3 (Jiang et al., 2025b) train generators or query rewriters for autonomous search interaction. R3-RAG (Li et al., 2025b) adopts a two-stage cold-start + RL approach that jointly trains the LLM to reason and retrieve step by step, using a process reward based on retrieved document relevance. While this relevance-based process signal improves retrieval quality, it targets the *acquisition* side of the pipeline: the LLM is trained end-to-end without separation between retrieval and generation concerns, and accumulated context remains unmanaged across steps. C-3PO (Chen et al., 2025) trains a lightweight, modular policy model for agent routing and document filtering, but filters only newly retrieved documents rather than maintaining persistent memory across reasoning steps. Our *Path-Outcome Dual Reward* applies RL specifically to *memory management*-the path reward evaluates the coherence and sufficiency of curated working memory at each step, not retrieval relevance per se-explicitly balancing local curation quality with global reasoning success while keeping the generator and retriever frozen.

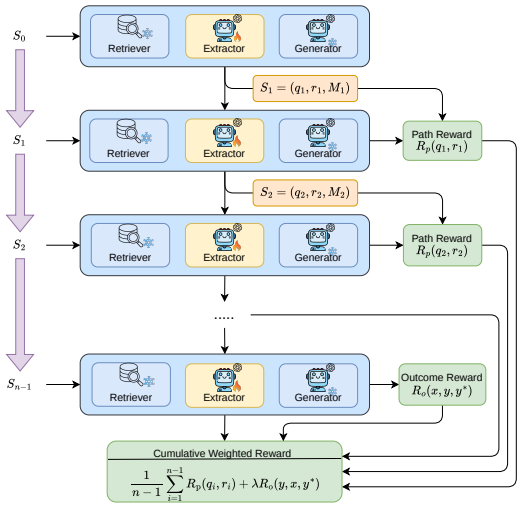

Figure 2: **Path-Outcome Dual Reward Training.** A single reasoning trajectory is unrolled to illustrate the actively trained Extractor operating alongside the frozen Retriever and Generator modules. At each intermediate step, the system receives a local path reward, which is combined with a global outcome reward evaluated at the terminal state. These dual signals are aggregated into a final cumulative weighted reward objective to optimize the dynamic memory extraction policy.

3 Methodology

We formulate question answering as a sequential decision-making problem. Given a question x and an external knowledge corpus K, the goal is to generate a correct final answer y by constructing a reasoning trajectory $T = \{S_0, a_0, S_1, a_1, \ldots, a_{n-1}, S_n\}$, where each state S_i encapsulates the reasoning progress and action a_i transforms S_i into S_{i+1}. The process begins with initial state $S_0 = x$ and continues until terminal state S_n that contains the final answer y. The objective is to learn a policy π that maximizes the likelihood of producing the correct answer y^*.

3.1 State-Aware RAG Framework

State-Aware RAG reconceptualizes retrieval-augmented generation around global working memory management. Unlike existing RAG approaches that are susceptible to context noise accumulation and reasoning drift, our approach maintains a per-

sistent, actively-curated memory state that evolves across reasoning trajectories.

State Representation. Each reasoning state $S_i = (q_i, r_i, M_i)$ comprises the current sub-question q_i, the intermediate response r_i, and global working memory M_i. Initially, $S_0 = (x, \varnothing, \varnothing)$. The memory M_i represents the updated memory *after* completing reasoning step i. M_i is not a passive document accumulation but rather a curated knowledge state containing only reasoning-relevant information. This enables the system to build upon prior discoveries, maintain coherence by resolving contradictions, and focus resources on knowledge gaps.

Modular Architecture. We factorize the policy π into three cooperative agents: *Retriever* R selects candidate documents from corpus K based on the current query; *Generator* G produces intermediate responses r_i and sub-questions q_{i+1} conditioned on the current state S_i; *Extractor* E serves as the central control module that filters, consolidates, and curates information to maintain working memory. Only the extractor is trained while retriever and generator remain frozen, preserving pre-trained capabilities and enabling targeted optimization.

Action Space and Reasoning Cycle. Actions a_i represent composite information management operations that transform state S_i to S_{i+1}. Unlike rigid retrieval-then-generate pipelines, our action space enables flexible, memory-aware strategies. Each action implements one or more primitives-question generation, retrieval, information curation, response generation, memory management-based on the current state. These system dynamics enable consolidation when M_{i-1} already contains sufficient evidence needed to answer q_i, or information gathering when M_{i-1} lacks the necessary information. Figure 1 illustrates a typical reasoning cycle. Because memory updates are conditioned on previous reasoning states, the system learns adaptive search dynamics-adjusting behavior based on accumulated knowledge rather than following predetermined paths.

Memory Management via Extractor. The extractor orchestrates adaptive working memory through two complementary phases. (1) *Consolidation*: given sub-question q_i, retrieved documents D_i, and current memory M_{i-1}, the extractor extracts relevant information:

$$I_i = E_{\text{consolidate}}(q_i, D_i, M_{i-1}) \tag{1}$$

where I_i represents distilled knowledge optimized for immediate reasoning needs. This performs relevance filtering (identifying facts supporting q_i), contradiction resolution (reconciling conflicts between D_i and M_{i-1}), and redundancy elimination (de-duplication). (2) *Memory Update*: after response generation $r_i = G(q_i, I_i)$, the extractor updates memory:

$$M_i = E_{\text{update}}(x, M_{i-1}, I_i, (q_i, r_i)) \tag{2}$$

This integrates the new question-response pair (q_i, r_i) and distilled knowledge I_i with M_{i-1}. The extractor learns this memory update through our dual-reward training signal (Section 3.2).

3.2 Training via Path-Outcome Dual Rewards

Training focuses exclusively on the extractor while freezing retriever and generator components, which preserves their pre-trained capabilities and reduces computational overhead. The key challenge lies in the temporal credit assignment problem inherent to multi-step reasoning (Chen et al., 2025; Zhang et al., 2025b): relying only on a final outcome reward yields sparse feedback that makes learning inefficient, while using only local step-wise rewards risks encouraging policies that are locally coherent but globally suboptimal. We address this through a Path-Outcome Dual Reward paradigm that provides complementary signals at different temporal scales.

Dual Reward Formulation. Both rewards are computed using an LLM-as-a-Judge. The path reward evaluates local reasoning quality at each step:

$$R_{\text{p}}(r_i, q_i) = \textsc{JudgePath}(r_i, q_i) \tag{3}$$

This assesses whether the extracted information I_i enables the generator to produce a response r_i that logically addresses q_i. The judge evaluates multiple dimensions including relevance of extracted information, sufficiency for answering the sub-question, logical coherence of the reasoning, and factual accuracy relative to the extracted content. The outcome reward evaluates global strategy:

$$R_{\text{o}}(y, x, y^*) = \textsc{JudgeOutcome}(y, x, y^*) \tag{4}$$

This evaluates whether the final answer y correctly addresses the original question x relative to ground truth y^*, capturing long-term dependencies that local rewards miss.

Training Objective. The extractor maximizes expected cumulative reward over reasoning trajectories:

$$\frac{1}{n-1}\sum_{i=1}^{n-1} R_{\text{p}}(q_i, r_i) + \lambda R_{\text{o}}(y, x, y^*) \quad (5)$$

where λ balances local coherence with global strategy. By jointly optimizing both objectives, the extractor learns to select information that is both immediately useful and strategically beneficial for the final answer.

3.3 Inference

Our framework supports two inference modes that provide different trade-offs between computational efficiency and reasoning thoroughness, both leveraging the trained state-aware memory system. Appendix C provides prompts and implementation details.

Monte Carlo Tree Search (MCTS) Exploration. For complex reasoning requiring exhaustive exploration (Figure 1), we adapt MCTS to operate over our state-aware framework (Jiang et al., 2025a; Hu et al., 2025b; Qi et al., 2024). Each tree node represents a reasoning state S_i. Edges correspond to five action types:

A1. Decompose & Answer: Incrementally decompose the problem by generating sub-question q_i; retrieve relevant documents and consolidate useful information I_i; generate an answer r_i; and finally update the memory M_i.

A2. Consolidate: Synthesize an intermediate conclusion by reasoning over the current memory M_{i-1} without additional retrieval.

A3. Refine: Verify and improve the current response r_i by re-evaluating it against the memory M_{i-1}. This enables self-correction when an answer appears incomplete, incorrect, or unsupported.

A4. Redirect: Reformulate the current question q_i to approach it differently. This enables resilience when retrieval fails or appears unproductive.

A5. Conclude: Produce the final answer y when current state S_i contains sufficient evidence.

The MCTS algorithm proceeds through repeated rollouts, traversing the tree using Upper Confidence Bound for Trees (UCT) (Kocsis and Szepesvári, 2006) and computing rewards $R_{\text{o}}(y, x, y^*)$ using the generator. Unlike prior MCTS-RAG approaches that treat branches independently, our method maintains globally coherent memory across the search tree so information discovered in one branch updates M_i and becomes available to other branches. Detailed prompts for each action are provided in Appendix C.3.

Socratic Planning. Socratic mode is a computationally efficient variant that restricts the action space to A1 (Decompose & Answer) and A5 (Conclude), producing a single reasoning chain rather than a tree. Building on iterative retrieval approaches like IR-CoT (Trivedi et al., 2023), the system iteratively decomposes the question into sub-questions one at a time, answering each with memory-augmented retrieval, until the generator determines the main question is answerable. Unlike methods that passively accumulate retrieved context, our approach applies the trained extractor to filter and consolidate information at each step, preventing context contamination that degrades performance in longer chains.

4 Experiment Setup

4.1 Training Setup

To adapt the extractor for memory management tasks, we require data that captures complete reasoning trajectories rather than static question-answer pairs. We adopt the diversity-aware query sampling strategy from SimpleDeepSearcher (Sun et al., 2025) to select 871 seed questions from *HotpotQA* (Yang et al., 2018) and *2WikiMultihopQA* (Ho et al., 2020). This selection optimizes for domain heterogeneity, keyword diversity, and reasoning complexity (2–5 hops). We synthesize complete reasoning trajectories using State-Aware RAG with MCTS and DeepSeek-R1 as the oracle extractor. Our oracle assumption is that a sufficiently powerful model can perform near-optimal information consolidation and memory management with appropriate prompting, thereby providing supervision for training smaller, deployable extractors. After filtering trajectories that fail to reach correct answers, we obtain approximately 10K reasoning paths (5–10 steps each), totaling approximately 70K state-action pairs.

We use Qwen3-8B (Yang et al., 2025) as the generator, Qwen3-Embedding-4B (Zhang et al., 2025c) as the retriever over the 2023 Wikipedia dump, and Qwen3-4B (Yang et al., 2025) as the extractor. Following established RL practices, we train the extractor in two stages: supervised fine-tuning (SFT) on synthesized trajectories for adaptation, followed by reinforcement learning with GRPO (Shao et al., 2024) using *Path-Outcome Dual Reward*. Detailed

Method	Multi-Hop QA								Single-Hop QA							
	2WikiMQA		HotpotQA		MuSiQue		Bamboogle		SimpleQA		NQ		TriviaQA		PopQA	
	sEM	Acc	sEM	Acc	sEM	Acc	sEM	Acc	sEM	Acc	sEM	Acc	sEM	Acc	sEM	Acc
Direct Generation																
Qwen3-8B (Yang et al., 2025)	23.6	21.7	28.1	30.5	10.6	6.2	15.2	17.6	18.9	19.2	37.3	39.3	40.8	42.6	19.9	21.9
CoT (Wei et al., 2023)	27.9	24.9	31.1	33.8	10.6	6.4	15.2	18.2	20.4	23.5	37.7	38.2	40.6	40.9	22.2	22.3
Standard RAG																
RAG (Lewis et al., 2021)	36.9	30.2	46.6	47.5	13.0	14.6	31.1	31.2	33.1	36.7	42.1	42.4	58.5	58.6	39.2	41.3
RAG + Rerank (Glass et al., 2022)	37.8	35.3	47.9	48.4	13.5	13.7	32.1	32.7	34.5	34.7	42.7	43.8	60.2	60.8	40.8	43.7
Iterative Retrieval																
IR-CoT (Trivedi et al., 2023)	47.6	48.3	50.9	51.8	16.4	17.8	32.2	33.8	35.9	37.0	43.2	44.7	65.6	67.2	54.5	56.8
FLARE (Jiang et al., 2023b)	48.2	47.9	49.1	52.8	16.8	18.1	31.7	33.0	34.8	36.9	44.6	46.2	65.7	68.0	54.4	54.9
Search-o1 (Li et al., 2025a)	30.6	32.8	37.5	40.8	19.2	21.3	37.6	38.4	41.6	44.0	54.2	57.4	55.6	61.1	48.6	50.5
Query Reformulation																
Self-RAG* (Asai et al., 2023)	43.8	45.6	45.7	47.0	14.3	15.1	27.6	30.1	36.3	49.1	45.9	47.3	66.4	68.5	54.9	55.7
RaFe* (Mao et al., 2024)	44.0	48.3	45.0	48.9	17.5	17.2	28.3	34.0	37.2	49.3	45.8	47.8	65.0	68.1	54.9	55.3
Planning-Enhanced RAG																
MCTS-RAG (Hu et al., 2025b)	43.2	47.9	47.1	51.7	24.9	27.5	34.5	35.8	40.0	44.2	57.4	59.6	70.9	72.7	48.1	50.4
RAG-Star (Jiang et al., 2025a)	45.9	48.8	49.0	53.4	27.0	30.2	32.5	35.3	40.7	44.9	58.8	60.3	72.2	74.9	50.3	50.9
RL-Enhanced RAG																
Search-R1* (Jin et al., 2025)	51.6	53.3	58.6	60.2	27.6	23.6	55.6	57.6	44.6	47.8	61.3	62.1	73.7	73.9	51.9	53.8
S3* (Jiang et al., 2025b)	51.6	54.1	59.0	60.1	23.9	25.8	58.1	60.1	46.2	48.3	66.1	68.0	78.5	79.9	57.4	57.4
Memory-Augmented RAG																
MemoRAG (Qian et al., 2025)	51.3	53.1	52.8	56.7	20.0	24.7	31.7	35.8	43.7	46.1	48.7	54.4	70.9	72.0	60.7	62.8
HippoRAG (Gutierrez et al., 2024)	58.1	60.5	55.4	56.9	22.8	27.2	36.8	38.1	43.8	54.7	50.3	54.1	71.1	72.0	48.8	49.7
HippoRAG 2 (Gutiérrez et al., 2025)	60.2	62.1	58.4	60.3	24.7	38.5	41.0	49.3	44.8	55.4	51.8	59.3	72.9	72.1	48.9	50.0
State-Aware RAG-Soc (*ours*)	59.9	62.6	58.6	60.5	29.8	42.2	60.8	62.8	**50.3**	**62.9**	66.5	**74.6**	**83.8**	**84.8**	**74.5**	**68.5**
State-Aware RAG-MCTS (*ours*)	**62.5**	**63.5**	**59.9**	**62.9**	**30.9**	**45.5**	**62.2**	**65.7**	**50.3**	**62.9**	**67.1**	75.8	82.6	82.9	74.0	68.3

Table 1: **Main results on single-hop and multi-hop QA benchmarks.** * indicates methods that fine-tune retriever or generator components; sEM denotes sub-string exact match; and Acc is accuracy using an Claude 3.7 Sonnet as the judge. Bold indicates best; underline indicates second-best. State-Aware RAG achieves consistent improvements across all benchmarks while training only the lightweight extractor.

hyperparameters are provided in Appendix B.

4.2 Evaluation

Benchmarks. Following prior work (Hu et al., 2025b; Jiang et al., 2025a), we evaluate on eight QA benchmarks. For multi-hop reasoning: *2WikiMultihopQA* (Ho et al., 2020), *HotpotQA* (Yang et al., 2018), *MuSiQue* (Trivedi et al., 2022), and *Bamboogle* (Press et al., 2023). For single-hop QA: *SimpleQA* (Wei et al., 2024), *Natural Questions* (Kwiatkowski et al., 2019), *TriviaQA* (Joshi et al., 2017), and *PopQA* (Mallen et al., 2023). To balance computational cost with statistical accuracy, we randomly sample 1,000 examples from each dataset; for Bamboogle, we use all 125 development examples.

Metrics. We employ two complementary metrics: *Sub-EM* (Substring Exact Match) measures whether the ground truth appears as a substring in the response, providing strict lexical matching. *Acc.* (Accuracy) evaluates semantic equivalence using Claude 3.7 Sonnet[2] as an LLM-as-a-Judge, which better handles paraphrasing and formatting differences.

[2]Claude 3.7 Sonnet

4.3 Baselines

We compare against representative baselines spanning seven categories: *(1) Direct Generation* establishes the parametric knowledge baseline. *(2) Standard RAG*: RAG (Lewis et al., 2021) and RAG+Rerank (Glass et al., 2022). *(3) Iterative Retrieval*: IR-CoT (Trivedi et al., 2023), FLARE (Jiang et al., 2023b), and Search-O1 (Li et al., 2025a). *(4) Query Reformulation*: Self-RAG (Asai et al., 2023) and RaFe (Mao et al., 2024). *(5) Planning-Enhanced RAG*: MCTS-RAG (Hu et al., 2025b) and RAG-Star (Jiang et al., 2025a). *(6) RL-Enhanced RAG*: Search-R1 (Jin et al., 2025) and S3 (Jiang et al., 2025b). *(7) Memory-Augmented RAG*: MemoRAG (Qian et al., 2025), HippoRAG (Gutierrez et al., 2024), and HippoRAG 2 (Gutiérrez et al., 2025). For fair comparison, we select baselines with comparable generator sizes in the 7–14B parameter range.

5 Results and Discussion

Table 1 presents comprehensive results across eight QA benchmarks. State-Aware RAG achieves substantial improvements over all baseline categories. On multi-hop tasks, our MCTS variant achieves 58.0% average accuracy, outperforming the best memory-augmented baseline HippoRAG 2 by +8.6% and the best RL-enhanced baseline S3 by +9.3%. On single-hop tasks, Socratic planning achieves 72.7% average accuracy (+9.3% over S3). The consistent gains across both simple and complex reasoning tasks demonstrate that dynamic working memory management provides a fundamental advantage over passive document accumulation.

For single-hop tasks, Socratic and MCTS modes achieve comparable performance, suggesting Socratic planning is sufficient when reasoning complexity is low. For multi-hop tasks, MCTS provides additional gains (e.g., +3.3% on MuSiQue, +2.9% on Bamboogle), justifying the additional computational cost when exhaustive exploration is beneficial. Notably, our extractor achieves strong performance while training on only 871 seed questions, demonstrating that the Path-Outcome Dual Reward framework enables effective learning from limited high-quality data.

5.1 Ablation Studies

We conduct ablations on Bamboogle to analyze key framework components.

Variant	Sub-EM	Acc
Socratic Planning		
State-Aware RAG (Full)	60.8	62.8
w/o Extractor	33.2	34.7
w/o Consolidation	51.5	55.2
w/o Memory Update	55.1	58.9
MCTS Planning		
State-Aware RAG (Full)	62.2	65.7
w/o Extractor	36.8	39.1
w/o Consolidation	53.0	56.3
w/o Memory Update	55.9	59.5

Table 2: **Memory management ablation on Bamboogle.** Each memory management component contributes to overall performance, with the trained extractor having the most significant impact.

Memory Management Operations. Table 2 isolates contributions of key memory management components. Removing the extractor entirely causes performance collapse (−28.1% for Socratic, −26.6% for MCTS); this variant degenerates to IR-CoT and MCTS-RAG respectively, confirming that passive document accumulation without active curation creates severe context noise. Eliminating memory updates degrades performance (−3.9% for Socratic, −6.2% for MCTS), demonstrating that persistent, global memory-our core contribution-provides meaningful benefits for multi-step reasoning. The larger MCTS drop indicates that exhaustive exploration particularly benefits from globally coherent memory enabling cross-branch knowledge transfer. Removing filtering and consolidation yields substantial losses (−7.6% for Socratic, −9.4% for MCTS), revealing that persistent memory alone is insufficient and that active curation prevents low-quality information from polluting memory states.

Training Strategy	Sub-EM	Acc
Socratic Planning		
Prompt-Only	53.5	57.4
SFT-Only	58.1	60.5
RL w/ Outcome Reward	59.3	61.7
RL w/ Path Reward	60.2	61.3
RL Dual Reward (Full)	60.8	62.8
Claude 3.7 as Extractor	66.3	69.7
MCTS Planning		
Prompt-Only	56.4	59.2
SFT-Only	60.2	63.4
RL w/ Outcome Reward	60.6	63.2
RL w/ Path Reward	60.9	64.0
RL Dual Reward (Full)	62.2	65.7
Claude 3.7 as Extractor	68.6	74.5

Table 3: **Extractor training strategy ablation on Bamboogle.** RL with dual rewards outperforms single-reward variants, and has greater influence in MCTS where further exploration and memory management operations are necessitated.

Extractor Training Strategies. Table 3 validates the Path-Outcome Dual Reward paradigm. Prompt-only extractors achieve moderate performance, indicating that effective memory management capabilities cannot be reliably elicited in "smaller" models through prompting alone. Single-reward RL variants demonstrate complementary strengths: outcome-only rewards guide global strategy but provide sparse local feedback, while path-only rewards improve intermediate reasoning. Interestingly, the path-only improvement suggests that LLM-as-a-Judge, though costly, provides flexible and effective dense reward signals beyond traditional outcome-based supervision. Our full dual-reward training achieves optimal performance, con-

firming that balancing local coherence with global strategy is essential for effective memory management. The gap to Claude 3.7 (-6.9% for Socratic) indicates further gains can be achieved through scaling the extractor.

5.2 Component Analysis

To assess modularity and scalability of our framework, we evaluate performance when substituting different generators and retrievers while keeping the trained extractor fixed (Figure 3).

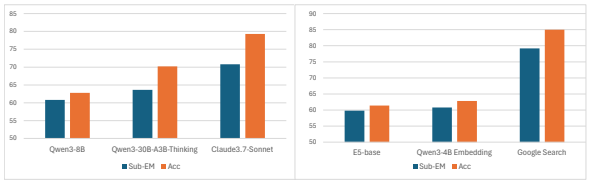

Figure 3: **Component scaling on Bamboogle (Socratic planning).** *Left*: generator scaling. *Right*: retriever/search scaling. The trained extractor generalizes across component choices without retraining.

Generator Scaling. Upgrading from Qwen3-8B to Qwen3-30B-A3B yields +7.4% accuracy, while replacing with Claude 3.7 Sonnet achieves +16.5%. These gains emerge without retraining the extractor, confirming that effective information management complements rather than constrains overall reasoning capacity.

Retriever Variations. Scaling from E5-base to Qwen3-Embedding-4B yields modest gains (+1.4%). However, replacing the static Wikipedia corpus with Google Search produces substantial improvements (+22.2%), highlighting that knowledge source coverage and ranking is often a primary bottleneck.

5.3 Inference Analysis

Figure 4 examines how key inference hyperparameters affect performance on Bamboogle. For both Socratic planning and MCTS, we vary the maximum reasoning steps from 3 to 15 and the number of rollouts from 3 to 15, analyzing their individual impacts while holding the other parameter constant.

Scaling Reasoning Steps. With rollouts fixed at 10, both inference modes exhibit rapid gains from 3 to 5 steps (+3.1% for Socratic, +2.7% for MCTS), after which performance plateaus. This saturation indicates that most Bamboogle questions can be resolved within 5 reasoning hops, and our framework efficiently recognizes when sufficient

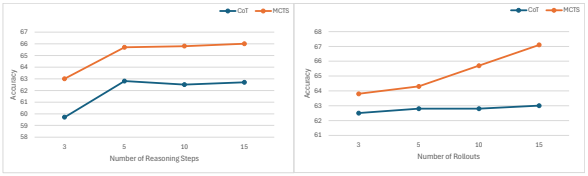

Figure 4: **Effect of inference hyperparameters on accuracy (%) on Bamboogle.** *Left*: scaling maximum reasoning steps with rollouts fixed at 10. *Right*: scaling number of rollouts with reasoning steps fixed at 5.

evidence has been accumulated. MCTS consistently outperforms Socratic across all step budgets (+2.9% average), confirming that exhaustive exploration benefits reasoning regardless of depth constraints.

Scaling Number of Rollouts. With the number of reasoning steps fixed at 5, the two modes exhibit markedly different scaling behaviors. Socratic performance remains flat across rollouts (62.5% to 63.0%), which is expected since Socratic follows a deterministic single-trajectory approach. In contrast, MCTS shows consistent gains with increased rollouts (+3.3% from 3 to 15), demonstrating that broader exploration of the reasoning tree discovers higher-quality solution paths. The widening gap between MCTS and Socratic at higher rollout budgets (from +1.0% at 3 rollouts to +4.1% at 15 rollouts) underscores the value of systematic exploration when computational resources permit.

Cost. On *2WikiMultihopQA*, MCTS requires $6.8\times$ more generator calls than Socratic mode (85.9 vs 12.7) but yields only a 2.6-point improvement in accuracy (62.5 vs 59.9). This limited gain suggests that for many applications, Socratic mode may offer a better cost-performance tradeoff.

6 Conclusion

Complex multi-hop reasoning tasks expose a fundamental limitation in existing RAG systems: passive information accumulation without consolidation can lead to noisy context, redundant retrieval, and error propagation across reasoning steps. State-Aware RAG addresses this through an explicit *working memory*-a persistent representation that is continuously updated by a trainable extractor. The extractor is trained via Path-Outcome Dual Reward reinforcement learning that successfully balances local coherence with global reasoning success. Experiments on eight QA benchmarks demonstrate state-of-the-art results, with improvements of +8.6% over the best memory-augmented base-

line and +9.3% over the best RL-enhanced baseline. Our modular architecture exhibits plug-and-play flexibility, generalizing seamlessly to stronger generators and retrievers without retraining. Future work includes extending to other reasoning-intensive tasks, adapting inference strategies to task complexity, and exploring extractor scaling laws to understand the minimum model capacity required for effective and feasible memory management deployment.

Limitations

While State-Aware RAG demonstrates significant improvements, several limitations warrant acknowledgment. First, the reliance on LLM-as-a-Judge for computing path and outcome rewards during reinforcement learning introduces substantial inference costs, as each candidate trajectory requires multiple LLM calls for reward estimation. While this provides reliable reward signals, it limits training scalability. Future work could explore distilling reward models into smaller critics or developing more efficient reward estimation techniques. Second, our component analysis demonstrates that retrieval quality fundamentally bounds reasoning performance: replacing the static Wikipedia corpus with Google Search yields +22.2% gains, highlighting that even optimal memory management cannot compensate for information that was never retrieved. Integrating State-Aware RAG with dynamic web-scale knowledge sources remains an important direction. Third, generalization to other domains (e.g., scientific literature, enterprise documents) and other reasoning-intensive tasks remains untested. Finally, MCTS inference mode, while more accurate, incurs substantially higher computational costs ($\sim 7\times$ more generator calls) for modest gains, which may limit deployment in latency-sensitive applications.

References

Anonymous. 2025. MEM1: Learning to synergize memory and reasoning for efficient long-horizon agents. In *Submitted to The Fourteenth International Conference on Learning Representations*. Under review.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *Preprint*, arXiv:2310.11511.

Alan D. Baddeley and Graham Hitch. 1974. In Gordon H. Bower, editor, *Working Memory*, volume 8 of *Psychology of Learning and Motivation*, pages 47–89. Academic Press. [link].

Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. 2024. Llm2vec: Large language models are secretly powerful text encoders. *Preprint*, arXiv:2404.05961.

Guoxin Chen, Minpeng Liao, Peiying Yu, Dingmin Wang, Zile Qiao, Chao Yang, Xin Zhao, and Kai Fan. 2025. C-3po: Compact plug-and-play proxy optimization to achieve human-like retrieval-augmented generation. *Preprint*, arXiv:2502.06205.

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

Michael Glass, Gaetano Rossiello, Md Faisal Mahbub Chowdhury, Ankita Rajaram Naik, Pengshan Cai, and Alfio Gliozzo. 2022. Re2g: Retrieve, rerank, generate. *Preprint*, arXiv:2207.06300.

Bernal Jimenez Gutierrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. HippoRAG: Neurobiologically inspired long-term memory for large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. 2025. From rag to memory: Non-parametric continual learning for large language models. *Preprint*, arXiv:2502.14802.

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.

Kelly Hong, Anton Troynikov, and Jeff Huber. 2025. Context rot: How increasing input tokens impacts llm performance. Technical report, Chroma.

Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. 2025a. HiAgent: Hierarchical working memory management for solving long-horizon agent tasks with large language model. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32779–32798, Vienna, Austria. Association for Computational Linguistics.

Yunhai Hu, Yilun Zhao, Chen Zhao, and Arman Cohan. 2025b. Mcts-rag: Enhancing retrieval-augmented generation with monte carlo tree search. *Preprint*, arXiv:2503.20757.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023a. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Confer-*

ence on Empirical Methods in Natural Language Processing, pages 13358–13376, Singapore. Association for Computational Linguistics.

Jinhao Jiang, Jiayi Chen, Junyi Li, Ruiyang Ren, Shijie Wang, Xin Zhao, Yang Song, and Tao Zhang. 2025a. RAG-star: Enhancing deliberative reasoning with retrieval augmented verification and refinement. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7064–7074, Albuquerque, New Mexico. Association for Computational Linguistics.

Pengcheng Jiang, Xueqiang Xu, Jiacheng Lin, Jinfeng Xiao, Zifeng Wang, Jimeng Sun, and Jiawei Han. 2025b. s3: You don't need that much data to train a search agent via rl. *Preprint*, arXiv:2505.14146.

Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023b. Active retrieval augmented generation. *Preprint*, arXiv:2305.06983.

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.

Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. 2023. Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive nlp. *Preprint*, arXiv:2212.14024.

Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *Proceedings of the 17th European Conference on Machine Learning*, ECML'06, page 282–293, Berlin, Heidelberg. Springer-Verlag.

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.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Preprint*, arXiv:2005.11401.

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

Yuan Li, Qi Luo, Xiaonan Li, Bufan Li, Qinyuan Cheng, Bo Wang, Yining Zheng, Yuxin Wang, Zhangyue Yin, and Xipeng Qiu. 2025b. R3-RAG: Learning step-by-step reasoning and retrieval for LLMs via reinforcement learning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 10491–10507, Suzhou, China. Association for Computational Linguistics.

Zhiyu Li, Shichao Song, Hanyu Wang, Simin Niu, Ding Chen, Jiawei Yang, Chenyang Xi, Huayi Lai, Jihao Zhao, Yezhaohui Wang, Junpeng Ren, Zehao Lin, Jiahao Huo, Tianyi Chen, Kai Chen, Ke-Rong Li, Zhiqiang Yin, Qingchen Yu, Bo Tang, and 3 others. 2025c. Memos: An operating system for memory-augmented generation (mag) in large language models. *ArXiv*, abs/2505.22101.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Yu Liu, Yanbing Liu, Fangfang Yuan, Cong Cao, Youbang Sun, Kun Peng, WeiZhuo Chen, Jianjun Li, and Zhiyuan Ma. 2025. Opera: A reinforcement learning–enhanced orchestrated planner-executor architecture for reasoning-oriented multi-hop retrieval. *Preprint*, arXiv:2508.16438.

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. *Preprint*, arXiv:2212.10511.

Hieu Man, Nghia Trung Ngo, Franck Dernoncourt, and Thien Huu Nguyen. 2024. ULLME: A unified framework for large language model embeddings with generation-augmented learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 230–239, Miami, Florida, USA. Association for Computational Linguistics.

Hieu Man, Nghia Trung Ngo, Viet Dac Lai, Ryan A. Rossi, Franck Dernoncourt, and Thien Huu Nguyen. 2025. LUSIFER: Language universal space integration for enhanced multilingual text embedding models. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Shengyu Mao, Yong Jiang, Boli Chen, Xiao Li, Peng Wang, Xinyu Wang, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. 2024. Rafe: Ranking feedback improves query rewriting for rag. *Preprint*, arXiv:2405.14431.

Niklas Muennighoff, Hongjin Su, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. 2025. Generative representational instruction tuning. *Preprint*, arXiv:2402.09906.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Xufang Luo, Hao Cheng, Dongsheng Li, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Jianfeng Gao. 2025. Secom: On memory construction and retrieval for personalized conversational agents. In *The Thirteenth International Conference on Learning Representations*.

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.

Zhenting Qi, Mingyuan Ma, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. 2024. Mutual reasoning makes smaller llms stronger problem-solvers. *Preprint*, arXiv:2408.06195.

Hongjin Qian, Zheng Liu, Peitian Zhang, Kelong Mao, Defu Lian, Zhicheng Dou, and Tiejun Huang. 2025. Memorag: Boosting long context processing with global memory-enhanced retrieval augmentation. *Preprint*, arXiv:2409.05591.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

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.

Shuang Sun, Huatong Song, Yuhao Wang, Ruiyang Ren, Jinhao Jiang, Junjie Zhang, Fei Bai, Jia Deng, Wayne Xin Zhao, Zheng Liu, Lei Fang, Zhongyuan Wang, and Ji-Rong Wen. 2025. Simpledeepsearcher: Deep information seeking via web-powered reasoning trajectory synthesis. *Preprint*, arXiv:2505.16834.

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*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, Toronto, Canada. Association for Computational Linguistics.

Huaijie Wang, Shibo Hao, Hanze Dong, Shenao Zhang, Yilin Bao, Ziran Yang, and Yi Wu. 2024a. Offline reinforcement learning for llm multi-step reasoning. *Preprint*, arXiv:2412.16145.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024b. Improving text embeddings with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11897–11916, Bangkok, Thailand. Association for Computational Linguistics.

Jason Wei, Nguyen Karina, Hyung Won Chung, Yunxin Joy Jiao, Spencer Papay, Amelia Glaese, John Schulman, and William Fedus. 2024. Measuring short-form factuality in large language models. *Preprint*, arXiv:2411.04368.

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.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. 2025. Longmemeval: Benchmarking chat assistants on long-term interactive memory. In *The Thirteenth International Conference on Learning Representations*.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2023. Recomp: Improving retrieval-augmented lms with compression and selective augmentation. *Preprint*, arXiv:2310.04408.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Kristian Kersting, Jeff Z. Pan, Hinrich Schütze, Volker Tresp, and Yunpu Ma. 2025. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *Preprint*, arXiv:2508.19828.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Hongkang Yang, Zehao Lin, Wenjin Wang, Hao Wu, Zhiyu Li, Bo Tang, Wenqiang Wei, Jinbo Wang, Zeyun Tang, Shichao Song, Chenyang Xi, Yu Yu, Kai Chen, Feiyu Xiong, Linpeng Tang, and Weinan E. 2024. Memory^3: Language modeling with explicit memory. *Journal of Machine Learning*, 3(3):300–346.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Guibin Zhang, Muxin Fu, and Shuicheng Yan. 2025a. Memgen: Weaving generative latent memory for self-evolving agents. *Preprint*, arXiv:2509.24704.

Wenlin Zhang, Xiangyang Li, Kuicai Dong, Yichao Wang, Pengyue Jia, Xiaopeng Li, Yingyi Zhang, Derong Xu, Zhaocheng Du, Huifeng Guo, Ruiming Tang, and Xiangyu Zhao. 2025b. Process vs. outcome reward: Which is better for agentic rag reinforcement learning. *Preprint*, arXiv:2505.14069.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025c. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

A Baseline Method Specifications

Table 4 provides detailed specifications for all baseline methods discussed in experiments (Table 1).

B Implementation Details

Here we provide comprehensive details on the training and inference setup for reproducibility.

B.1 Supervised Fine-Tuning Phase

We utilize Axolotl[3] as our training framework for the supervised fine-tuning phase. The extractor is initialized from Qwen3-4B-Thinking and fine-tuned using LoRA for parameter-efficient training. The detailed hyperparameters are presented in Table 5.

B.2 Reinforcement Learning Phase

For the RL training phase, we adopt verl[4] as our training framework, coupled with vLLM[5] as the inference engine for efficient rollout generation. The complete RL training hyperparameters are detailed in Table 6. The policy model is initialized from the model obtained after supervised fine-tuning. We use GRPO (Group Relative Policy Optimization) (Shao et al., 2024) which estimates advantages by comparing multiple responses per prompt rather than using a learned value function. We set the reward balance coefficient $\lambda = 2$, emphasizing outcome-based optimization while maintaining sufficient path-level supervision to guide local information selection decisions.

B.3 Inference Phase

In inference, we set the maximum reasoning steps to 5 for both CoT planning and MCTS, with the number of rollouts set to 10. For deployment, we establish a comprehensive infrastructure integrating multiple components:

Retriever Server: We construct our retrieval server using the Wikipedia 2023 dump as the primary knowledge source. We employ Qwen3-4B-Embedding as our dense retriever with FAISS[6] for efficient vector search.

LLM Service: We use SGLang[7] as our LLM server with LiteLLM[8] proxy for load balancing across multiple endpoints. We use 4096 max tokens with a default temperature of 1 for the generator and 0.1 otherwise. All agents use top-p sampling with $p = 0.9$.

C System Prompts

This section provides the complete prompts used in State-Aware RAG, organized by their role in the framework.

C.1 Prompt Overview

Each reasoning step involves five operations that map to specific prompts. ① **Question Generation** uses the Generator with the Sub-question prompt to decompose complex questions into atomic sub-questions and to propose search queries. ② **Retrieval** is performed by the Retriever using dense retrieval without any LLM prompt. ③ **Consolidation** employs the Extractor with the Extract prompt to filter documents and extract relevant facts. ④ **Response Generation** uses the Generator with the Answer prompt to produce answers from the filtered information. ⑤ **Memory Update** again uses the Extractor with the Extract prompt to update working memory with the question-answer pair and associated information. Notably, the Extract prompt serves dual purposes: consolidating retrieved documents and updating working memory.

The MCTS actions invoke different combinations of these prompts. **A1. Decompose & Answer** is the primary reasoning step, using the Generator and Extractor with the Sub-question, Answer, and Extract prompts. **A2. Consolidate** uses only the

[3] https://github.com/axolotl-ai-cloud/axolotl
[4] https://github.com/volcengine/verl
[5] https://github.com/vllm-project/vllm
[6] https://github.com/facebookresearch/faiss
[7] https://github.com/sgl-project/sglang
[8] https://www.litellm.ai/

Method	Retriever	Generator Backbone	Corpus
	Standard RAG		
RAG (Lewis et al., 2021)	Qwen3-Embedding-4B	Qwen3-8B	Wikipedia
RAG + Rerank (Glass et al., 2022)	Qwen3-Embedding-4B	Qwen3-8B	Wikipedia
	Iterative Retrieval		
IR-CoT (Trivedi et al., 2023)	Qwen3-Embedding-4B	Qwen3-8B	Wikipedia
FLARE (Jiang et al., 2023b)	Qwen3-Embedding-4B	Qwen3-8B	Wikipedia
Search-o1 (Li et al., 2025a)	Bing	Qwen3-8B	Web
	Query Reformulation		
Self-RAG* (Asai et al., 2023)	Contriever	Llama2-7B	Wikipedia
RaFe* (Mao et al., 2024)	Contriever	Qwen-max	Wikipedia
	Planning-Enhanced RAG		
MCTS-RAG (Hu et al., 2025b)	Bing	Qwen3-8B	Web
RAG-Star (Jiang et al., 2025a)	BGE-large-en-v1.5	Llama-3.1-8B-Inst	Wikipedia
	RL-Enhanced RAG		
Search-R1*(Jin et al., 2025)	E5-base-v2	Qwen-2.5-7B	Wikipedia
S3* (Jiang et al., 2025b)	E5-base-v2	Qwen-2.5-7B-Inst	Wikipedia
	Memory-Augmented RAG		
MemoRAG(Qian et al., 2025)	BGE-M3	Phi-3-mini-128K-Inst	Wikipedia
HippoRAG(Gutierrez et al., 2024)	Qwen3-Embedding-4B	Qwen3-8B	Wikipedia
HippoRAG 2(Gutiérrez et al., 2025)	Qwen3-Embedding-4B	Qwen3-8B	Wikipedia
State-Aware RAG (*ours*)	Qwen3-Embedding-4B	Qwen3-8B	Wikipedia

Table 4: Baseline method specifications. * indicates methods that fine-tune retriever or generator components. When possible we reproduce results using the same embedding and generator models as used in StateAware RAG. We use the following additional components - RaFe: Qwen-7b-base model for query re-writing; RAG-Star: Llama-3.1-8B-Instruct fine-tuned as the reward model; and Memo-RAG: Mistral- 7B-Instruct-v0.2-32K as the memory model.

Hyperparameter	Value
Learning Rate	2e-4
Batch Size	128
Number of Epochs	2
Optimizer	AdamW (torch fused)
LR Scheduler	Cosine
Warmup Ratio	0.1
Weight Decay	0.0
Base Model	Qwen3-4B-Thinking-2507
Sequence Length	8192
Adapter	LoRA
LoRA Rank (r)	32
LoRA Alpha (α)	64
LoRA Dropout	0.05
Chat Template	Qwen3
Flash Attention	Enabled
Sample Packing	Enabled

Table 5: Key hyperparameters in the supervised fine-tuning phase.

Hyperparameter	Value
Learning Rate	5e-7
Train Batch Size	64
Mini Batch Size	32
KL Coefficient	0.001
KL Loss Coefficient	0.001
Clip Ratio (Low/High)	0.2 / 0.28
Max Prompt Length	2048
Max Response Length	4096
Responses per Prompt	4
Total Epochs	5
Advantage Estimator	GRPO
Tensor Parallel Size	2
Sequence Parallel Size	4

Table 6: Key hyperparameters in the RL training phase.

Generator with the Consolidate prompt to synthesize information without retrieval. **A3. Refine** reattempts response generation using the Generator and Extractor with the Refine and Extract prompts. **A4. Redirect** reformulates the question using only the Generator with the Redirect prompt. **A5. Conclude** produces the final answer using the Generator and Extractor with the Finalize and Extract prompts. In Socratic mode, only A1 and A5 are used.

C.2 Core Framework Prompts

These prompts execute the five operations in each reasoning step (Figure 1, right).

Parameter	Value
Max Depth	5
Top-k (per expansion)	3
Number of Rollouts	10
Exploration Weight	1.0
Retriever Top-k	5

Table 7: MCTS search parameters.

Sub-question Generation Prompt (A1)

Used as part of the Decompose & Answer action to break down complex questions into atomic, targeted subquestions.

```
You are an expert assistant specializing in multi-hop
      question answering and reasoning decomposition.
      Your task is to analyze whether a main question can
       be answered with the provided context, and if not,
       generate a strategic subquestion that advances the
       reasoning process.

## Core Principle: The generated subquestion must NOT be
      answerable using the provided context. If a logical
       subquestion can be answered by the context, it is
      not a true knowledge gap, and you must look for the
       next piece of missing information.

## Step-by-Step Instructions:
1. Analyze the Main Question: Deconstruct the question to
       identify its core intent (e.g., factual lookup,
      comparison, causal link), key entities, and the
      information required for a complete answer.
2. Map Context to Requirements: Systematically check if
      the provided context contains all the facts,
      entities, and relationships identified in Step 1.
3. Decision Point: Assess Answerability:
   - If YES (Context is Sufficient): The main question
      can be fully and confidently answered. No
      subquestion is needed.
   - If NO (Context is Insufficient): The context is
      missing at least one critical piece of information.
       Proceed to the next steps.
4. If the Context is Insufficient, Execute the Following:
   a. Identify the Core Knowledge Gap: Pinpoint the most
      immediate and crucial piece of missing information.
   b. Formulate the Subquestion: Create a clear, self-
      contained question that precisely targets this
      single knowledge gap.
   c. CRITICAL VALIDATION: Before finalizing, verify that
       your formulated subquestion CANNOT be answered by
      the provided context.

Question: <question>
Context: <context>
```

Query Generation Prompt

Used to expand a question into multiple retrieval queries before the retrieval step.

```
You are a highly advanced Reasoning Engine. Your primary
      function is to deconstruct a user's Input (a
      question or statement) into a series of precise,
      self-contained, and essential search queries. The
      goal is to generate queries that, when answered,
      provide all the necessary facts to answer/verify
      the Input.

## Guiding Principles for Queries
1. The Zero-Synthesis Principle (Most Important): You
      MUST NOT introduce any new information, entities,
      or concepts that are not explicitly present in the
      original Input.
2. Fully Self-Contained: The query MUST BE SELF-CONTAINED,
       meaning it should be understandable and answerable
       without needing to refer to the original Input,
      other queries, or any external context.
3. Atomic: Each query must ask for one single,
      indivisible fact. Deconstruct questions containing
      conjunctions ("and", "or") or multiple attributes
      into separate queries.
4. Essential & Non-Redundant: Every query must be
      necessary for the final answer, and must seek a
      unique piece of information not covered by other
      queries.

## Instructions:
1. Parse the Input:
    - If the input is a question: Identify its type, key
      entities, and the required reasoning steps.
    - If the input is a statement: Deconstruct it into
      its core, verifiable claims.
2. Generate Strategic Queries: Formulate a list of search
       queries to resolve the Input.
3. Ensure Self-Containment: Each query must be
      understandable and answerable on its own.

Input: <question>
```

Extract Prompt

This prompt serves multiple purposes: (1) extracting relevant information from retrieved documents, (2) reflecting on existing memory, and (3) consolidating new information into updated memory. The same template is used with different inputs for each purpose.

```
You are a meticulous and insightful research analyst.
      Your primary objective is to build a comprehensive
      dossier of all information from the provided text
      that could help a user fully understand and answer
      their question. You prioritize thoroughness,
      context, and nuance. You must think step-by-step to
       ensure no helpful detail, however tangential, is
      overlooked.

## Instructions:
- Step 1: Question Deconstruction: First, carefully
      analyze the user's Question. Identify and list the
      primary subject, all key entities (people,
      organizations, concepts), and the specific
      information or insight the user is seeking. This is
       your 'search brief'.
- Step 2: Candidate Identification: Next, read the entire
       Raw Data and identify and quote ALL passages that
      seem potentially related to the concepts from Step
      1. Be liberal and inclusive in this initial pass;
      we will filter and refine in the next step. If no
      passages appear even remotely related, state this
      and proceed to Step 5.
- Step 3: Systematic Relevance Evaluation: Now, for each
      candidate passage quoted in Step 2, you must
      perform a systematic evaluation. Iterate through
      each quote and assess it against the following
      criteria: Directly Answering, Contextual,
      Supporting Evidence, Methodological, Alternative
      Perspectives, Related Concepts, Implications,
      Enrichment, Entities. For each candidate quote, you
       must state exactly which criterion (or criteria)
      it meets and provide a one-sentence justification
      for your assessment. If a quote meets no criteria,
      mark it as 'Not Relevant'.
- Step 4: Extraction: Extract ALL relevant information.
      Ensure the extraction is strictly verbatim and
      includes full sentences to preserve context.
- Step 5: Final Decision: Based on your analysis in the
      preceding steps, state your final decision: '
      relevant' or 'not_relevant'. A document is only '
      not_relevant' if it contains ZERO information that
      could relate to any entity or concept in the
      question.

Question: <question>
```

```
Raw Data: <document>
```

Answer Generation Prompt (A1)

Used as part of the Decompose & Answer action to generate intermediate answers.

```
You are an expert assistant specializing in precise, well-
      reasoned question answering. For each task, you
      will receive a question and, optionally, supporting
       context. Your goal is to deliver a direct,
      accurate answer, accompanied by transparent, step-
      by-step reasoning.

## Instructions:
1. Question Analysis: Carefully read and understand the
      question. Identify key components and clarify what
      is being asked.
2. Context Utilization: If context is provided, analyze
      it thoroughly. Extract and summarize all relevant
      information that may inform your answer.
3. Information Gap Identification: If the context does
      not fully answer the question, identify missing
      information. Formulate specific follow-up queries
      that would help fill these gaps.

Question: <question>
Context: <context>
```

C.3 MCTS Action Prompts

These prompts support MCTS actions A2-A5. Action A1 uses the core prompts above.

Consolidate Prompt (A2)

Used for the Consolidate action to synthesize knowledge from context without additional retrieval.

```
You are a specialized AI assistant for multi-step
      reasoning. Your sole function is to perform a
      single, focused reasoning step. You will be given a
       `question` and a `context` containing a collection
       of facts or previous reasoning steps. Your task is
       to analyze this information and produce a single,
      consolidated synthesis. Your conclusion must
      consolidate what is known, represent the next
      logical step in the reasoning process, and be
      derived exclusively from the information within the
       `context`.

## Instructions:
1. Analyze the Objective: Examine the main question to
      understand the overall goal of the reasoning task.
2. Review the `context`: Scrutinize all facts,
      definitions, and prior conclusions provided in the
      `context`. This is the sole source of information.
3. Determine the Next Logical Step: Based on `context`
      and `question`, decide on the most valuable
      reasoning action to perform.

## Critical Constraints:
1. No External Information: Do NOT introduce any facts,
      assumptions, or information not present in the `
      context`.
2. No New questions: Do not ask for new information. Your
       role is to synthesize, not to query.

Question: <question>
Context: <context>
```

Refine Prompt (A3)

Used for the Refine action to verify and improve existing answers.

```
You are an expert assistant specializing in rigorous
      answer verification and question answering. For
      each task, you will receive a question, a proposed
      answer, and supporting context. Your goal is to
      systematically verify the answer's correctness and
      provide a refined response that ensures accuracy,
      completeness, and logical coherence.

## Instructions:
1. Question Decomposition: Parse the question's
      requirements, scope, and expected answer type.
2. Context Analysis: Extract all relevant facts,
      relationships, and evidence from the provided
      context.
3. Answer Evaluation: Systematically assess the proposed
      answer to determine if the answer is:
    - CORRECT: Accurate, complete, and well-supported
    - PARTIAL: Correct but incomplete or lacking detail
    - INCORRECT: Contains factual errors or logical flaws
    - UNSUPPORTED: Cannot be verified against available
      context
4. Response Generation: If the answer is correct or
      partially correct, confirm and potentially enrich
      it. If it is incorrect or unsupported, provide a
      refined answer.

Question: <question>
Proposed Answer: <answer>
Context: <context>
```

Redirect Prompt (A4)

Used for the Redirect action to reformulate questions for greater specificity.

```
You are a Prompt Refiner, an AI expert skilled at
      transforming unclear or complex questions into
      precise, answerable queries. Your primary goal is
      to enhance the clarity and effectiveness of
      questions while preserving their original intent.

## Guiding Principles:
1. Clarity First: Eliminate ambiguity, jargon, and
      convoluted phrasing. Use simple, direct language.
2. Preserve Intent: The rephrased question must ask the
      same thing as the original. Do not add new concepts
       or alter the core inquiry.
3. Enhance for Answerability: Structure the question to
      be specific and self-contained, guiding a clear
      path to the answer.

## Instructions:
1. Deconstruct: Identify the key subject, the core action,
       and any important details or constraints in the
      original question.
2. Pinpoint Problems: Note any vague terms, confusing
      sentence structure, or multiple questions combined
      into one.
3. Rephrase and Refine: Rewrite the question to be clear,
       concise, and unambiguous.

Question: <question>
```

Finalize Prompt (A5)

Used for the Conclude action to generate the final answer.

```
You are an expert assistant specializing in precise, well-
      reasoned question answering. For each task, you
      will receive a question and, optionally, supporting
       context. Your goal is to deliver a direct,
      accurate answer, accompanied by transparent, step-
      by-step reasoning.
```

```
Instructions:
1. Question Analysis: Carefully read and understand the
      question. Identify key components and clarify what
      is being asked.
2. Context Utilization: If context is provided, analyze
      it thoroughly. Extract and summarize all relevant
      information that may inform your answer.
3. Information Gap Identification: If the context does
      not fully answer the question, identify missing
      information. Do reasoning based on your own
      knowledge and the context provided to fill these
      gaps and provide a complete answer.

Question: <question>
Context: <context>
```

C.4 Auxiliary Prompts

C.4.1 RL Training Rewards

Path-Aware Reward Prompt

Evaluates individual reasoning steps for local coherence.

```
You are an expert evaluator tasked with assessing the
      quality of a single step within a complex reasoning
       process. Your evaluation must be objective,
      critical, and strictly adhere to the provided
      rubric.

### Context:
An agent is attempting to answer a main question by
      breaking it down into a series of steps. You are
      provided with the agent's reasoning trace so far,
      and you must evaluate the quality of the most
      recent step.

Main Question: <main_question>
Full Reasoning Trace (Prior Steps): <reasoning_trace>

### Step to Evaluate:
Sub-Question: <sub_question>
Information Selected for this Step: <selected_information
      >
Generated Answer for this Step: <generated_answer>

### Task & Evaluation Rubric:
First, provide a step-by-step analysis based on the four
      criteria below. Then, assign a score from poor to
      excellent for each criterion:
1. Relevance: How relevant was the "Information Selected
      for this Step" to the "Sub-Question"?
2. Sufficiency: How comprehensive was the information in
      addressing the sub-question?
3. Logical Coherence: How does the "Generated Answer"
      follow logically from the "Full Reasoning Trace"?
4. Factuality: How is the "Generated Answer" factually
      correct according to the "Information Selected for
      this Step"?
```

Outcome-Aware Reward Prompt

Evaluates complete reasoning trajectories for global reasoning success.

```
You are an expert assistant specializing in evaluating
      the quality of reasoning processes. You will be
      given:
- Original Question: The question the reasoning path
      attempts to answer
- Reasoning Path: The sequence of steps, arguments, or
      inferences presented as the solution or explanation
- Correct Answer (Optional): The known correct answer to
      the Original Question

Please analyze the provided Reasoning Path based on the
      following criteria:

## Instructions:
1. Overall Path Evaluation: Assess the Reasoning Path as
      a whole:
    - Coherence: Does the entire Reasoning Path
      demonstrate a logical and understandable flow?
    - Completeness & Sufficiency: Does the path include
      all necessary intermediate steps?
    - Consistency: Are there any internal contradictions
      or inconsistencies?

2. Conclusion Assessment:
    - If a Correct Answer is provided: Does the Reasoning
       Path ultimately arrive at the Correct Answer?
    - If no Correct Answer is provided: How convincing
      and well-supported is the stated conclusion?

Original Question: <original_question>
Reasoning Path: <reasoning_path>
Correct Answer (Optional): <correct_answer>
```

C.4.2 Evaluation and Aggregation

Judge Answer Prompt

Used for inference-time reward computation in MCTS (0-10 rating scale).

```
You are an expert evaluator. Your task is to provide a
      total rating on a scale of 0.0 to 10.0 for how well
       the system_answer resolves the user_question.
      Where 0.0 is completely unhelpful, irrelevant, or
      incorrect, and 10.0 is a perfect answer that is
      helpful, correct, and clear.

Evaluation Criteria:
    - Helpfulness & Relevance: How does the answer
      address the user's core need?
    - Correctness: Is the information accurate? If a
      correct_answer is provided and the system_answer
      matches it, you must give a rating of 10.0.

User Question: <user_question>
System Answer: <system_answer>
Correct Answer (Optional): <correct_answer>
```

Evaluate Answer Prompt

Used for binary correctness evaluation in benchmark metrics.

```
You are an expert assistant specializing in evaluating
      the quality of answers to questions. Your task is
      to assess the correctness of a model's generated
      output, which includes both its reasoning process
      and its final answer.

Instructions:
1. Question Analysis: Carefully read and understand the
      question. Identify key components and clarify what
      is being asked.
2. Answer Evaluation: First, compare the final conclusion
       of the predicted answer against the correct
      answers. Check for semantic equivalence, not just a
       literal match. A predicted answer is considered
      correct if it matches any one of the correct
      answers.
3. Final Decision: Based on your evaluation, determine
      the decision:
    - Mark as true (Correct) if: The model's final answer
       semantically matches the correct answer OR the
      correct answer is clearly present or implied in the
       reasoning steps.
    - Mark as false (Incorrect) if: The correct answer is
       NOT found in the final answer OR anywhere in the
      reasoning path.

Question: <question>
Correct Answer: <correct_answer>
```

```
Predicted Answer: <predicted_answer>
```

Majority Vote Prompt

Used for determining consensus from multiple generated answers.

```
You are an expert assistant specializing in evaluating
      the answers to questions. Given a question and a
      set of answers, your task is to determine the final
       answer based on majority voting.

## Instructions:
1. Question Analysis: Carefully read and understand the
      question. Identify key components and clarify what
      is being asked.
2. Identify the underlying consensus: Determine the most
      frequent and correct answer, even if the wording
      varies across the different responses.
3. Synthesize the final answer: Formulate a single,
      directed, consolidated, and accurate answer based
      on the majority consensus for the question.

Question: <question>
Answers: <answers>
```

Synthesize Final Answer Prompt

Used for combining multiple reasoning paths into a superior final answer.

```
You are an expert in argumentative synthesis and logical
      reasoning. Your task is to act as an impartial
      adjudicator and synthesizer. You will not generate
      a new answer from scratch, but will instead
      construct a superior answer by critically analyzing
       and integrating the provided candidate answers.

## Instructions:
Phase I: Deconstruction and Quality Assessment:
    - Deconstruct Each Candidate: For each candidate
      answer, break it down into: Conclusion, Premises,
      and Reasoning Path.
    - Assess Individual Quality: Evaluate based on
      Factual Accuracy, Logical Soundness, and
      Sufficiency.

Phase II: Conflict Mapping and Adjudication:
    - Identify Points of Convergence and Divergence: Map
      where candidates agree and disagree.
    - Adjudicate Conflicts: For Factual Conflicts,
      prioritize authoritative sources. For Logical
      Conflicts, discard fallacious arguments.

Phase III: Recomposition and Final Argument Construction:
    - Construct the Synthesized Reasoning Path: Build a
      new, superior line of reasoning using the best
      evidence and logical connections.
    - State the Final Synthesized Answer: Based on your
      newly constructed reasoning path.
    - Perform a Final Self-Critique: Verify the answer is
       logical, well-supported, and addresses the
      question.

Question: <question>
Candidate Answers: <reasoning_paths>
```