RASPberry: Retrieval-Augmented Monte Carlo Tree Self-Play with Reasoning Consistency for Multi-Hop Question Answering

Anonymous ACL submission

Abstract

Complex multi-hop question answering requires large language models (LLMs) not only to retrieve external knowledge but also to reason over the retrieved information in order to arrive at the final solution. This involves two key challenges: (i) how to effectively explore the solution space and generate more potentially correct solution candidates, and (ii) how to select the optimal solution from multiple solution candidates, both of which require a training-free approach without introducing a more powerful teacher model. To address these challenges, we propose **R**etrieval-**A**ugmented Monte Carlo Tree **S**elf-**P**lay with Reasoning Consistency (**RASPberry**), which introduces a more flexible action-level sampling granularity compared to existing methods, leverages Monte Carlo Tree Search (MCTS) for efficient solution space exploration, and utilizes an enhanced version of reasoning consistency to guide the selection of the optimal solution. Experimental results demonstrate that RASPberry effectively tackles the aforementioned two challenges, achieving more efficient RAG inference-time scaling. Our code is available at `https://github.com/NLP-LEE/RASPberry`.

1 Introduction

Retrieval-Augmented Generation (RAG) (Fan et al., 2024) enables Large Language Models (LLMs) (Brown et al., 2020) to incorporate external document knowledge during the question-answering (QA) process, significantly enhancing the performance of LLMs in single-hop QA. However, more complex multi-hop QA (Yang et al., 2018; Ho et al., 2020) presents greater challenges for LLMs. It requires LLMs to not only identify relevant knowledge from multiple documents but also to perform reasoning based on it to derive the correct response.

Although a more powerful reasoner can currently be trained through reinforcement learning (RL) to extend reasoning capabilities with RAG (Guo et al., 2025), applying RL itself involves high complexity and requires substantial computational resources. As a result, the community is increasingly focusing on a complementary and challenging problem: how to achieve effective inference-time scaling with RAG on a **smaller** LLM in a **training-free** way, without the need for **stronger teacher model supervision**, to address complex multi-hop QA tasks. Specifically, two key challenges need to be addressed to achieve this goal:

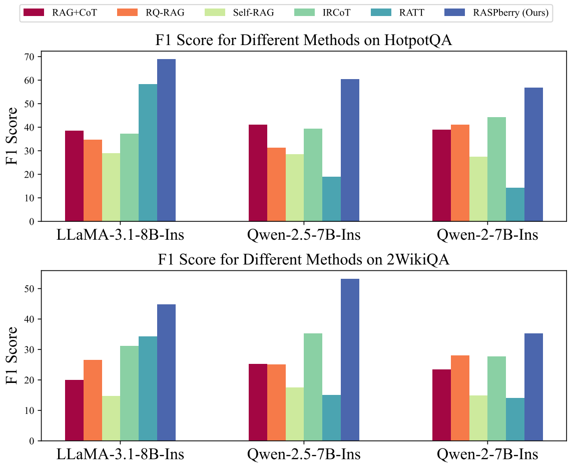


Figure 1: The F1 score achieved in multi-hop QA. Our RASPberry consistently achieves significant performance improvements across all datasets and models.

First, thoroughly exploring the solution space to generate the correct candidate solutions. Although current approaches combine chain-like (Wei et al., 2022) or tree-like (Zhang et al., 2024d) thinking structures with retrieval mechanisms, along with multiple sampling strategies (Wang et al., 2023), to achieve inference-time scaling in RAG scenarios, these methods still face significant limitations. Specifically, due to the constraints of chain-like thinking structures, the model's solution paths often get trapped in local optima. While tree-like structures offer more flexibility in solution exploration, traditional tree structures struggle

Method	#Action	Re-Retrieval	Sample Granularity ⊕ Final Selection
RAG-CoT (Wei et al., 2022)	2	✗	+*Beam*: Token-Level ⊕ Log-Likelihood +*BoN*: Sequence-Level ⊕ Log-Likelihood +*SC*: Sequence-Level ⊕ Answer Consistency
RQ-RAG (Chan et al., 2024)	4	✗	
Self-RAG (Asai et al., 2024)	3	✗	
IRCoT (Trivedi et al., 2023)	3	✓	
RATT (Zhang et al., 2024d)	5	✓	Sequence-Level ⊕ Correction & Integration
RASPberry (Ours)	**7**	✓	**Action-Level ⊕ Reasoning Consistency**

Table 1: Comparison between the related top-notch baseline methods and our proposed RASPberry.

to evaluate all possible solution paths when faced with complex problems (Zhang et al., 2024e). In other words, neither approach is effective or efficient in thoroughly exploring the solution space. Although recent work has introduced Monte Carlo Tree Search (MCTS) (Browne et al., 2012) to guide the exploration of the solution space (Zhang et al., 2024b; Qi et al., 2024), these methods do not consider the actions required to integrate external knowledge, and thus lack a suitable action set for search tree expansion in RAG scenarios.

Second, accurately determining the correct solution from the collection of generated candidate solutions. Without additional training or stronger teacher model supervision, smaller LLMs' self-scoring tends to be nearly random, leading to failures of self-reward methods such as self-verification (Weng et al., 2023). Moreover, since smaller LLMs are more likely to produce incorrect solutions during multiple sampling attempts, this also results in the failure of self-consistency methods. Although the reasoning consistency (Qi et al., 2024) alleviates this issue by considering the consistency of the reasoning process rather than just the answer, its design is tailored for scenarios that do not rely on external document support. As such, it is not fully applicable in RAG scenarios, as it does not take into account the need to ensure that the external documents supporting reasoning remain consistent when computing consistency.

To address the two challenges, we propose **R**etrieval-**A**ugmented Monte Carlo Tree **S**elf-**P**lay with Reasoning Consistency (**RASPberry**). Specifically, RASPberry consists of two components, each targeting one of the challenges: **(i) Retrieval-Augmented Monte Carlo Tree Self-Play** enables sufficient exploration of the solution space in RAG scenarios by integrating the MCTS algorithm, which generates a broader set of potentially correct candidate solutions; **(ii) Retrieval-Retained Reasoning Consistency** is used to filter the candidate solution paths obtained in (i), selecting the optimal solution path as the final answer.

As shown in Figure 1, without the need for additional training costs or a stronger teacher model, RASPberry achieves stable and significant improvements across three mainstream small LLMs (Bai et al., 2023; Yang et al., 2024; Dubey et al., 2024) and two complex multi-hop question answering datasets (Yang et al., 2018; Ho et al., 2020), enabling effective inference-time scaling in RAG scenarios that require external document knowledge.

2 Related Work

RAG for Multi-Hop Question Answering. Multi-hop question answering (QA) (Yang et al., 2018; Ho et al., 2020) presents a greater challenge to the capabilities of LLMs. For example, consider the question: "Which genus of flowering plant is found in an environment further south, Crocosmia or Cimicifuga?" This requires first retrieving documents containing information about Crocosmia and Cimicifuga, then identifying their respective locations, and subsequently reasoning based on geographic information to draw a conclusion. In other words, LLMs cannot directly answer based on the documents alone but must perform a degree of reasoning on top of this information.

Existing methods facilitate reasoning over retrieved documents by iteratively combining RAG with the Chain-of-Thought (CoT) process (Trivedi et al., 2023). Additionally, some works focus on query rewriting and query decomposition prior to retrieval to expand the range of retrieved documents (Chan et al., 2024). Meanwhile, Asai et al. (2024) proposed a self-reflection mechanism for post-reasoning self-correction. Furthermore, recent research has explored the combination of tree-like reasoning structures with RAG (Zhang et al., 2024d) to further enhance the flexibility of solution space exploration. However, unguided path exploration becomes highly challenging when the solution path is too complex. Moreover, the limited set of executable actions increases the risk of the solution path getting trapped in local optima.

As shown in Table 1, we propose RASPberry, which utilizes Monte Carlo Tree Search (MCTS) algorithm. Compared to traditional chain-like and tree-like reasoning, MCTS offers more flexible action-level sampling granularity and can estimate the potential reward of the current path based on simulation results to guide path exploration. Additionally, we integrate all of the key RAG mechanisms mentioned above into the MCTS process, resulting in a richer action set that further expands the search space. We also introduce a final path selection method to identify the optimal solution, resulting in a more effective inference-time scaling.

Inference-Time Scaling for Reasoning. Recent works primarily guide the optimization and expansion of reasoning paths in LLMs by combining a reward model aligned with human preferences (Xie et al., 2024; Zhang et al., 2024a; Chen et al., 2024a; Zhang et al., 2024c). However, this introduces additional training costs. Alternatively, some approaches employ MCTS methods for self-refinement (Zhang et al., 2024b) or self-play (Qi et al., 2024), enabling inference-time scaling in an inference-only manner. However, the designs of these works are focused on scenarios where only the model's internal parameterized knowledge is used for solving, such as mathematical reasoning or commonsense reasoning. Since these models do not have the ability to access external document knowledge, they fail to effectively adapt to the complex multi-hop QA scenarios in RAG.

In this work, we propose RASPberry, which introduces a rich and comprehensive action set tailored for RAG scenarios, used for tree expansion during the MCTS process. This enables an integration of the RAG mechanism with the MCTS reasoning structure, allowing the LLM to not only leverage its internal parameterized knowledge for reasoning but also flexibly utilize external retrieved documents to provide supportive information.

3 Preliminary

We introduce the mechanism of **Monte Carlo Tree Search** (**MCTS**) (Browne et al., 2012), which is essential for understanding our proposed RASPberry. MCTS is a decision-making algorithm widely applied in games and complex decision-making processes, which builds a search tree based on a predefined set of actions and simulates possible outcomes to estimate the value of each action. Typically, the MCTS comprises four key phases:

Selection: Starting from the root, the algorithm navigates through promising child nodes based on specific strategies (e.g., **Upper Confidence Bound applied to Trees, UCT**), continuing until it reaches a leaf node. The UCT is calculated as follows:

$$UCT(s,a) = \frac{Q(s,a)}{N(s,a)} + c\sqrt{\frac{\ln N_{\text{parent}}(s)}{N(s,a)}}, \quad (1)$$

where $Q(s,a)$ and $N(s,a)$ denote the estimated value and visit count of node s under action a, respectively (initialized to 0), while $N_{parent}(s)$ represents the visit count of s's parent node. c is a constant that balances exploitation and exploration, which is empirically set to 2 in this work. Specifically, if a node has no children, the node itself is selected. If a node has children but not all have been explored, an unexplored child is *randomly* selected. If all children have been explored, the child with the maximum UCT score is selected.

Expansion: At the leaf node, if it does not represent a terminal state (e.g., reaching the maximum depth or arriving at the final solution), feasible child nodes are added based on the current node's action set to represent potential future moves.

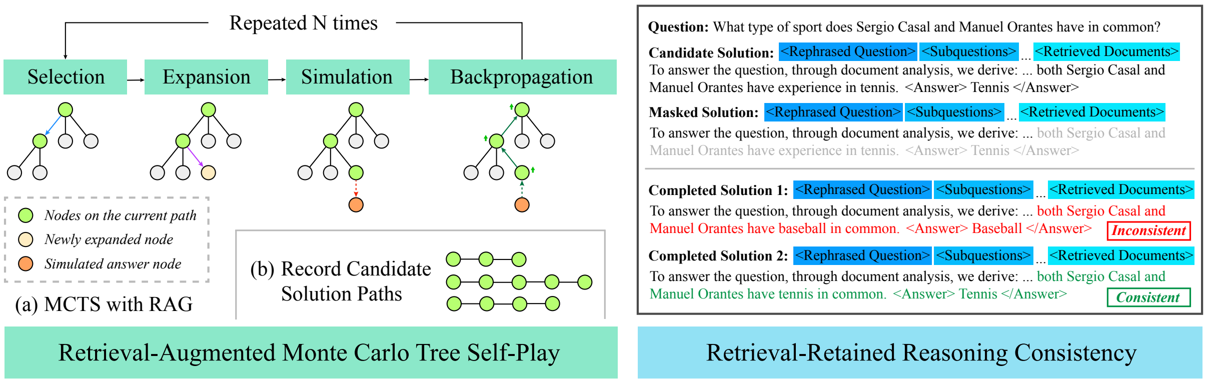
Simulation: From the newly added node, the algorithm performs random simulations (often termed *rollouts*), arbitrarily selecting moves until the game reaches its terminal state, thereby evaluating the node's potential (estimated value Q).

Backpropagation: After the simulation, the value of the terminal node (calculated based on a custom reward function) is propagated back to the root, updating the statistical data (visit counts N, estimated values Q) of all visited nodes during the simulation to guide future decisions.

Through iterative execution of these stages, MCTS incrementally builds a decision tree, optimizing strategies in scenarios where the vast state space makes direct computation infeasible.

4 RASPberry

As shown in Figure 2, our proposed **RASPberry** consists of two main components: Retrieval-Augmented Monte Carlo Tree Self-Play (**RA-MCT**


Repeated N times
Selection
Expansion
Simulation
Backpropagation
Nodes on the current path
Newly expanded node
Simulated answer node
(a) MCTS with RAG
(b) Record Candidate Solution Paths
Retrieval-Augmented Monte Carlo Tree Self-Play
Inconsistent
Consistent
Retrieval-Retained Reasoning Consistency

Question: What type of sport does Sergio Casal and Manuel Orantes have in common?

Candidate Solution: <Rephrased Question> <Subquestions> ... <Retrieved Documents>
To answer the question, through document analysis, we derive: ... both Sergio Casal and Manuel Orantes have experience in tennis. <Answer> Tennis </Answer>

Masked Solution: <Rephrased Question> <Subquestions> ... <Retrieved Documents>
To answer the question, through document analysis, we derive: ... both Sergio Casal and Manuel Orantes have experience in tennis. <Answer> Tennis </Answer>

Completed Solution 1: <Rephrased Question> <Subquestions> ... <Retrieved Documents>
To answer the question, through document analysis, we derive: ... both Sergio Casal and Manuel Orantes have baseball in common. <Answer> Baseball </Answer>

Completed Solution 2: <Rephrased Question> <Subquestions> ... <Retrieved Documents>
To answer the question, through document analysis, we derive: ... both Sergio Casal and Manuel Orantes have tennis in common. <Answer> Tennis </Answer>

Figure 2: Our proposed RASPberry consists of Retrieval-Augmented Monte Carlo Tree Self-Play (left) for solution candidates generation and Retrieval-Retained Reasoning Consistency (right) for final solution discrimination.

Self-Play in §4.1) for solution space exploration and generation of candidate reasoning paths, and Retrieval-Retained Reasoning Consistency (**RR-RC** in §4.2) for optimal final path selection.

4.1 Retrieval-Augmented MCT Self-Play

As shown in Figure 2 (left) (a), we tightly integrate the MCTS in §3 with RAG, where each tree node represents the response generated by the LLM given all previously generated content along the current path after executing a specific action, serving as a unit in constructing the overall solution path. The details are as follows:

Reasoning Actions in the RAG Setting. In order to make the MCTS algorithm more adaptable to the RAG setting, we design a comprehensive action set that incorporates nearly all the key concepts of RAG for constructing the search tree. Specifically, the action set consists of seven actions, which are:

- ◦ A_1: *Query decomposition* (Zhou et al., 2023).
- ◦ A_2: *Query rephrasing* (Ma et al., 2023).
- ◦ A_3: *Document retrieval* (Ram et al., 2023).
- ◦ A_4: *Document analysis* (Wei et al., 2022).
- ◦ A_5: *Answer extraction* (Wei et al., 2022).
- ◦ A_6: *Critical rethinking* (Asai et al., 2024).
- ◦ A_7: *Document re-retrieval* (Trivedi et al., 2023).

$\{A_1, A_2\}$ are query optimization actions, aimed at enhancing query understanding and improving the retrieval of relevant documents by decomposing the query into multiple subqueries or rephrasing it in different ways. $\{A_3, A_7\}$ are actions for document retrieval, with A_7 differing from A_3 in requiring an assessment of the need for further retrieval and the construction of a follow-up query based on existing information before execution. $\{A_4, A_5, A_6\}$ are reasoning and analysis actions based on the retrieved documents, incorporating self-reflection for error correction and ultimately extracting an answer that addresses the user query.

Based on these, we define the action set $\mathbf{A} = \{A_1, A_2, A_3, A_4, A_5, A_6, A_7\}$. At each step i, MCTS executes an action a_i from $\mathbf{A}$. We then use a_i to prompt the LLM to generate the next node state s_i, based on the previously generated solution path $r \oplus s_1 \oplus s_2 \oplus ... \oplus s_{i-1}$, where r represents the root, i.e., the user query, and s represents the reasoning steps (node states) generated by the LLM. Additionally, it is important to note that certain actions require partial orders. For example, $\{A_6, A_7\}$ can only happen after A_4, and $\{A_4, A_5\}$ can only happen after A_3. More details on action-related prompts can be found in Appendix B.

MCTS Reward Function. For the calculation of the estimated value Q of each node, we adopt a method similar to that used in AlphaGo (Silver et al., 2017), scoring each intermediate node based on its contribution to the final correct answer. That is, actions that more frequently lead to the correct answer are given higher scores, making them more likely to be selected during the tree expansion.

During the simulation, when a valid answer node is reached, we score the answer node using Self-Consistency (SC) (Wang et al., 2023). Specifically, we sample M candidate answers and choose the most frequent answer as the final answer for the current path. The estimated value of the answer node s_d is given by $Q(s_d, a_d) = m/M$, where m is the count of the most frequent answer. This score is then backpropagated along the solution path $P = r \oplus s_1 \oplus s_2 \oplus ... \oplus s_d$, meaning the score of each intermediate node s_i is updated as $Q(s_i, a_i) = Q(s_i, a_i) + Q(s_d, a_d)$.

Solution Space Exploration with MCTS Rollout. Starting from the root node r (the user query), we iteratively explore solution space by following the MCTS procedure outlined in §3, performing multiple rounds of simulation (rollout) to achieve more accurate node value estimations. When the search reaches a terminal node (either the answer node or the maximum tree depth), we obtain a solution path from the root node r to the terminal node s_d.

However, while traditional MCTS selects one path as the final solution based on a specific metric (Browne et al., 2012), defining a reliable single metric to select a solution path that contains the correct answer is challenging without incurring additional training costs (Qi et al., 2024). Therefore, as shown in Figure 2 (left) (b), we collect all valid paths (paths containing the answer node) generated during the MCTS rollout as candidate solution paths. Subsequently, an additional validation process (in §4.2) is applied to filter the final path, significantly reducing the difficulty for small LLMs in selecting the optimal solution path.

4.2 Retrieval-Retained Reasoning Consistency

Inspired by the Reasoning Consistency (RC) in rStar (Qi et al., 2024), we propose Retrieval-Retained Reasoning Consistency (RR-RC) to better adapt to the RAG setting. Unlike RC, which randomly selects a reasoning step to begin masking, RR-RC ensures that, after masking, the retrieved documents in the remaining solution path provide sufficient information to support subsequent reasoning steps, as illustrated in Figure 2 (right).

Specifically, for solution path $\mathbf{P} = [r \oplus s_1 \oplus s_2 ... \oplus s_k] \oplus ... \oplus s_d$, we mask the reasoning steps starting from a randomly selected step i ($i < d$), which is behind the last document retrieval step ($i > k$). And the content before the last document retrieval step k is kept unchanged to ensure that the necessary retrieved information accessible to the model is retained. Subsequently, we provide masked solution path $\mathbf{P}_{masked} = [r \oplus s_1 \oplus s_2 ... \oplus s_k] \oplus ... \oplus s_{i-1}$ as prior information to the LLM to complete the subsequent reasoning steps. As shown in Figure 2 (right), we compare the generated answer after completion with the original answer. If they are consistent, we consider the solution path to be a valid path for final selection.

To improve efficiency, unlike the peer discrimination mechanism introduced by rStar (which requires extra effort to select a model with similar capabilities), RR-RC uses the same model for self-discrimination as the one used to generate the solution paths. Additionally, during the completion process in RR-RC, we introduce the Best-of-N Sampling (BoN) mechanism to encourage the completion of more diverse solution paths. The underlying intuition is that, in the absence of a teacher providing feedback, one can self-verify by adopting different reasoning approaches while given a predefined solution path. If the same answer can be obtained through different reasoning paths, we can consider the answer more likely to be correct, without the need for peer verification from others.

Final Solution Path Selection. After applying RR-RC to all candidates, we compute the final score of each solution path in the filtered solution paths by integrating its reward with the estimated value of the answer node obtained from rollouts. Specifically, given the filtered answer set $\mathbf{A}_f$ after applying RR-RC and the unfiltered answer set $\mathbf{A}_u$ before its application, we compute the final score $R(ans)$ of each answer ans in $\mathbf{A}_f$ as:

$$R(ans) = Q(ans) + \frac{N_f(ans)}{N_u(ans)}, \qquad (2)$$

where $N_f(ans)$ and $N_u(ans)$ represent the frequency of ans in $\mathbf{A}_f$ and $\mathbf{A}_u$, respectively. We posit that answers retained to a greater extent after RR-RC filtering are more likely to be correct. $Q(ans)$ denotes the estimated value of answer ans during simulation. Finally, we select the solution path corresponding to the answer with the highest final score R as the final solution path.

5 Experiments

In this work, we select three mainstream small LLMs, namely LLaMA-3.1-8B-Instruct (Dubey et al., 2024), Qwen-2.5-7B-Instruct (Yang et al., 2024), and Qwen-2-7B-Instruct (Bai et al., 2023). Additionally, we choose two commonly used knowledge-intensive multi-hop QA datasets, namely HotpotQA (Yang et al., 2018) and 2WikiMultiHopQA (2WikiQA) (Ho et al., 2020), both of which are based on wiki documents. We divide the supporting documents in the dataset by their respective topics, embed them into text vectors using a mainstream dense retriever (BGE-M3 (Chen et al., 2024b)), and then use FAISS (Douze et al., 2024) to maintain a local vector database for retrieval. Furthermore, we include a wide range of baseline methods for comparison, categorized into single-round and multi-round RAG baselines:

Dataset	Method	LLaMA-3.1-8B-Ins			Qwen-2.5-7B-Ins			Qwen-2-7B-Ins		
		P	R	F1	P	R	F1	P	R	F1
HotpotQA	RAG-CoT	38.42	51.77	38.55	41.54	42.77	41.06	39.33	39.92	38.84
	RQ-RAG	34.97	36.92	34.63	32.50	31.20	31.23	41.56	42.24	41.01
	Self-RAG	28.57	30.00	28.99	29.60	28.27	28.59	28.18	29.33	27.55
	IRCoT	37.32	39.93	37.24	39.78	40.35	39.42	44.66	46.42	44.35
	RATT	58.72	60.89	58.29	18.98	19.77	18.87	13.92	15.18	14.34
	RASPberry (Ours)	**70.54**	**69.44**	**68.89**	**59.98**	**65.43**	**60.51**	**56.69**	**60.72**	**56.85**
2WikiQA	RAG-CoT	20.33	20.13	20.04	24.97	25.67	25.26	23.27	23.62	23.39
	RQ-RAG	26.67	27.17	26.60	25.50	25.00	25.07	27.68	28.78	28.02
	Self-RAG	14.40	15.47	14.72	17.42	18.03	17.56	14.75	15.03	14.87
	IRCoT	31.63	31.07	31.13	34.67	37.32	35.21	27.77	29.17	27.75
	RATT	34.70	35.50	34.35	15.23	15.90	15.07	13.49	15.38	14.06
	RASPberry (Ours)	**45.00**	**44.83**	**44.87**	**52.07**	**56.37**	**53.25**	**35.19**	**36.23**	**35.27**

Table 2: Overall performance comparison, *P* represents precision, while *R* represents recall.

(a) Single-Round RAG Baselines. (i) RAG-CoT, which is RAG combined with naive CoT (Wei et al., 2022). (ii) RQ-RAG (Chan et al., 2024), which adds query rewriting and subquery decomposition. (iii) Self-RAG (Asai et al., 2024), which incorporates self-reflection for reasoning results.

(b) Multi-Round RAG Baselines. (i) IRCoT (Trivedi et al., 2023), which enables iterative interleaving of RAG and CoT. (ii) RATT (Zhang et al., 2024d), which integrates RAG with a tree-like structure (Tree-of-Thought (Yao et al., 2024)).

It is important to note that for all methods in our experiments, we adopt a unified dense retrieval mechanism and set the number of returned documents per retrieval to 4 by default. Design differences and further implementation details are provided in Table 1 and Appendix A.

5.1 Main Results

As shown in Table 2, we compare the performance of various baselines with our proposed RASPberry. Notably, except for RATT and RASPberry, which adopt a unique final path selection strategy, all other baselines employ the commonly used Best-of-N (BoN) strategy based on likelihood evaluation of generated sequences. Notably, except for RATT and RASPberry, which adopt a unique final path selection strategy, all other baselines employ the commonly used Best-of-N (BoN) strategy based on likelihood evaluation of generated sequences. Additionally, since RAG-CoT, RQ-RAG, and Self-RAG are all single-round RAG processes, we set their sampling sequence count to 3 (BoN@3) by default. On the other hand, IRCoT, RATT, and RASPberry are multi-round RAG processes, so in addition to setting their sampling count to 3, we configure the number of RAG rounds/rollouts to 8.

It is worth noting that RASPberry consistently outperforms the baseline across all datasets and models. Although RATT, which incorporates tree-like thinking structure, demonstrates superior performance under certain experimental settings (using LLaMA-3.1-8B-Instruct on HotpotQA), even in such cases, RASPberry achieves a 10.6 point higher F1 score. Furthermore, RASPberry consistently maintains a significant performance improvement over RATT across all experimental settings.

5.2 Comparison of Sampling Strategies

Meanwhile, we also explore the use of alternative sampling strategies, beyond BoN, for the remaining baselines, specifically Beam Search (Beam) and Self-Consistency (SC). As shown in Table 3, although Beam Search, which applies a finer-grained token-level sampling strategy, and SC, which clusters answers based on semantic similarity, can alleviate some of the issues caused by the coarser sequence-level sampling of BoN, token-level sampling tends to get stuck in local optima. Furthermore, due to the capability limitations of small LLMs, the correct answers obtained through sampling are often in the minority, which leads to suboptimal performance with SC on small LLMs.

In contrast, our proposed RASPberry adopts a more flexible action-level sampling granularity (which lies between token-level and sequence-level, with different actions having different preset output

Method	Beam@3	BoN@3	SC@3
LLaMA-3.1-8B-Ins			
RAG-CoT	54.47	38.55	56.39
RQ-RAG	56.47	34.63	49.94
Self-RAG	47.15	28.99	42.81
IRCoT	56.70	37.24	43.23
RATT	—	58.29	—
RASPberry (Maj)	—	66.27	—
RASPberry (Ours)	—	**68.89**	—
Qwen-2.5-7B-Ins			
RAG-CoT	49.42	41.06	51.94
RQ-RAG	52.31	31.23	49.11
Self-RAG	53.69	28.59	49.32
IRCoT	53.12	39.42	47.05
RATT	—	18.87	—
RASPberry (Maj)	—	60.02	—
RASPberry (Ours)	—	**60.51**	—

Table 3: Comparison with different sampling strategies on HotpotQA. *Maj* represents (answer) majority vote. All configurations are the same as those in Table 2.

lengths). Additionally, by integrating the MCTS algorithm, previously executed actions (stored as tree node states) can be reused in different subsequent actions, further extending the sampling scope of the solution space. Combined with RR-RC, which guides the final path selection by considering the solution paths' validity, RASPberry consistently achieves superior performance compared to existing methods that use various sampling strategies.

5.3 Scaling Efficiency Analysis

To validate the scalability advantage of our proposed RASPberry (i.e., analyzing whether the method can incorporate more correct candidate solutions as the number of samples increases), we configured different rollouts (reflecting the number of samples) and compared it with various scalable baselines. It is important to note that we only compared methods capable of multi-round retrieval, specifically IRCoT and RATT, to ensure that the retrieved information is similarly scalable, avoiding issues related to insufficient external information in single-round RAG that could skew the comparison of their scalability. We measured the maximum achievable F1 score under ideal configurations (i.e., selecting the optimal answer from the candidate solutions) as the number of rollouts increased. To evaluate the scalability, we fitted a linear model to the performance expansion and used the slope of the fitting line to quantify the scalability (a larger slope indicates better scalability).

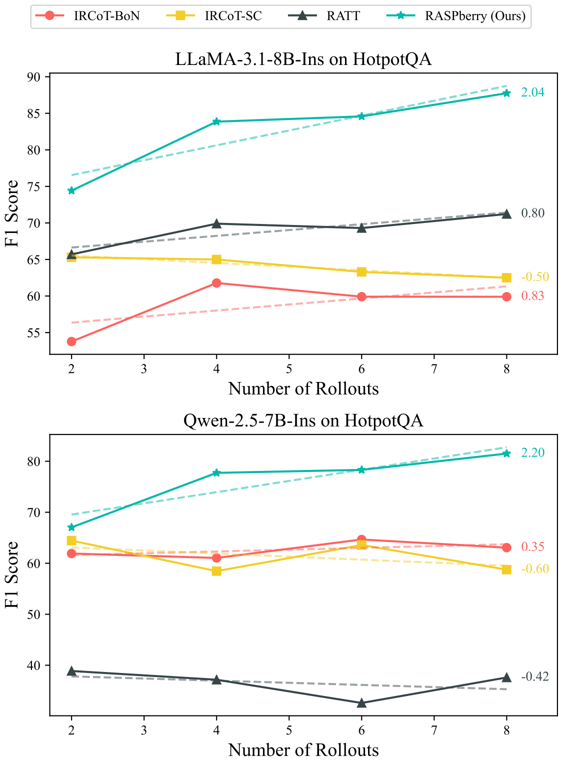


Figure 3: Performance comparison on HotpotQA under different number of rollouts. The dashed line represents the linear fit, with the slope indicated.

As shown in Figure 3, even with a relatively low number of samples (2 rollouts), our proposed RASPberry outperforms the baseline methods. Furthermore, both variants of IRCoT (IRCoT-BoN and IRCoT-SC) and RATT fail to achieve effective scalability in reasoning, as they exhibit minimal or even deteriorating performance with an increasing number of rollouts (with slopes less than 1, or even negative). In contrast, RASPberry demonstrates effective and efficient scalability, achieving a performance scaling slope greater than 2. This confirms that our proposed RASPberry is able to more thoroughly explore the solution space and is less likely to get stuck in local optima.

5.4 The Effectiveness of RA-MCT Self-Play

To independently verify the effectiveness of RA-MCT Self-Play in exploring the solution space within RASPberry, we removed RR-RC, which is responsible for final path selection, and instead simply employed a majority vote mechanism to select the most frequent answer from the candidate solutions. As shown in Table 3, even in this scenario, our proposed method demonstrates nontrivial performance improvement, confirming that RA-MCT Self-Play alone is capable of effectively exploring more correct answers as candidate solutions. However, it is important to note that applying our proposed, more tailored RR-RC further leads to a significant performance gain.

Moreover, as shown in Figure 3, as the number of MCTS rollouts increases, RA-MCT Self-Play gradually explores more correct solutions, rather than being confined to local optima like other baselines. This further demonstrates the effectiveness of RA-MCT Self-Play in RASPberry.

Method	**(BoN+SC)@3**	**RR-RC (Ours)**
	LLaMA-3.1-8B-Ins	
RAG-CoT	31.39	**45.79**
RQ-RAG	41.96	**43.42**
Self-RAG	28.53	**35.39**
IRCoT	40.31	**52.89**

Table 4: Performance of baselines with simple joint effect (BoN+SC) and our proposed RR-RC on HotpotQA.

5.5 The Effectiveness of RR-RC

To independently verify the effectiveness of RR-RC, which is responsible for path selection in RASPberry, we first applied RR-RC to the baseline methods. As shown in Table 4, although our proposed RR-RC incorporates the evaluation strategies of BoN and SC, it differs from directly merging BoN and SC evaluations (where the total score, obtained by summing the scores from BoN and SC methods, is considered as the final evaluation score). RR-RC, on the other hand, integrates both the consistency and validity of reasoning paths into the evaluation criteria, ensuring that the final selected solution is more reliable. Compared to merely merging BoN and SC, RR-RC is able to select a more accurate final solution. However, it is important to note that, due to the inherent limitations in the exploration efficiency of the baselines' solution space, the RR-RC method may not directly yield significant advantages, as its evaluation still requires considering the confidence in the candidate paths generated during the process.

However, for our proposed RA-MCT Self-Play, which can effectively explore the solution space, combining it with the RR-RC method yields significant advantages. As shown in Table 5, we compared various mainstream solution selection methods and confirmed that, for small LLMs, the performance difference between Self-Verification (which is based on self-scoring) and Random Select (which randomly chooses a candidate solution as the final answer) is minimal. Additionally, when we apply RR-RC without the BoN strategy, using only a single sampled path to complete the reasoning process (Self-Naive) introduces a certain level of randomness, leading to performance instability, which is due to the inherent limitations of small LLMs. However, when we use a stronger teacher model as a discriminator, such as GPT-4o-Mini, even without the BoN strategy, it can still select a larger number of correct final solutions. It is worth noting that when the BoN strategy is applied to RR-RC on small LLMs (Self-BoN), the performance achieved is comparable to that of using a stronger model, further demonstrating the efficiency and scalability of our proposed RR-RC method.

Model	**Discrimination**	**F1**
LLaMA-3.1-8B-Ins	Random Select	51.79
	Majority Vote	66.27
	Self-Verification	55.09
	Self-Integration	16.95
	RR-RC (Ours)	
	Self-Naive	64.75
	Self-BoN	68.89
	GPT-4o-Mini	**70.39**
	Optimal	*87.75*

Table 5: Performance on HotpotQA with different final solution selection methods (discriminator). *Optimal* represents the performance upper bound achieved when always selecting the optimal solution.

6 Conclusion

For complex multi-hop question answering, existing methods fail to ensure (i) effective exploration of the solution space and (ii) correct selection of the final solution. To address these challenges, we propose RASPberry. Compared to existing methods, RASPberry uses a more flexible sampling granularity, enabling more effective solution space exploration. Additionally, we adopts an enhanced version of reasoning consistency tailored to the RAG scenario, offering a more comprehensive final solution selection. Overall, our proposed RASPberry achieves more effective and efficient RAG inference-time scaling in a training-free manner, without the need for a stronger teacher model.

Limitations

Although our proposed RASPberry achieves effective and efficient inference-time scalability in multi-hop QA, due to computational limitations, we have only explored a training-free, inference-only design path. Future work could focus on treating MCTS as a automatic process for synthesizing long reasoning chains, and then filtering out high-quality reasoning paths that align with human preferences. These solution paths could be used as preference data to fine-tune the LLM's internal parameters via reinforcement learning, enabling the model to directly generate the desired high-quality reasoning chains during inference. This would facilitate a more integrated and streamlined process.

Ethical Considerations

It is widely acknowledged that LLMs are capable of generating predictions that exhibit bias. This issue becomes especially pronounced when the input queries possess sensitive characteristics. In light of some potential issues, this study advocates for usage under research purposes. Appropriate care should thus be taken when applying such approaches for any non-research purpose.

In this study, our use of existing artifacts is consistent with their intended purposes. All the datasets and models used in this work are publicly available. Specifically, LLaMA-3.1-8B-Instruct have Llama 3.1 Community License Agreement[1]. Qwen-2.5-7B-Instruct, Qwen-2-7B-Instruct, HotpotQA dataset, and 2WikiMultiHopQA (2WikiQA) dataset have Apache-2.0 license[2].

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*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, 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.

Chi-Min Chan, Chunpu Xu, Ruibin Yuan, Hongyin Luo, Wei Xue, Yike Guo, and Jie Fu. 2024. Rq-rag: Learning to refine queries for retrieval augmented generation. *arXiv preprint arXiv:2404.00610*.

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. 2024a. Alphamath almost zero: process supervision without process. *arXiv preprint arXiv:2405.03553*.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024b. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library. *arXiv preprint arXiv:2401.08281*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6491–6501.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting in retrieval-augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5303–5315.

[1] https://huggingface.co/meta-llama/Llama-3.1-70B-Instruct/blob/main/LICENSE

[2] https://www.apache.org/licenses/LICENSE-2.0

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

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 11:1316–1331.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*.

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.

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*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2023. Large language models are better reasoners with self-verification. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2550–2575.

Yuxi Xie, Anirudh Goyal, Wenyue Zheng, Min-Yen Kan, Timothy P Lillicrap, Kenji Kawaguchi, and Michael Shieh. 2024. Monte carlo tree search boosts reasoning via iterative preference learning. *arXiv preprint arXiv:2405.00451*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

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

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024a. Rest-mcts*: Llm self-training via process reward guided tree search. *arXiv preprint arXiv:2406.03816*.

Di Zhang, Xiaoshui Huang, Dongzhan Zhou, Yuqiang Li, and Wanli Ouyang. 2024b. Accessing gpt-4 level mathematical olympiad solutions via monte carlo tree self-refine with llama-3 8b. *arXiv preprint arXiv:2406.07394*.

Di Zhang, Jianbo Wu, Jingdi Lei, Tong Che, Jiatong Li, Tong Xie, Xiaoshui Huang, Shufei Zhang, Marco Pavone, Yuqiang Li, et al. 2024c. Llama-berry: Pairwise optimization for o1-like olympiad-level mathematical reasoning. *arXiv preprint arXiv:2410.02884*.

Jinghan Zhang, Xiting Wang, Weijieying Ren, Lu Jiang, Dongjie Wang, and Kunpeng Liu. 2024d. Ratt: Athought structure for coherent and correct llmreasoning. *arXiv preprint arXiv:2406.02746*.

Xuan Zhang, Chao Du, Tianyu Pang, Qian Liu, Wei Gao, and Min Lin. 2024e. Chain of preference optimization: Improving chain-of-thought reasoning in llms. *arXiv preprint arXiv:2406.09136*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, et al. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

A Implementation Details

In this work, we set the default configurations of RASPberry as follows:

Parameter	Default Value
MCTS Configuration	
Number of Rollouts	8
Max Tree Depth	10
MCTS Exploration Weight	2.0
LLM Configuration	
Number of Votes	3
Temperature	0.8
Top-K	40
Top-P	0.95

Table 6: Default configurations of our RASPberry.

In the experiments, we used the following datasets: HotpotQA[3] and 2WikiMultiHopQA[4] (2WikiQA). The LLMs employed are LLaMA-3.1-8B-Instruct[5], Qwen-2.5-7B-Instruct[6], and Qwen-2-7B-Instruct[7]. For all baseline methods, we set the model configurations identical to those in Table 6. For single-round RAG baselines, we set the round to 1. For multi-round RAG baselines, we set the round to 8, aligning with the number of rollouts in MCTS to ensure fairness.

B Prompt Examples

Since both the HotpotQA and 2WikiQA datasets are multi-hop question-answering datasets based on wiki knowledge, we construct a unified set of prompts for these datasets. For each action that requires generation by the LLM, we provide five demonstrations in the prompts. Specifically, the prompts for different actions are provided in the color boxes below.

[3] https://hotpotqa.github.io

[4] https://github.com/Alab-NII/2wikimultihop

[5] https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct

[6] https://huggingface.co/Qwen/Qwen2.5-7B-Instruct

[7] https://huggingface.co/Qwen/Qwen2-7B-Instruct

A_1: Query Decomposition

Given an input question, decompose it into multiple smaller and indivisible sub-questions. The original question will be enclosed in <Original Question> and </Original Question>. Corresponding sub-questions should be enclosed in <Subquestions> and </Subquestions> tags.
<Question>
In what school district is Governor John R. Rogers High School, named after John Rankin Rogers, located?
</Question>
<Subquestions>
1.Where is Governor John R. Rogers High School geographically located?
2.What is the name of the school district that includes Governor John R. Rogers High School?
</Subquestions>
<Question>
Which Australian racing driver won the 44-lap race for the Red Bull Racing team?
</Question>
<Subquestions>
1.Which 44-lap race was won by a driver for the Red Bull Racing team?
2.Which Australian racing drivers are part of the Red Bull Racing team?
3.Which Australian racing driver, as part of the Red Bull Racing team, won the 44-lap race?
</Subquestions>
<Question>
What star of *Parks and Recreation* appeared in November?
</Question>
<Subquestions>
1.Which actors are considered stars of *Parks and Recreation*?
2.What event or appearance involving a star of *Parks and Recreation* occurred in November?
3.Which specific star of *Parks and Recreation* made an appearance in November?
</Subquestions>
<Question>
Which genus of flowering plant is found in an environment further south, Crocosmia or Cimicifuga?
</Question>
<Subquestions>
1.What are the typical environments where the genus Crocosmia is found?
2.What are the typical environments where the genus Cimicifuga is found?
3.Which environment, associated with Crocosmia or Cimicifuga, is located further south?
</Subquestions>
<Question>
In what year did the man who shot the Chris Stockley, of The Dingoes, die?
</Question>
<Subquestions>
1.Who was the man who shot Chris Stockley, a member of The Dingoes?
2.In what year did the man who shot Chris Stockley die?
</Subquestions>
<Question>
{User Query}
</Question>
<Subquestions>

A_2: Query Rephrasing

Given an input question, rephrase it into a more intuitive and easier-to-understand version. The original question is enclosed within <Original Question> </Original Question> tags, and the corresponding rephrased question is enclosed within <Rephrased Question> </Rephrased Question> tags.

<Original Question>
In what school district is Governor John R. Rogers High School, named after John Rankin Rogers, located?
</Original Question>
<Rephrased Question>
What school district is Governor John R. Rogers High School in?
</Rephrased Question>
<Original Question>
Which Australian racing driver won the 44-lap race for the Red Bull Racing team?
</Original Question>
<Rephrased Question>
Who is the Australian racing driver that won a 44-lap race for the Red Bull Racing team?
</Rephrased Question>
<Original Question>
What star of *Parks and Recreation* appeared in November?
</Original Question>
<Rephrased Question>
Which actor from *Parks and Recreation* made an appearance in November?
</Rephrased Question>
<Original Question>
Which genus of flowering plant is found in an environment further south, Crocosmia or Cimicifuga?
</Original Question>
<Rephrased Question>
Between Crocosmia and Cimicifuga, which plant genus is typically found further south?
</Rephrased Question>
<Original Question>
In what year did the man who shot the Chris Stockley, of The Dingoes, die?
</Original Question>
<Rephrased Question>
What is the year of death for the man who shot Chris Stockley, of The Dingoes?
</Rephrased Question>
<Original Question>
{User Query}
</Original Question>
<Rephrased Question>

A_4: Document Analysis

...{Previous Actions}...
To answer the question, through document analysis, we derive: Based on the content provided in the document:
<Document Analysis>
1.Document 1 describes Rogers High School in Arkansas, managed by the Rogers School District. This is not relevant to the query about “Governor John R. Rogers High School.”
2.Document 2 describes Rogers High School in Minnesota, part of Elk River School District 728. This is also not related to the query.
3.Document 3 describes John R. Rogers High School in Spokane, Washington, part of Spokane Public Schools (District No. 81). While it shares a connection with John Rankin Rogers, it is not the school in question.
4.Document 4 provides information about Governor John R. Rogers High School, located in the Puyallup School District of Washington, United States. It specifically states that this school is named after the former governor John Rankin Rogers and opened in 1968.
Conclusion: Governor John R. Rogers High School is located in the **Puyallup School District**.
</Document Analysis>

...{4× demonstrations}...

...{Previous Actions}...
To answer the question, through document analysis, we derive: Based on the content provided in the document:
<Document Analysis>

A_5: Answer Extraction

```
...{Previous Actions}...
Summarizing the information above, now we extract the answer, the answer is:
<Answer>
Puyallup School District of Washington
</Answer>
...{Previous Actions}...
Summarizing the information above, now we extract the answer, the answer is:
<Answer>
Barcelona
</Answer>
...{Previous Actions}...
Summarizing the information above, now we extract the answer, the answer is:
<Answer>
Lowestoft
</Answer>
...{Previous Actions}...
Summarizing the information above, now we extract the answer, the answer is:
<Answer>
Crocosmia
</Answer>
...{Previous Actions}...
Summarizing the information above, now we extract the answer, the answer is:
<Answer>
1987
</Answer>
...{Previous Actions}...
Summarizing the information above, now we extract the answer, the answer is:
<Answer>
```

A_6: Critical Rethinking

...{Previous Actions}...
By rethinking the above content, we can find that:
<Rethinking>
To critically rethink whether the information and analysis provided can reliably generate the conclusion that “Governor John R. Rogers High School is located in the Puyallup School District of Washington,” let’s analyze the key points and logical consistency:
Review of the Provided Information:
1.**Documents Relevant to the Query:**
• **Document 4** explicitly mentions **Governor John R. Rogers High School** and states that it is in the **Puyallup School District of Washington, United States**. It also confirms the school’s name, connection to John Rankin Rogers, and establishment date (1968).
• **Document 3** discusses a similarly named **John R. Rogers High School**, located in Spokane, Washington, and part of Spokane Public Schools. While related to the same individual, it is a different school.
2.**Irrelevant Documents:**
• **Documents 1 and 2** describe Rogers High Schools in Arkansas and Minnesota, respectively. These schools do not match the query’s specified name or location.
3.**Analysis of Document Alignment:**
• Document 4 directly matches the query by providing the specific name, location, and school district of Governor John R. Rogers High School.
• The analysis appropriately dismisses Documents 1, 2, and 3 as unrelated to the query.
Reconsideration of Logical Flow:
1.The document analysis correctly identifies Document 4 as the authoritative source for answering the query.
2.The irrelevant documents do not introduce any conflicting information.
3.The naming distinction between “Governor John R. Rogers High School” (Puyallup) and “John R. Rogers High School” (Spokane) is clearly maintained.
Conclusion Verification:
Based on the information provided in Document 4, the conclusion that “Governor John R. Rogers High School is located in the Puyallup School District of Washington” is well-supported and logically derived. There is no ambiguity or conflicting evidence that undermines this conclusion.
</Rethinking>

...{4× demonstrations}...

...{Previous Actions}...
By rethinking the above content, we can find that:
<Rethinking>

Re-retrieval Necessity in A_7: Document Re-retrieval

<Document Analysis>
Based on the retrieved documents, there is insufficient information to conclusively answer the question ...
To accurately answer the question, additional context or documents specifically detailing ... Expanding the scope of the search to include ... may help locate the necessary information.
</Document Analysis>
According to the document analysis, do we need to retrieve more documents to answer the question? Respond with Yes or No.
Response: Yes
<Document Analysis>
Based on the content provided in the document:
...
Conclusion: The team that featured in both the 2011 and 2012 Copa del Rey Finals is **Barcelona**.
</Document Analysis>
According to the document analysis, do we need to retrieve more documents to answer the question? Respond with Yes or No.
Response: No

...{3× demonstrations}...

</Document Analysis>
{Current Document Analysis}
</Document Analysis>
According to the document analysis, do we need to retrieve more documents to answer the question? Respond with Yes or No.
Response:

Remaining Query Construction in A_7: Document Re-retrieval

<Original Question>
In what year did the man who shot Chris Stockley, of The Dingoes, die?
</Original Question>
<Document Analysis>
...
</Document Analysis>
Given the original question and the document analysis, please create a remaining question that requires further retrieval.
<Remaining Question>
What is the year of death for Dennis Allen, the Melbourne drug dealer who shot Chris Stockley of The Dingoes?
</Remaining Question>

...{4× demonstrations}...

<Original Question>
{User Query}
</Original Question>
<Document Analysis>
...
</Document Analysis>
Given the original question and the document analysis, please create a remaining question that requires further retrieval.
<Remaining Question>