

# MCTS-RAG: Enhancing Retrieval-Augmented Generation with Monte Carlo Tree Search

Anonymous EMNLP submission

## Abstract

We introduce MCTS-RAG, a novel approach that enhances the reasoning capabilities of small language models on knowledge-intensive tasks by leveraging retrieval-augmented generation (RAG) to provide relevant context and Monte Carlo Tree Search (MCTS) to refine reasoning paths. MCTS-RAG dynamically integrates retrieval and reasoning through an iterative decision-making process. Unlike standard RAG methods, which typically retrieve information independently from reasoning and thus integrate knowledge suboptimally, or conventional MCTS reasoning, which depends solely on internal model knowledge without external facts, MCTS-RAG combines structured reasoning with adaptive retrieval. This integrated approach enhances decision-making, reduces hallucinations, and ensures improved factual accuracy and response consistency. The experimental results on multiple reasoning and knowledge-intensive datasets (*i.e.*, ComplexWebQA, GPQA, and FoolMeTwice) show that our method enables small-scale LMs to achieve performance comparable to frontier LLMs like GPT-4o by effectively scaling inference-time compute, setting a new standard for reasoning in small-scale models.

## 1 Introduction

Recent advancements in MCTS-based reasoning have demonstrated remarkable improvements in structured decision-making and logical inference (Kocsis and Szepesvári, 2006; Browne et al., 2012; Xie et al., 2024a). The rStar framework (Qi et al., 2024), for instance, has shown that systematic search and exploration can significantly enhance reasoning performance, enabling small-scale LMs (*i.e.*, models with up to 7B parameters) to compete with much larger models. However, a key limitation of these approaches is their heavy reliance on internal knowledge, which hinders their effectiveness in knowledge-intensive tasks.

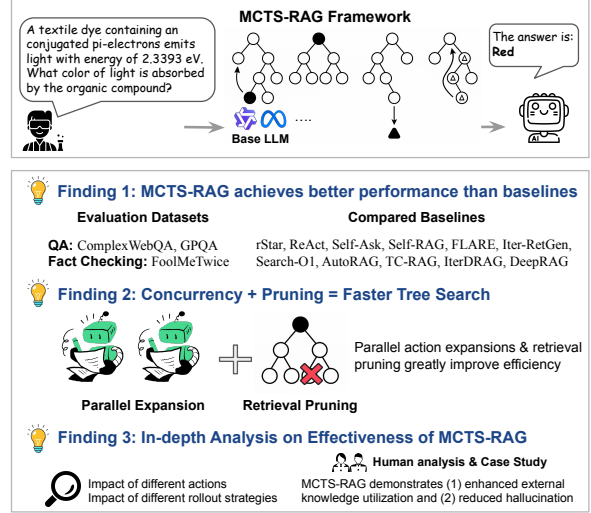


Figure 1: Overview of the research. The top panel illustrates the proposed MCTS-RAG framework, while the bottom panel summarizes three key findings from our experiments and analysis.

On the other hand, RAG has been widely used to solve knowledge-intensive tasks (Lewis et al., 2020; Karpukhin et al., 2020; Izacard and Grave, 2021), but its effectiveness with small-scale LMs remains limited. small-scale LMs struggle with query formulation and retrieved content comprehension, often generating vague queries and misinterpreting key details (Fan et al., 2025). Moreover, existing RAG systems do not dynamically adjust their retrieval strategies based on changing informational or reasoning requirements, which results in unnecessary or repetitive retrieval steps (Li et al., 2024; Gao et al., 2024). For example, when answering a multi-hop question like “Which novel inspired the movie that won Best Picture in 1994?”, a standard retrieval system might retrieve documents about Forrest Gump (*i.e.*, Best Picture winner in 1994), but fail to recognize the need for additional reasoning or retrieval steps to establish the connection between Forrest Gump and the novel written

by Winston Groom. This limitation arises because small-scale language models often lack the ability to refine queries iteratively and integrate retrieved information into a coherent reasoning process.

To address the aforementioned limitations, we propose MCTS-RAG<sup>1</sup>, a novel framework that integrates MCTS’s reasoning and search capabilities with adaptive retrieval mechanisms. At a high level, MCTS-RAG operates by iteratively refining both retrieval and reasoning through a search-based process. Given a query, it explores multiple reasoning paths, dynamically incorporating retrieval actions at key decision points. Retrieved knowledge is then used to evaluate intermediate states, and beneficial retrieval pathways are reinforced through backpropagation. This structured search mechanism ensures that the model efficiently acquires and utilizes relevant information for more accurate reasoning. To further enhance efficiency, MCTS-RAG employs parallel expansion and retrieval pruning strategies during the search, reducing redundant computation while maintaining search quality. In contrast to prior approaches, by integrating retrieval with search-based reasoning, MCTS-RAG is able to systematically explore relevant knowledge and reason over it to obtain the correct answer.

MCTS-RAG has the following key features: **Improved reasoning accuracy:** New retrieval actions enable SLMs to acquire external knowledge and enhance the quality of question answering (§3.2). **Optimized query formulation:** The refinement process ensures that each query focuses on specific information needs, improving the effectiveness of retrieval query generation (§3.3). **Enhanced retrieval quality:** Reflecting on and summarizing retrieved information helps reduce semantic discrepancies and ensures alignment with the core problem (§3.3). **High efficiency:** Parallel expansion and retrieval pruning reduce redundant computation during search, greatly improving inference efficiency without compromising performance (§4.5).

MCTS-RAG demonstrates superior performance on various knowledge-intensive benchmarks, including ComplexWebQA (CMQA) (Talmor and Berant, 2018), GPQA (Rein et al., 2024), and FoolMeTwice (FMT) (Eisenschlos et al., 2021a). Specifically, it achieves over 20% improvement with Llama 3.1-8B and 6% with Qwen2.5-7B on CWQA, roughly 15% and 10% gains on GPQA,

and over 10% (Llama) and 4% (Qwen) on FMT, while outperforming other competitive baselines. Our efficiency analysis demonstrates that MCTS-RAG delivers the best overall trade-off compared to other baseline systems, achieving the highest accuracy with moderate latency and token cost.

## 2 Related Work

**Inference-time Scaling.** Inference-time scaling enhances reasoning without modifying model parameters by optimizing computational allocation during generation. A core approach involves reasoning diversification and selection: generating multiple candidates (Wang et al., 2023) and choosing optimal outputs via voting (Liang et al., 2024) or verifier-guided ranking (Cobbe et al., 2021). Structured search algorithms, such as beam search (Xie et al., 2024b) and tree-of-thought frameworks (Yao et al., 2023a), explicitly model reasoning paths. Recently, Monte Carlo Tree Search (MCTS) has been applied to balance exploration and exploitation in reasoning tasks, iteratively refining solutions through selection, expansion, simulation, and backpropagation (Hao et al., 2023). Further, integrating MCTS with LLMs using value functions (Zhang et al., 2024) or predefined reasoning heuristics (Qi et al., 2024) has improved efficiency in mathematical reasoning and code generation.

**Retrieval-Augmented Generation.** The RAG system enhances LLMs in knowledge-intensive tasks by incorporating external information. Query optimization techniques, including expansion and transformation, improve retrieval quality (Ma et al.; Jagerman et al., 2023). Iterative retrieval methods, such as IRCOT (Trivedi et al., 2023) and ITER-RETGEN (Shao et al., 2023), refine retrieval and generation. LLM-driven retrieval strategies, such as WebGPT (Nakano et al., 2021) and Toolformer (Schick et al., 2023), have demonstrated notable improvements in efficiency by leveraging large language models to interact with external tools or search engines, thus streamlining the process of gathering relevant data. Meanwhile, self-reflection mechanisms in systems like Self-RAG (Asai et al.; Islam et al., 2024) and Auto-RAG (Yu et al., 2024) further enhance retrieval relevance by employing iterative introspection to refine intermediate outputs. More recently, ReARTeR (Sun et al., 2025) improves retrieval by reasoning over multiple answer candidates with mutual evaluation. However, its sequential processing and full candidate scoring

<sup>1</sup>Our code is available at <https://anonymous.4open.science/r/MCTS-RAG-AD04>, and will be released publicly upon publication.

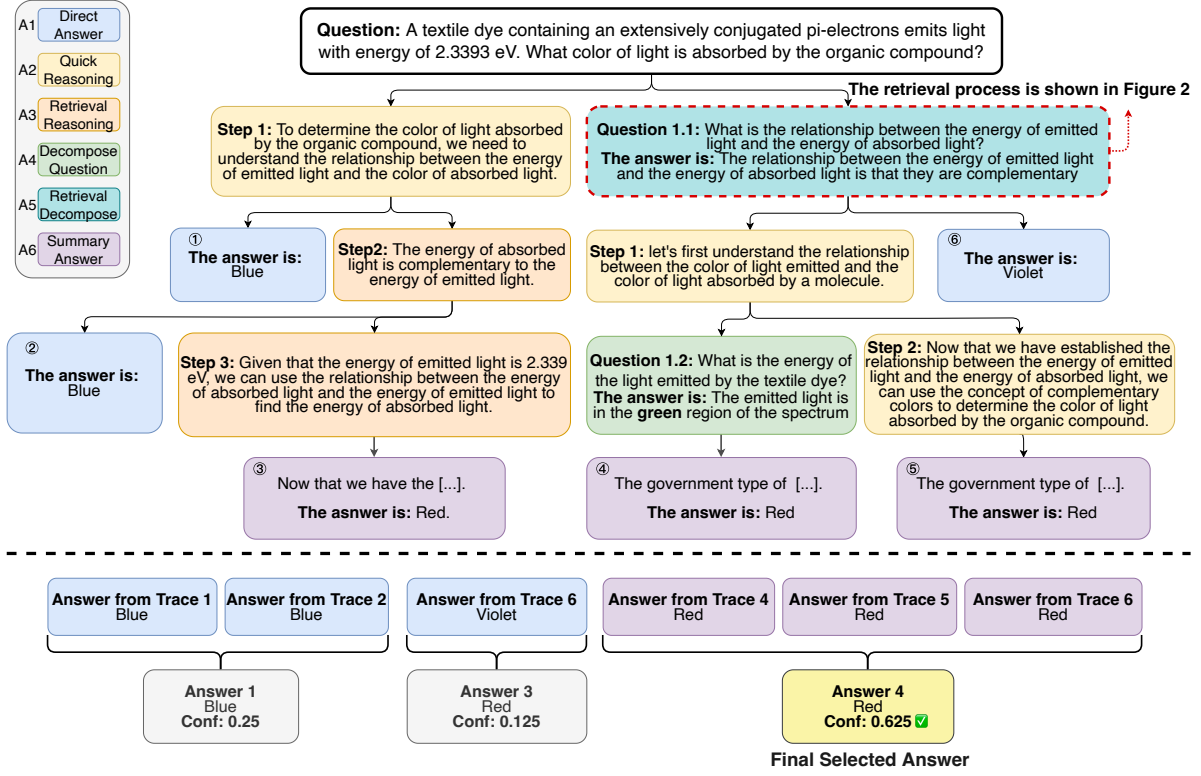


Figure 2: An illustration of MCTS-RAG workflow for answering the question sampled from ComplexWebQA.

lead to high latency. RAG-Star (Jiang et al., 2024a) uses a fixed search tree for token-level retrieval control, but lacks dynamic pruning and does not support concurrent reasoning paths. In contrast, MCTS-RAG supports parallel expansion of diverse reasoning strategies and incorporates lightweight retrieval pruning to skip unnecessary external calls. This leads to better efficiency without sacrificing answer quality.

### 3 MCTS-RAG

#### 3.1 Preliminaries

rStar (Qi et al., 2024) is a recently proposed self-consistency framework designed to enhance the reasoning capabilities of language models without requiring additional fine-tuning or reliance on stronger teacher models. rStar achieves this by breaking down the reasoning process into two distinct yet interconnected phases: *generation* and *discrimination*. In the **Generation Phase**, the model proactively explores multiple reasoning trajectories through human-like reasoning actions, including step-by-step inference and question decomposition. Subsequently, the **Discrimination Phase** evaluates these candidate reasoning paths, selecting and refining them to identify the most logically consistent

and accurate responses.

However, the original rStar framework is limited by its inability to dynamically acquire external knowledge, restricting its performance in knowledge-intensive queries. In addition, it suffers from significant latency, often requiring  $4-5\times$  the inference time compared to Standard RAG methods. To address the inherent limitations of rStar, we propose an integrated reasoning framework that combines the iterative reasoning capabilities of rStar with RAG. At a high level, our approach builds on the iterative generative-discriminative structure of rStar and introduces additional operations specifically designed to facilitate dynamic external knowledge retrieval. This enables the language model to seamlessly integrate relevant external information into its reasoning process, significantly improving factual accuracy and decision robustness. The following subsections detail the proposed MCTS-RAG framework.

#### 3.2 Action Space Definition

We design a set of discrete actions at each MCTS decision point: A1–A3 from rStar (Qi et al., 2024), along with two new RAG-related actions A4 and A5 and a summary action A6, enabling dynamic knowledge acquisition and enhanced reasoning syn-

ergy for improved decision-making.

**A1: Direct Answer:** Provide an immediate response based on existing reasoning or previously known context, suitable for straightforward queries or when additional analysis is unnecessary.

**A2: Quick Reasoning:** Execute rapid, incremental reasoning steps based on the current context, ideal for exploratory paths or preliminary judgments to efficiently guide the search.

**A3: Decompose Question:** Break complex queries into smaller, manageable sub-questions, allowing for clearer problem-solving pathways and improved reasoning efficiency, particularly beneficial for multi-part or intricate problems.

**A4: Retrieval Reasoning:** Actively retrieve relevant knowledge from internal or external sources before proceeding with the next reasoning step, critical for queries requiring supplementary information or when existing context is incomplete.

**A5: Retrieval Decompose:** Integrate both decomposition and retrieval, first breaking down complex questions and then acquiring relevant knowledge to solve individual sub-problems. This action is highly effective for queries involving detailed context-dependent sub-questions.

**A6: Summarized Answer:** Generate a concise, structured summary that synthesizes results from previous reasoning and retrieved information, providing coherent and comprehensive responses especially useful for queries that demand summarization or integration of multifaceted information.

Algorithm 1 illustrates the procedure for updating each action’s reward. To further enhance exploration, we employ Upper Confidence Bound for Trees (UCT) (Kocsis and Szepesvári, 2006) in our MCTS framework—a crucial method that balances exploitation and exploration. The UCT formula is:

$$\text{UCT}(s, a) = \bar{Q}(s, a) + C \cdot \sqrt{\frac{\ln N(s)}{N(s, a)}}$$

#### Algorithm 1 $R(s, a)$ Computation and Update

---

**Require:** State  $s$ , action  $a$ , completions  $\{o_1, \dots, o_K\}$ , log-likelihoods  $\{\ell_1, \dots, \ell_K\}$   
**Ensure:** Representative answer  $o^*$ , confidence  $\text{Conf}(o^*)$ , reward  $R(s, a)$

```

1: Initialize empty clusters  $\mathcal{C} \leftarrow \emptyset$ 
2: for  $j = 1$  to  $K$  do
3:    $o_j \leftarrow j$ -th completion
4:    $\text{matched} \leftarrow \text{False}$ 
5:   for each representative  $r$  in  $\mathcal{C}$  do
6:     if  $\text{Equiv}(o_j, r)$  then
7:       Add  $o_j$  to cluster  $\mathcal{C}_r$ 
8:        $\text{matched} \leftarrow \text{True}$ 
9:       break
10:    end if
11:  end for
12:  if not  $\text{matched}$  then
13:    Create new cluster  $\mathcal{C}_{o_j} \leftarrow \{o_j\}$ 
14:  end if
15: end for
16: Let  $\mathcal{O}^* \leftarrow$  majority cluster with size  $n^*$ 
17: Select representative  $o^* \in \mathcal{O}^*$ 
18:  $\text{Conf}(o^*) \leftarrow \frac{n^*}{K}$ 
19:  $R(s, a) \leftarrow \frac{1}{n^*} \sum_{o_j \in \mathcal{O}^*} \ell_j$ 
20:  $Q(s, a) \leftarrow Q(s, a) + R(s, a)$ 
21:  $N(s, a) \leftarrow N(s, a) + 1$ 
    return  $o^*, \text{Conf}(o^*), R(s, a)$ 

```

---

where  $\bar{Q}(s, a) = \frac{Q(s, a)}{N(s, a)}$  is the average reward for action  $a$  in state  $s$ , with  $Q(s, a)$  as the cumulative reward and  $N(s, a)$  as the visit count.  $N(s)$  is the total number of visits to state  $s$ .  $C$  is the exploration constant, controlling the balance between exploitation and exploration.

Within MCTS-RAG, search depth limits how many levels are expanded from the root node to control the search range, while the number of roll-outs indicates how many times the simulation is run from a selected node until termination or a preset limit to estimate its value. By running simulations within a controlled depth and updating node statistics via UCT, MCTS effectively balances exploration and exploitation with finite computational resources, continuously refining its search strategy.

### 3.3 Retrieval Process

Our approach dynamically retrieves information within an evolving MCTS reasoning environment, enabling timely and relevant integration of external knowledge. The model autonomously determines when retrieval is required, generates targeted queries, and critically integrates external knowledge to improve reasoning accuracy. By interweaving retrieval with reasoning, we streamline information flow and produce concise yet informative outputs. If previously retrieved data adequately answers the current reasoning step—determined by



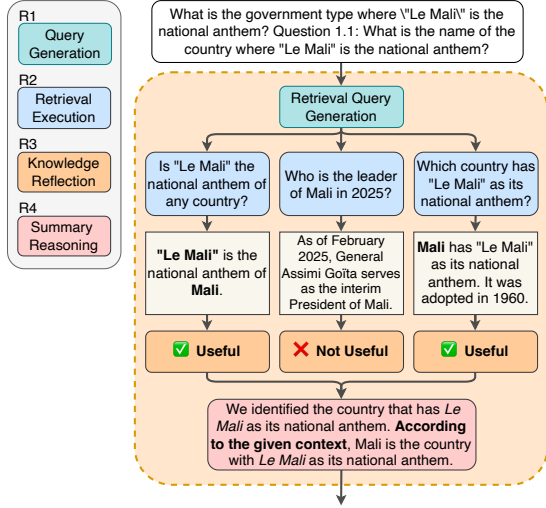


Figure 3: An illustration of MCTS-RAG retrieval process (*i.e.*, R1-R4) within one step of the retrieval decomposition action highlighted in Figure 3.

checking whether the information satisfies predefined accuracy thresholds or resolves open reasoning paths—the model foregoes additional retrieval, thus avoiding redundancy.

**R1: Query Generation:** If a knowledge gap is detected, the model generates search queries.

**R2: Query Execution:** External retrieval tools are used to obtain the most relevant information.

**R3: Knowledge Reflection:** Retrieved data is evaluated for relevance and consistency to determine its inclusion in the reasoning process.

**R4: Summary Answer:** Refined information is integrated, enabling the model to answer sub-questions or advance reasoning.

This interleaved retrieval process ensures that the model’s reasoning is continuously updated and validated against external data, thereby reducing errors and enhancing the robustness of final output.

### 3.4 Determining Final Answer

At the conclusion of the MCTS exploration (illustrated in the bottom part of Figure 3), the best answer is selected through a voting mechanism and consistency analysis over candidate solutions. Specifically, each reasoning trajectory obtained from the MCTS yields a candidate answer  $c_j$ , resulting in a candidate answer set  $\mathcal{C} = \{c_1, c_2, \dots, c_M\}$ . These candidate answers are grouped into a set of unique answers

$\mathcal{A} = \{a_1, a_2, \dots, a_N\}$  based on semantic consistency. The final score for each unique answer  $a_k$  is computed as the sum of the rewards of all candidates grouped under  $a_k$ , where the reward of each candidate  $c_j$  is the product of rewards for all nodes along its corresponding reasoning trajectory.

$$\text{Score}(a_k) = \frac{\sum_{c_j \in \mathcal{C}(a_k)} \text{Reward}(c_j)}{\sum_{c_j \in \mathcal{C}} \text{Reward}(c_j)} \quad (1)$$

The best answer is then determined as

$$a^* = \arg \max_{a_k \in \mathcal{A}} \text{Score}(a_k), \quad (2)$$

ensuring that the most frequent and consistent reasoning trajectory is chosen. Essentially, our approach operates as a reward-accumulation voting scheme: by normalizing and aggregating the trajectory rewards of semantically consistent candidate clusters, it guarantees that the selected answer is both the most coherent and the highest-scoring.

## 4 Experiment Setup

### 4.1 Evaluation Benchmark

We evaluate MCTS-RAG and other competitive baseline systems on three complex reasoning tasks: (1) **ComplexWebQA (CWQA)** (Talmor and Berant, 2018), which requires multi-step reasoning over web-based queries; (2) **GPQA** (Rein et al., 2023), which tests knowledge-intensive science question answering; and (3) **FoolMeTwice (FMT)** (Eisenschlos et al., 2021b), a challenging fact-checking benchmark that assesses the model’s ability to verify factual claims.

### 4.2 Baseline Systems

Beyond **CoT prompting**, **rStar**, and **Standard RAG**, we also compare MCTS-RAG with several recent RAG variants, including: **ReAct** (Yao et al., 2023b) alternates between reasoning and retrieval, allowing the model to dynamically refine its understanding based on external evidence. **Self-Ask with Search (Self-Ask)** (Press et al., 2023) with Search decomposes complex queries into subquestions, retrieves relevant external information, and synthesizes the answers to enhance multi-step reasoning. **Search-O1** (Li et al., 2025) executes a single retrieval step before generating an answer, limiting its ability to iteratively verify information. **Self-RAG** (Asai et al., 2024) extends standard RAG by generating and issuing its own retrieval

queries at each decoding step, enabling deeper multi-hop evidence gathering. **FLARE** (Jiang et al., 2023) employs an active learning strategy to select the most informative passages for retrieval, improving answer relevance with fewer retrieval calls. **Iter-RetGen** (Shao et al., 2023) alternates retrieval and generation phases in multiple passes, refining responses through iterative editing. **AutoRAG** (Kim et al., 2024) automates retrieval pipeline configuration and model selection to optimize end-to-end retrieval-augmented generation performance. **TC-RAG** (Jiang et al., 2024b) integrates Turing-complete control flows into RAG, supporting complex multi-step reasoning via case-based retrieval loops. **IterDRAG** (Yue et al.) introduces dynamic retrieval-generation loops that adapt retrieval strategies based on intermediate model outputs. **DeepRAG** (Guan et al., 2025) interleaves deep reasoning modules between retrieval steps to enhance multi-hop inference capabilities. **ReARTEr** (Sun et al., 2025) uses a reward model and explanations to improve step-by-step reasoning with MCTS. **RAG-Star** (Jiang et al., 2024a) combines MCTS planning with external retrieval to guide and verify multi-step reasoning. To ensure fair comparisons, we use the same base LLMs—Qwen2.5-7B and Llama 3.1-8B—for all the evaluated systems.

### 4.3 Implementation Details

**RAG Setup.** To maintain consistency across methods, we use a shared retrieval corpus and identical retriever configurations. We employ the Bing Search Engine and LangChain for retrieval, with Bing offering extensive and up-to-date web information and LangChain providing modular support for retrieval-augmented generation workflows. For the CWQA dataset, we collect approximately 100K web snippets retrieved via Bing using question templates; these snippets are dynamically retrieved at inference time based on the input question. For GPQA, we construct a static corpus comprising 80K passages sampled from Wikipedia and 60K web documents retrieved from Bing, totaling 140K documents. The retriever encodes questions and selects top-ranked documents from this hybrid source. For FMT, a domain-specific dataset, we use its original associated documents (about 30K passages) provided as part of the benchmark and treat them as the retrieval pool. In all cases, top-10 retrieved documents are fed into the reasoning module for answer generation and verification. This setup en-

Methods	Qwen2.5-7B			Llama 3.1-8B		
	CWQA	GPQA	FMT	CWQA	GPQA	FMT
CoT	34.6	35.0	57.3	27.7	28.7	56.5
GPT-4o	54.4	53.0	55.4	54.5	53.0	55.4
Qwen2.5-72B	44.5	40.6	58.4	44.6	40.6	58.4
rStar	55.4	32.3	55.9	37.6	28.7	56.4
Standard RAG	44.2	40.6	58.4	35.6	31.7	51.5
GPT-4o	59.4	54.9	61.4	59.4	54.9	61.4
Qwen2.5-72B	48.5	43.1	59.4	48.8	46.2	59.8
ReAct	45.5	41.6	62.4	47.5	49.3	55.4
Self-Ask	44.6	42.6	60.9	44.6	52.8	58.4
Self-RAG	46.2	43.1	61.1	47.1	53.9	60.2
FLARE	50.1	45.3	62.6	50.1	56.6	62.1
ReARTEr	51.8	46.4	63.3	51.4	57.1	64.3
Iter-RetGen	52.2	47.5	63.1	52.3	57.6	63.2
Search-O1	49.5	54.5	64.4	54.6	58.8	65.9
AutoRAG	57.2	55.1	66.1	57.2	59.3	66.9
TC-RAG	57.7	55.5	66.7	59.6	62.3	68.7
RAG-Star	58.1	57.9	67.4	60.1	66.3	69.8
IterDRAG	59.3	58.2	67.1	61.1	67.2	71.1
DeepRAG	60.9	61.3	66.9	62.3	69.8	72.9
<b>MCTS-RAG</b>	<b>61.4</b>	<b>64.6</b>	<b>68.3</b>	<b>67.3</b>	<b>71.3</b>	<b>73.8</b>

Table 1: Answer accuracy of MCTS-RAG and other methods (with and without retrieval modules).

sures that variations in performance stem from differences in reasoning mechanisms.

**MCTS-RAG Setup.** To facilitate structured reasoning, we configure our setup with a rollout of 4, allowing multiple steps of reasoning expansion. Each query can be decomposed into at most two subquestions, ensuring a controlled breakdown of complex queries. We set the maximum reasoning depth to 5, enabling deep but efficient multi-hop reasoning. Moreover, in order to reduce latency during tree expansion, we implement concurrent action evaluations: different reasoning actions at a given search node are expanded in parallel. This design leverages asynchronous computation to significantly speed up MCTS traversal without sacrificing the fidelity of action-value estimation.

### 4.4 Main Findings

Table 1 compares reasoning methods on CWQA, GPQA, and FMT for Llama 3.1-8B and Qwen2.5-7B. Our approach consistently outperforms baselines, demonstrating strong multi-step reasoning and retrieval capabilities. On CWQA, it achieves over a 20% gain with Llama 3.1-8B and around 6% with Qwen2.5-7B. Similarly, it surpasses competitors on GPQA by roughly 42% and 32%, respectively, benefiting from refined verification strategies. On FMT, it leads by over 17% with Llama

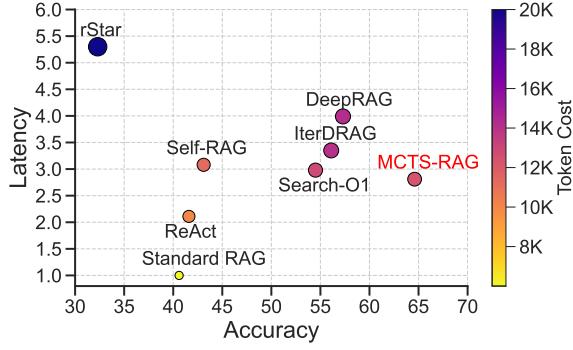


Figure 4: Performance comparison of reasoning methods on GPQA using Qwen2.5-7B. The horizontal axis shows accuracy, the vertical axis shows relative latency the decoding time relative to the Standard RAG baseline, and marker size and color correspond to average token generated.

3.1-8B and 12% with Qwen2.5-7B, proving its resilience against misleading distractors. These results highlight our method’s superior generalization and efficiency, especially in fact-checking and science-related tasks. Compared to baselines like Standard RAG, ReAct, Self-Ask, Search-O1, TC-RAG, IterDRAG, and DeepRAG, our structured multi-step reasoning can retrieve and process evidence more accurately, and on average we improve the performance by about 14% over the baseline under three datasets. Compared to rStar, MCTS-RAG enables broader retrieval, extracting critical insights while minimizing hallucinations, achieving an average improvement of 17%.

#### 4.5 Efficiency Analysis

As shown in Figure 4, serves as the baseline with normalized latency (1.0 $\times$ ) but also the lowest accuracy (40.6%). ReAct improves accuracy to 42.6% at the cost of 2.0 $\times$  latency and slightly lower token consumption (8,884). Self-RAG further increases accuracy to 43.1% with 2.8 $\times$  latency and 11,000 tokens. Search-O1 balances accuracy (54.5%) and latency (2.8 $\times$ ) with moderate token cost (11,300), making it suitable for latency-sensitive scenarios. IterDRAG (56.1%, 3.3 $\times$ , 12,500 tokens) and DeepRAG (57.3%, 4.0 $\times$ , 13,700 tokens) trade additional latency for marginal accuracy gains. rStar performs poorly (32.3%, 5.5 $\times$ , 19,000 tokens), indicating low efficiency. **MCTS-RAG delivers the best overall trade-off, achieving the highest accuracy (64.6%) with moderate latency (2.8 $\times$ ) and token cost (11,892).**

Table 2 presents the token cost and relative la-

tency for Qwen2.5-7B under various retrieval and rollout configurations. While MCTS-RAG consumes more tokens per query than the Standard RAG baseline (11,892 vs. 9,993 tokens in the “Enable All” vs. “Disable A4” settings), it achieves significantly better latency efficiency, running only 2.81 $\times$  slower than RAG despite a 19% increase in token cost. In contrast, rStar’s latency grows nearly linearly with rollout depth, since it cannot parallelize across multiple reasoning branches.

This improvement stems from two key optimizations in MCTS-RAG: (1) **Parallel Expansion of Reasoning Actions:** during each rollout step, candidate subquestions (“actions”) are generated and verified in parallel, rather than sequentially as in rStar. This concurrency amortizes the overhead of verification across multiple branches, yielding higher token throughput per unit time. (2) **Dynamic Pruning of Unnecessary Retrievals:** Our framework allows the model to autonomously determine whether a retrieval step is needed. If the current context suffices, the model skips retrieval entirely. This implicit gating mechanism reduces redundant queries without introducing additional pruning modules. In contrast to rStar’s costly answer-level evaluations, our approach streamlines inference by eliminating unnecessary retrievals based solely on the model’s internal confidence.

As shown in the rollout analysis, increasing the rollout number from 4 to 16 raises token cost from 11,892 to 28,972, but latency only increases from 2.8 $\times$  to 4.5 $\times$  relative to Standard RAG. This sublinear latency growth confirms that our concurrent action expansion and adaptive branch pruning jointly deliver a favorable trade-off: modest increases in token consumption yield diminishing increments in inference time, substantially outperforming both Standard RAG and rStar in overall efficiency.

#### 4.6 Fine-grained Analysis

We evaluate the effectiveness of retrieval actions and rollout times in Table 2. Specifically, we conduct an ablation by disabling different retrieval modules (A4, A5, or both) to gauge their impact on overall performance. In addition, we vary the number of rollouts from 4 to 16 to investigate how deeper search affects accuracy and efficiency.

**Impact of Different Actions.** Retrieval actions, especially A4 and A5, are key for multi-step reasoning. Enabling all retrievals boosts GPQA (+32.3%) and FMT (+17.9%). Disabling A5 improves GPQA

Settings	CWQA	GPQA	FMT	Token	Latency
<i>Analysis of Retrieval Modules</i>					
Disable A1	61.3	64.6	68.1	11300	2.7×
Disable A2	60.5	63.5	67.5	15460	3.1×
Disable A3	60.7	63.9	67.6	16050	3.3×
Disable A4&A5	55.5	32.3	50.4	8884	2.4×
Disable A4	55.7	36.3	55.9	9993	2.6×
Disable A5	56.2	44.1	62.4	9714	2.5×
Enable All	61.4	64.6	68.3	11892	2.8×
<i>Analysis of Rollout Numbers</i>					
4 rollout	61.4	64.6	68.3	11892	2.8×
8 rollout	64.4	63.7	68.1	16963	3.2×
12 rollout	68.7	75.2	69.4	21860	3.9×
16 rollout	71.2	84.3	74.1	28972	4.5×

Table 2: Accuracy of Qwen2.5-7B-Instruct under various retrieval and rollout configurations. *Token* = average tokens generated; *Latency* = relative decoding time vs. Standard RAG.

(+11.8%) and FMT (+12.0%) over disabling A4, suggesting A4’s stronger role. CWQA sees minimal impact (+5.9%). These findings highlight retrieval trade-offs and the importance of recursive evidence aggregation.

**Impact of Different Rollout Strategies.** More rollouts enhance performance. Specifically, increasing from 4 to 8 slightly aids CWQA (+3.0%), while 8 to 12 boosts GPQA (+11.5%). Scaling to 16 further improves GPQA (+9.1%) and FMT (+4.7%), showing the value of iterative reasoning.

## 4.7 Human Analysis and Case Study

To better understand the strengths and limitations of MCTS-RAG, we conduct a comprehensive analysis of its successful cases in comparison to baseline methods, along with a thorough error analysis.

**Successful Case Analysis.** Our case study reveals the following two key improvements introduced by MCTS-RAG: (1) **Enhanced External Knowledge Utilization:** Compared to other reasoning methods, MCTS-RAG achieves higher accuracy, primarily due to its richer reasoning space and more effective utilization of external knowledge. Figure 8 clearly illustrates how Monte Carlo Tree Search tightly integrates reasoning and retrieval processes, significantly enhancing the quality and richness of information used during reasoning, thereby substantially improving inference accuracy. (2) **Reduced Hallucination Risks:** More-

over, MCTS-RAG mitigates hallucination risks through detailed and explicit reasoning steps. On one hand, the explicit reasoning pathways enable the model to more accurately interpret retrieved external knowledge, reducing errors arising from ambiguity or misunderstanding (as illustrated in Figure 9 in Appendix). On the other hand, these thorough reasoning procedures generate clearer and more contextually relevant queries, thus improving the precision of external information retrieval (as illustrated in Figure 10 in Appendix). Consequently, MCTS-RAG demonstrates substantial advantages over traditional reasoning methods in terms of improved accuracy and robustness.

**Error Case Analysis.** Our human analysis identifies the following three primary error types in MCTS-RAG: (1) **Amplification Error:** As illustrated in Figure 5, early retrieval errors in MCTS-RAG can be magnified, causing incorrect information to dominate subsequent reasoning and ultimately leading to a incorrect final answer. (2) **Factual Confusion:** We reveal that semantic mismatches between retrieved text and the reasoning process can lead to conflation or hallucinations. Figure 6 presents details on how semantically divergent retrieval results can lead to incorrect final answers. (3) **Information Overload:** Excessive additional information in MCTS-RAG can cause certain reasoning paths to deviate from the original question, leading to incorrect conclusions. Figure 7 presents a detailed example of some reasoning paths that prioritize irrelevant aspects.

## 5 Conclusion

The work introduces MCTS-RAG, an approach that integrates MCTS with RAG to improve multi-step reasoning accuracy and reliability. MCTS-RAG not only enables flexible formulation of high-quality retrieval queries but also refines the reasoning path through iterative tree exploration, thus reducing hallucinations caused by shallow retrieval or simplistic reasoning. To further enhance practicality, we introduce several efficiency optimizations, including parallel expansion and retrieval pruning, significantly reducing inference latency without compromising accuracy. Experimental results demonstrate that MCTS-RAG achieves strong performance on complex reasoning, knowledge-enhanced scientific QA, and fact-checking tasks.



## Limitations and Future Work

MCTS-RAG integrates MCTS-based reasoning and RAG to enhance reasoning capabilities, but several errors persist. Amplification errors occur when early retrieval mistakes propagate through search iterations. Factual confusion arises from semantic mismatches leading to incorrect reasoning. Information overload happens when excessive retrieval results cause reasoning to deviate from the target. Additionally, search latency remains a challenge, as deep MCTS search trees significantly increase reasoning time, particularly with multiple retrieval steps. Action selection complexity arises because the optimal choice among A1-A6 depends on query difficulty, necessitating a more adaptive decision mechanism. Inefficient expansion occurs when MCTS explores unnecessary branches due to a lack of effective pruning based on retrieval confidence or early error detection. Addressing these issues is essential for improving efficiency and reasoning accuracy.

We encourage future work to focus on optimizing search efficiency by developing adaptive action selection strategies, confidence-based retrieval filtering, and error-aware pruning mechanisms to improve MCTS exploration. Additionally, integrating reinforcement learning for dynamic search policy refinement may further enhance reasoning accuracy. Addressing these challenges will contribute to the development of more robust and scalable reasoning models, bridging the gap between retrieval-based methods and human-like problem-solving.

## References

- Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*.
- Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. [Self-RAG: Learning to retrieve, generate, and critique through self-reflection](#). In *The Twelfth International Conference on Learning Representations*.
- Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Bohnlshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43.

- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. [Training verifiers to solve math word problems](#).
- Julian Eisenschlos, Bhuwan Dhingra, Jannis Bulian, Benjamin Börschinger, and Jordan Boyd-Graber. 2021a. [Fool me twice: Entailment from Wikipedia gamification](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 352–365, Online. Association for Computational Linguistics.
- Julian Eisenschlos, Bhuwan Dhingra, Jannis Bulian, Benjamin Börschinger, and Jordan Boyd-Graber. 2021b. [Fool me twice: Entailment from wikipedia gamification](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.
- Tianyu Fan, Jingyuan Wang, Xubin Ren, and Chao Huang. 2025. [Minirag: Towards extremely simple retrieval-augmented generation](#).
- Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. [Retrieval-augmented generation for large language models: A survey](#).
- Xinyan Guan, Jiali Zeng, Fandong Meng, Chunlei Xin, Yaojie Lu, Hongyu Lin, Xianpei Han, Le Sun, and Jie Zhou. 2025. [Deeprag: Thinking to retrieval step by step for large language models](#).
- Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. 2023. [Reasoning with language model is planning with world model](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8154–8173, Singapore. Association for Computational Linguistics.
- Shayekh Islam, Md Asib Rahman, KSM Tozammel Hosain, Enamul Hoque, Shafiq Joty, and Md Rizwan Parvez. 2024. [Open-rag: Enhanced retrieval augmented reasoning with open-source large language models](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 14231–14244.
- Gautier Izacard and Edouard Grave. 2021. [Leveraging passage retrieval with generative models for open domain question answering](#). In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, Online. Association for Computational Linguistics.
- Rolf Jagerman, Honglei Zhuang, Zhen Qin, Xuanhui Wang, and Michael Bendersky. 2023. [Query expansion by prompting large language models](#). *arXiv preprint arXiv:2305.03653*.



- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. [Self-consistency improves chain of thought reasoning in language models](#). In *The Eleventh International Conference on Learning Representations*.
- Yuxi Xie, Anirudh Goyal, Wenyue Zheng, Min-Yen Kan, Timothy P Lillicrap, Kenji Kawaguchi, and Michael Shieh. 2024a. Monte carlo tree search boosts reasoning via iterative preference learning. *arXiv preprint arXiv:2405.00451*.
- Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, James Xu Zhao, Min-Yen Kan, Junxian He, and Michael Xie. 2024b. Self-evaluation guided beam search for reasoning. *Advances in Neural Information Processing Systems*, 36.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: deliberate problem solving with large language models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 11809–11822.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.
- Tian Yu, Shaolei Zhang, and Yang Feng. 2024. Auto-rag: Autonomous retrieval-augmented generation for large language models. *arXiv preprint arXiv:2411.19443*.
- Zhenrui Yue, Honglei Zhuang, Aijun Bai, Kai Hui, Rolf Jagerman, Hansi Zeng, Zhen Qin, Dong Wang, Xuanhui Wang, and Michael Bendersky. Inference scaling for long-context retrieval augmented generation. In *The Thirteenth International Conference on Learning Representations*.
- Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024. Rest-mcts\*: Llm self-training via process reward guided tree search. *arXiv preprint arXiv:2406.03816*.

## A Prompts for Each Action

### A1 (Direct Response)

#### Template:

A chat between a curious user and an AI assistant. The assistant gives step-by-step solutions to the user's questions. In the end of the assistant's response, a final answer must be given in the format of "The answer is: <ANSWER>.", where <ANSWER> should be a concise answer.

#### Usage Example:

{examples}

#### Instruction:

{instruction}

#### Note:

Please answer in a complete sentence.

---

### A2 (One-Step Reasoning)

#### Template:

A chat between a curious user and an AI assistant. The assistant gives step-by-step solutions to the user's questions with each step numbered. At the final step, a conclusive answer must be given in the format "The answer is: <ANSWER>.", where <ANSWER> should be a concise answer.

#### Instruction:

{instruction}

#### Note:

Let's think step by step.

---

### A3 (Decompose Answer)

#### Template:

Given a question, decompose it into sub-questions. For each sub-question, provide an answer in one complete sentence ending with "The answer is ". When the original question is answerable, start the sub-question with "Now we can answer the question: <original question>".

---

### A4 (Transform Retrieve Query)

#### Template:

Given a question, generate a search query that would help gather information to answer it. Your goal is to formulate a query that retrieves useful evidence or additional details relevant to the question. The query should be specific enough to ensure that the search results are both relevant and helpful. Please answer in one complete sentence, starting with "The query is: <your retrieve query>".

#### Question:

{question}

---



## A5 (Reflect Retrieved Knowledge)

### Template:

A chat between a curious user and an AI assistant. The assistant evaluates whether the retrieved information is relevant to the search query and sufficient to answer the question. Please provide a concise evaluation in one complete sentence, starting with "Evaluation:".

### Instruction:

Please assess if the retrieved information is related to the query and can be used to answer the question.

---

## A6 (Summarize Answers)

### Template:

Analyze the provided **Knowledge** and extract key information relevant to the **Original Question**. Present your analysis in a concise and organized format.

### Input:

- **Original Question:** {original\_question}
- **Knowledge:** {retrieved\_context}

### Output Format:

Key Points: Point 1: Relevant information; Point 2: Relevant information; Point 3: Relevant information...

### Requirement:

The output must be a single line summarizing all key points in one sentence without redundant description.

B Error Analysis

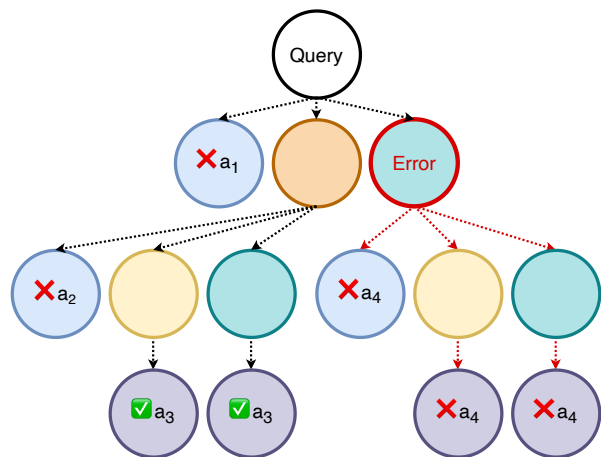


Figure 5: An illustration of MCTS Amplification Error. Early MCTS retrieval errors amplify mistakes, leading to a final answer favoring incorrect paths.

**Examples Illustrating Factual Confusion**

**Question:**  
Which U.S. President was in office when the first man landed on the Moon?

**Retrieve Query:**  
Who was the U.S. President when the first Moon landing happened?

**Retrieved documents:**  
John F. Kennedy’s famous 1962 speech about going to the Moon.  
Richard Nixon’s recorded phone call with Neil Armstrong during the landing(1969).

**Summary Reasoning:**  
*(The system assumes that because Kennedy set the goal, he was also the President when the landing occurred.)*  
Kennedy set the goal, he was also the President when the landing occurred.

**Incorrect Answer:** John F. Kennedy. ❌

**Correct Answer:** Richard Nixon ✅

Figure 6: An illustration of Factual Confusion. Wrong understanding of the relationship between project launch and moon landing, leading to wrong answers.

## Examples Illustrating Information Overload

### Question:

What is the capital of the country with the longest coastline?

### Retrieve Query:

Which country has the longest coastline?

### Retrieved documents:

The world's longest coastline is found in Canada, followed by Indonesia and Russia.

Together, the top 10 coastlines extend a total of almost 550,509 kilometers.

At 202,080 kilometers, Canada's coastline is the longest of the top 10, making up roughly 36.7% of the total length of all 10 coastlines.

### Summary Reasoning:

Canada is the country with the longest coastline.

### Quickly Reasoning:

*(Instead of immediately asking for Canada's capital, the system refocuses on the coastline details)*

Let's check the length of Canada's coastline. Canada's coastline is approximately 202,080 km, making it the longest in the world.

**Incorrect Answer:** 202,080 km. ❌

**Correct Answer:** Ottawa ✅

Figure 7: An illustration of Information Overload. Too much coastline information, resulting in the model answering the coastline length instead of the capital city.

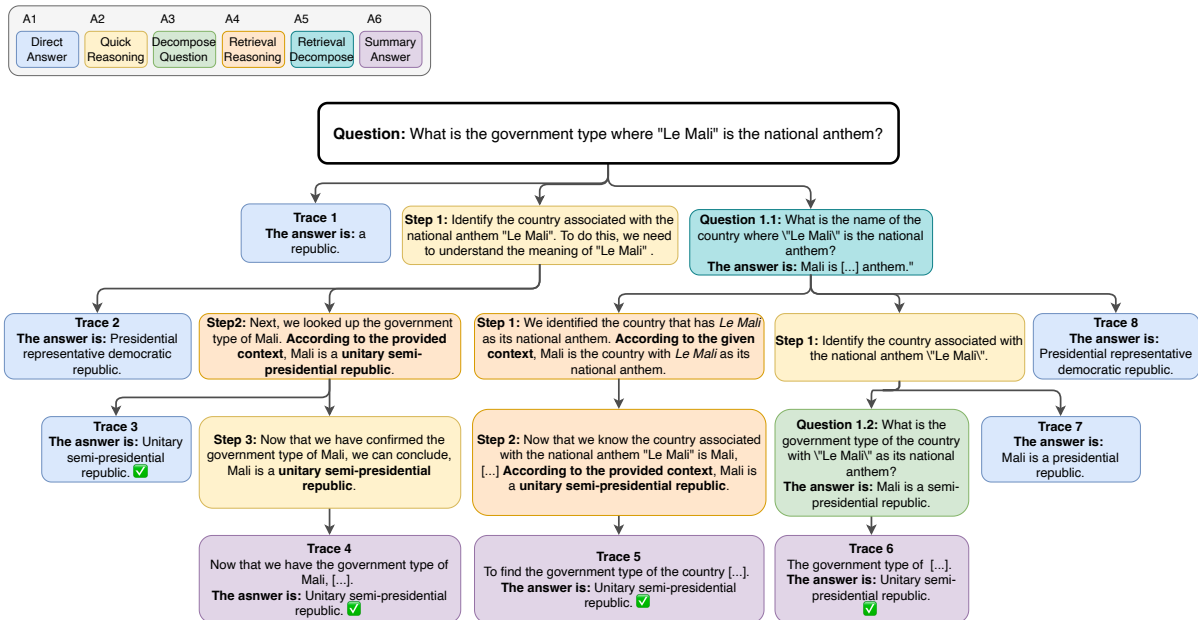


Figure 8: Illustration of how MCTS-RAG achieves a rich reasoning space and tightly integrates reasoning with retrieval.

### Examples Reasoning Correct Hallucination

**Question:**

Racemic 3-methylpent-1-ene is treated with Grubbs catalyst. How many possible products are there (excluding ethene)?

**Step 1:** The reaction involves racemic 3-methylpent-1-ene, a terminal alkene with one stereocenter. Olefin metathesis catalyzed by Grubbs catalyst typically produces internal alkenes, possibly with stereochemical variations. To understand potential outcomes clearly, I should retrieve general information about the reaction products formed from similar substrates.

**Retrieve Query:**

General products from Grubbs-catalyzed metathesis of terminal alkenes with a single stereocenter

**External Document:**

*"Grubbs-catalyzed metathesis of terminal alkenes typically produces two internal alkene products, excluding ethene."*

**Step 2:** According to the information retrieved, terminal alkenes with stereocenters undergoing Grubbs-catalyzed metathesis typically produce internal alkene products with E/Z stereochemistry. Racemic mixtures often simply double the number of stereoisomers. Thus, for racemic 3-methylpent-1-ene, I expect there will be 4 distinct products (two stereoisomers per enantiomer).

**Retrieve Query:**

Detailed stereochemical outcomes of Grubbs catalyst-mediated metathesis for racemic terminal alkenes including possibility of ring-closing metathesis, excluding ethene

**External Document:**

*"For racemic terminal alkenes with one stereocenter, Grubbs catalyst can facilitate both cross-metathesis and intramolecular ring-closing metathesis (RCM). Cross-metathesis generates internal alkene products, yielding four distinct E/Z stereoisomers due to the racemic substrate. Additionally, ring-closing metathesis generates cyclic alkene products with two more distinct stereoisomers (E/Z forms). Thus, the total number of distinct products, excluding ethene, is six."*

**Summary Answer:** There are 6 possible stereochemically distinct products (excluding ethene).

**Correct Answer:** 6 

Figure 9: Illustration of the effectiveness of MCTS-RAG. How further reasoning reduces retrieval-introduced hallucinations and improves accuracy.



**Examples Reasoning Processes Enhance Query Generation**

**Question:**  
Racemic 3-methylpent-1-ene is treated with Grubbs catalyst. How many possible products are there (excluding ethene)?

**Step 1:** The substrate is racemic 3-methylpent-1-ene, a terminal alkene with a stereocenter at the 3-position, implying two enantiomers.

**Step 2:** Given the racemic mixture, two reaction pathways should be considered: [...] Each enantiomer might yield distinct stereoisomers, thereby increasing total possible stereochemical outcomes.

**Retrieve Query:**  
Detailed stereochemical product distribution for Grubbs catalyst-mediated metathesis of racemic terminal alkenes with single stereocenter, excluding ethene

**External Document:**  
*"Racemic terminal alkenes possessing a single stereocenter commonly yield two types of products under Grubbs catalyst conditions:  
 (1) Cross-metathesis produces internal alkenes, generating E/Z stereoisomers. With racemic mixtures, each enantiomer independently produces an E/Z pair, giving four stereoisomers.  
 (2) Intramolecular ring-closing metathesis can also occur, forming cyclic products (cyclobutenes), each with cis/trans (E/Z) isomerism. Considering both enantiomers, two additional stereoisomers arise.  
 Thus, excluding ethene, a racemic terminal alkene with one stereocenter can yield up to six stereochemically distinct products."*

**Summary Answer:** There are 6 possible stereochemically distinct products (excluding ethene).

**Correct Answer:** 6 ✓

Figure 10: An illustration of the effectiveness of MCTS-RAG. Based on a clear chain of reasoning, it can generate higher quality retrieval queries and final answers, reduce hallucinations and improve accuracy.

**Examples Standard RAG Error**

**Question:**  
Racemic 3-methylpent-1-ene is treated with Grubbs catalyst. How many possible products are there (excluding ethene)?

**Retrieve Query:**  
Possible products of racemic 3-methylpent-1-ene with Grubbs catalyst

**External Document:**  
 Alkenes treated with Grubbs catalyst undergo olefin metathesis. For simple terminal alkenes, cross-metathesis typically yields symmetrical and unsymmetrical alkenes as products along with ethene as a byproduct.

**Answer:**  
 The Grubbs catalyst performs alkene metathesis, breaking and reforming double bonds. Racemic 3-methylpent-1-ene, being an alkene, will react via metathesis. Typically, this reaction yields symmetrical and asymmetrical products. Considering the double bond location and substituents, I predict there are three possible distinct products.

**Final Answer:**  
The answer is: 3

**Incorrect Answer:** 3. ✗

**Correct Answer:** 6 ✓

Figure 11: An illustration of standard RAG. Because the reasoning process is not clear enough, the final answer to the question is an illusion and the answer is wrong.