GRAT: Guiding Retrieval-Augmented Reasoning through Process Rewards Tree Search

Xianshu Peng[1], Wei Wei[1]*
[1]School of Computer Science & Technology, Huazhong University of Science and Technology
{xspeng, weiw}@hust.edu.cn

Abstract

Enhancing large models for complex multi-hop question-answering has become a research focus in the Retrieval-augmented generation (RAG) area. Many existing approaches aim to mimic human thought processes by enabling large models to perform retrieval-augmented generation step by step. However, these methods can only perform single chain reasoning, which lacks the ability for multi-path exploration, strategic look-ahead, stepwise evaluation, and global selection. In addition, to effectively decompose complex problems, these methods can only rely on labor-intensive intermediate annotations for supervised fine-tuning. To address these issues, we propose GRAT, an algorithm guided by Monte Carlo Tree Search (MCTS) and process rewards. GRAT not only enables self-evaluation and self-correction but also assigns fine-grained rewards to each intermediate step in the search path. These fine-grained annotations can be used for model self-training, which enables GRAT to continuously self-update its problem analysis and reasoning capabilities. We conducted experiments on four multihop QA datasets: HotPotQA, 2WikiMultiHopQA, MuSiQue, and Bamboogle, demonstrating that GRAT outperforms various RAG-based methods. Additionally, incorporating self-training significantly enhances GRAT's reasoning performance. [1]

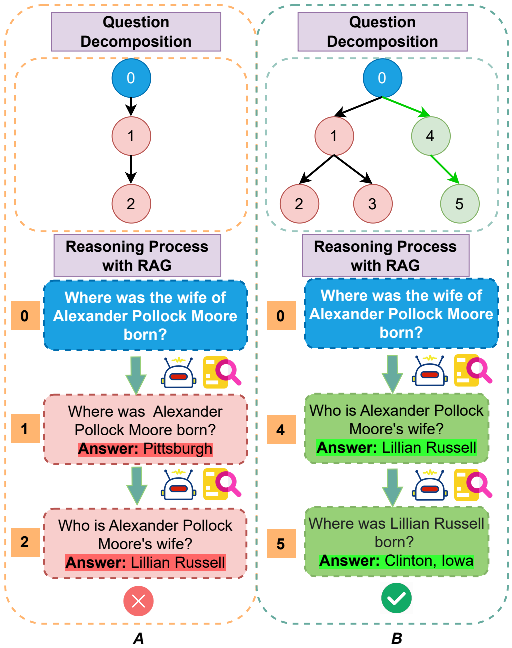

Figure 1: A illustrates the process of linearly decomposing a multi-hop problem step by step. It can be observed that any error in the intermediate steps will lead to an incorrect final answer. In contrast, B demonstrates the use of a tree-based search approach. Since the model possesses self-evaluation and exploration capabilities, it can abandon erroneous paths and select the correct one.

1 Introduction

In recent years, retrieval-augmented generation (RAG) has emerged as a key approach to addressing factual errors and hallucinations (Mallen et al., 2023; Min et al., 2023), as it provides up-to-date information for knowledge-intensive tasks (Chen, 2017; Petroni et al., 2021). However, for complex multi-hop questions, directly using RAG is challenging. On one hand, the complexity of natural language questions makes it difficult to decompose them. On the other hand, answering multi-hop questions requires a rigorous reasoning process and the ability to interact continuously with external knowledge bases. Many methods have been proposed to solve complex multi-hop problems: Self-Ask (Press et al., 2023) generates sub-questions step by step through self-questioning. IRCoT (Trivedi et al., 2023) interleaves retrieval with CoT generation to improve reasoning. LPKG (Wang et al., 2024) trains the model to parse the original complex question into different templates.

However, these methods are limited to linear,

* Corresponding author.
[1]https://github.com/pxspxspxs1/grat

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 27861–27875
July 27 - August 1, 2025 ©2025 Association for Computational Linguistics

left-to-right decision-making processes during inference, making it difficult to explore and perform strategic look-ahead for complex problems. At the same time, these methods lack the ability to self-evaluate and correct errors. If a mistake occurs in an intermediate step, it will lead to an incorrect final answer (Figure 1 A). Moreover, existing methods tend to focus only on historical information during reasoning, lacking exploration and evaluation of future steps. Additionally, due to the lack of supervised training for intermediate reasoning steps, models struggle to accurately decompose complex questions. Nevertheless, high-quality supervision data for intermediate steps requires costly annotation, making it difficult to collect.

To address the above issues, we propose GRAT (Figure 1 B), a method that: (1) Leverages the Monte Carlo Tree Search (MCTS) algorithm to explore the vast search space while self-evaluating each histroy reasoning step and future steps. It can select the most promising reasoning path within the constructed tree structure, allowing for the correction of erroneous exploration directions. (2) Utilizes fine-grained process rewards generated by GRAT to provide supervision signals for each step of the reasoning process. We collect the correct reasoning paths generated by GRAT as training data to perform self-training on the reasoning model, enhancing its ability to analyze complex questions. Through this unsupervised training approach, the reasoning model can continuously refine its ability to parse complex problems, and this self-update capability is challenging for previous methods. (3) Introduces a single-step simulation approach to efficiently complete rollouts, allowing the model to focus on future reasoning steps at the same time.

We conducted various experiments on four datasets: HotPotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and Bamboogle (Press et al., 2023), demonstrating the effectiveness of our approach. Additionally, we applied fine-tuning to self-train GRAT with the generated data. The experimental results show that GRAT achieves excellent performance in solving complex multi-hop problems both before and after training.

2 Method

2.1 Problem Setup

We first define this problem. Assume there is a language model M designed for a downstream question-answering task $T = \{\langle q, a\rangle\}$, where q represents a question and a represents the corresponding answer. A retrieval-augmented generation (RAG) model first retrieves relevant documents from a knowledge base D using a retriever R and then leverages them for answer generation. The process can be expressed as follows:

$$y = M(q, R(q, D)) \tag{1}$$

where y denotes the answer generated by the model based on the documents retrieved by R that are relevant to the question q.

In many previous works, the retrieval-augmentation process in this procedure is performed only once, making it difficult to answer questions that require multi-step reasoning. Given a complex multi-hop question q_m, answering it requires a series of reasoning steps $T = \{t_0, t_1, \ldots, t_n\}$, where each sub-step $t_i = (q_i, a_i)$ consists of a sub-question q_i and a sub-answer a_i. During the reasoning process, the sub-question q_i is often related to the sub-answer a_{i-1} from the previous step (Figure 1 B). Therefore, coherent multi-step retrieval-augmented generation will be key to answering such complex questions.

2.2 Multi-hop Question Inference and Reasoning Model

According to the introduction in Section 2.1, we assume that the model M used for inference should accomplish two tasks: (1) generating next sub-questions and (2) answering the sub-questions. In practice, M can be instantiated using different pretrained autoregressive models. Therefore, the generation of sub-questions can be expressed as

$$q_i = M(T_{0:i-1}, q) \tag{2}$$

And the answering of sub-questions can be expressed as

$$a_i = M(q_i, R(q_i, D)) \tag{3}$$

where (2) represents generating a new sub-question based on the history reasoning steps $T_{0:i-1}$ and the original question q. (3) represents generating the sub-answer based on the sub-question and the knowledge retrieved using the sub-question. Assuming the reasoning policy is π, our goal is to find the optimal performance of the following expression (4) in order to solve the multi-hop question.

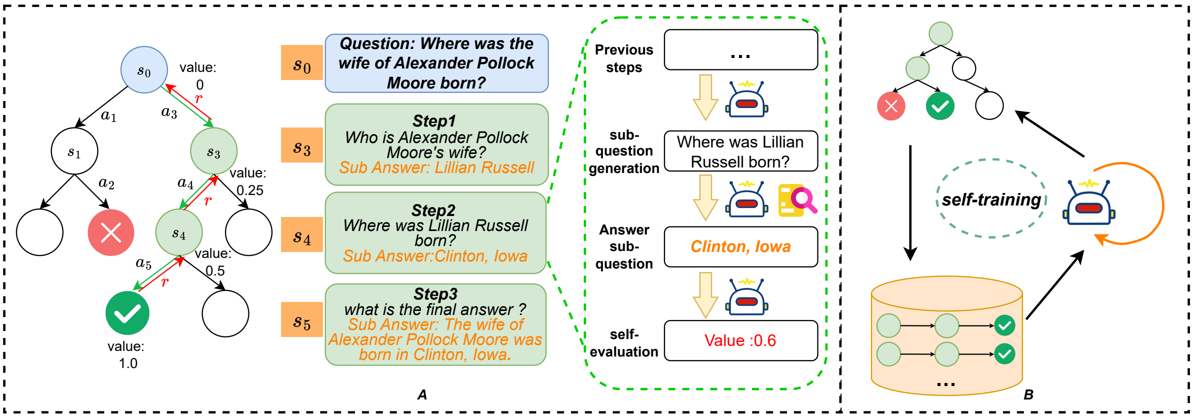

Figure 2: (1) Figure A illustrates the search process of GRAT. On the left, the constructed search tree is shown, where the path $s_0 \rightarrow s_3 \rightarrow s_4 \rightarrow s_5$ represents the final correct reasoning path. The figure presents each step's sub-question and its corresponding answer. On the right, the processing details for completing step s_4 are depicted. (2) Figure B demonstrates the self-training process of GRAT, where the model selects the correct reasoning path for self-update.

The tuple $(t_0, t_1, \ldots, t_K)$ represents the generated reasoning path, while a^* denotes the ground truth answer.

$$P_\pi(y = a^* \mid q) = \mathbb{E}_{(t_0,t_1,\ldots,t_K)\sim P_\pi(T|q)} [P(y = a^* \mid t_0, t_1, \ldots, t_K, q)] \quad (4)$$

2.3 Evaluation Model

The purpose of the evaluation model is to self-assess the feasibility of the already generated reasoning path. This helps the model evaluate whether the current reasoning step can help to solve the original question, thereby selecting a path more likely to lead to the correct solution. The evaluation model is represented as E, which can either be instantiated with the same model as the reasoning model or with a new model. It is expressed as:

$$v_i = E(T_i, q) \quad (5)$$

where the input consists of the current reasoning path T_i and the original problem q, and v_i represents the value of the current branch.

2.4 Monte Carlo Tree Search based RAG

In the process of complex problem answering, we use Monte Carlo Tree Search (MCTS) to progressively decompose the problem. It constructs a tree-like reasoning framework, where each node represents the state, which contains the completed history reasoning paths from root. And the transition from one node to another represents an action, which includes the following steps: generating the next sub-question, retrieving external knowledge to answer the sub-question, and forming a new state.

Our MCTS-based model evaluates and scores the reasoning path based on the current reasoning step and the future simulation results. It then selects more valuable paths according to the evaluation scores, balancing exploitation and exploration, and efficiently finding high-reward reasoning paths. The algorithm will perform multiple iterations until a computational budget is reached. The following part will introduce the components of the algorithm.

Selection. The first phase is the selection phase, where the search begins from the root node (s_0 in figure 2 A). In each selection iteration, the next node is chosen based on the children's values, in order to identify more promising nodes for the next expansion step. The selection phase ends when a leaf node is reached. During this process, we follow the method of Zhang et al. (2024), using UCB (Upper Confidence Bound) to select nodes, balancing exploitation and exploration, which is as follows:

$$UCB(child) = v_{child} + w\sqrt{\frac{2 \cdot \ln n_{parent}}{n_{child}}} \quad (6)$$

Here, v_{child} represents the value of the child node, while n_{parent} and n_{child} represent the number of times the parent and child nodes have been visited, respectively. w is a constant used to control the weight between exploitation and exploration. During selection, the child node with the highest UCB value is chosen each time. This approach allows for selecting the most valuable child node while also balancing the exploration of unknown nodes.

Expansion. After the selection phase is completed, the current node is an unexplored leaf node. Given the current node's state s_i, the reasoning model will generate d new actions, which are the next sub-questions $q_{i+1} = M(T_{0:i}, q)$. Once the sub-question is obtained, the retriever R is called to retrieve documents related to the sub-question. These documents, along with the sub-question, are then fed back into the reasoning model to generate the sub-answer: $a_{i+1} = M(q_{i+1}, R(q_{i+1}, D))$.

After completing the above steps, our method will invoke the evaluation model E in 2.3 to self-assess the newly generated child node. It is important to note that we set a threshold l (0.9 in the experiment), and if the evaluation score exceeds l, the search will be prematurely terminated. Finally, we generate d new child nodes $\{(q^1_{i+1}, a^1_{i+1}, v^1_{i+1}), ..., (q^d_{i+1}, a^d_{i+1}, v^d_{i+1})\}$. The complete expansion process is shown in Figure 2 A.

Simulation. The simulation phase is designed to evaluate the expected future rewards, providing an assessment of the current node's value from a future perspective. After expansion, we select the child node with the highest value for simulation. Previous methods, such as those in Hao et al. (2023) and Zhou et al., have used iterative generation and evaluation rollout methods, but these approaches incur significant generation and time costs. Therefore, we use a one-step rollout approach, where the reasoning model generates all the subsequent sub-questions at once, retrieves the relevant documents, and then answers the original question based on the documents. Finally, all the reasoning steps are input into the evaluation model to assess the value of this path. Assuming the value of the node being rolled-out is v_i, and the evaluation value obtained after simulation is v'_i, we use formula (7) to update the the original value of the node. At this point, the updated value of the node takes into account both the historical reasoning path and the future reasoning path.

$$v_i = v_i \cdot (1 - \alpha) + v'_i \cdot \alpha \tag{7}$$

Where α represents the parameter that controls the update.

Back-propagation. When simulation is completed, every node from the root to the leaf node with simulation has been visited once. Therefore, the visit count of all nodes along this path needs to be updated as $n_i = n_i + 1$. At the same time, since the value of the child node has changed, the value of its parent node should also change. Thus, we perform a backward update of the values of the nodes along this path, starting from the leaf node:

$$v_{parent} = \frac{\sum_{i=1}^{d} n_{child_i} \cdot v_{child_i}}{\sum_{i=1}^{d} n_{child_i}} \tag{8}$$

Here, d represents the number of child nodes, n_{child} represents the number of times the child node has been visited, and v_{child_i} represents the value of the child node.

Final answer generation. When the computational budget is reached, or the search is prematurely terminated, the process will move to the final answer generation stage. If the computational budget is reached, we start from the root node and use a greedy strategy to find the highest-value path as the final answer path T^*. If the value of a node exceeds the threshold l, leading to early termination, the path containing that node will be taken as the answer path T^*. Then, this answer path is input into the reasoning model to generate the final answer, as shown in equation (9). Here, y represents the final predicted answer.

$$y = M(T^*, q) \tag{9}$$

2.5 Reasoning model self-training

In order for the model to generate more reasonable reasoning paths, we can use the generated correct reasoning paths as training data to fine-tune the reasoning model (Figure 2 B). We filter the paths based on the correctness of the predicted answers and the value of the reasoning paths, selecting the paths with correct answers while also having higher values as the training data. Let the original training dataset be $\mathbb{D}_{train}$, where $(q, a^*) \in \mathbb{D}_{train}$. By filtering the training data, we obtain the supervised fine-tuning dataset: $(q, a^*, T, D) \in \mathbb{D}_{\text{SFT}}$, where each data entry contains a reasoning path T that leads to the correct answer and a set of documents D relevant to each sub-question. During training, we use the standard supervised fine-tuning method and the following loss function:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} logP(r_i|h, r_{<i}) \tag{10}$$

Here, h is the input, which consists of the concatenation of information such as the question, subpaths, documents, and prompts. r represents the target output, and N denotes the length of r.

3 Experiments

3.1 Experimental Setting

Datasets. To evaluate the ability of different models to answer complex questions, we selected four complex question answering datasets: (1) HotPotQA (Yang et al., 2018), (2) 2WikiMultiHopQA (Ho et al., 2020), (3) MuSiQue (Trivedi et al., 2022), and (4) Bamboogle (Press et al., 2023). Among these datasets, HotPotQA, 2WikiMultiHopQA, and MuSiQue include training, validation, and test sets, while Bamboogle is a smaller dataset consisting of only 125 test examples. All four datasets require reasoning over multiple different Wikipedia paragraphs to answer the questions. Following Wang et al. (2024) and Shao et al. (2023), we randomly selected 500 samples from the HotPotQA, 2WikiMultiHopQA, and MuSiQue datasets for testing. For Bamboogle, we used all 125 test examples. To ensure fair comparison, all methods employed a simple sparse retrieval method: BM25 (Robertson and Walker, 1994), which is a classic method based on term frequency statistics.

Evaluation metrics. Following Wei et al. (2024), we use accuracy (acc) as the evaluation metric, which measures whether the ground-truth answers are included in the model generations (Mallen et al., 2023,Schick et al., 2023).

Baselines. We used a total of five large language models with different parameter sizes as both the reasoning model and evaluation model (they are instantiated using the same model in our setup). These models include DeepSeek-R1(DeepSeek-AI et al., 2025), GPT-3.5-turbo (Ouyang et al., 2022), Llama3-Instruct-8B (AI@Meta, 2024), Qwen2.5-Instruct-14B (Yang et al., 2024), and Llama3-Instruct-70B(GPTQ INT4).

We use the following methods as baseline models: (1) **No Retrieval**: This approach directly utilizes the backbone LLM for reasoning without relying on external knowledge. It solely depends on the model's internal knowledge and reasoning capabilities. (2) **With Retrieval**: In this approach, the model retrieves relevant documents from the external knowledge base using the given question and then uses the backbone LLM to perform reasoning grounded in these retrieved documents. (3) **ToT** (Yao et al., 2023): The Tree of Thoughts method is an approach that uses a tree structure for reasoning over complex problems. We use Breadth-First Search (BFS) as the search algorithm. For efficiency reasons, we set the maximum depth to 4 and the width to 3. (4) **InstructRAG** (Wei et al., 2024): This method leverages the reasoning ability of LLMs to filter out the necessary documents for inference through self-synthesized rationales and generates a reasoning path. It also requires retrieving external knowledge. (5) **ActiveRAG** (Xu et al., 2024): This approach mimics human learning through a multi-agent framework, enabling it to comprehend retrieved knowledge from multiple perspectives and complete the reasoning process.

3.2 Main Results

Table 1 provides the main results of our experiments: Firstly, it can be observed that, as the model parameters increase, the overall ability of the model to solve complex problems improves. Additionally, using external retrieval significantly outperforms methods that do not rely on retrieval. Specifically, for Llama3-Instruct-8B, Qwen2.5-Instruct-14B, and Llama3-Instruct-70B, the With Retrieval method achieves an average improvement of 35.4%, 51.7%, and 6.8% over the No Retrieval method respectively, across the four datasets.

Our model, GRAT, outperforms baseline models on the 2WikiMultiHopQA, HotPotQA, and MuSiQue datasets when using Llama3-Instruct-8B, Qwen2.5-Instruct-14B, and Llama3-Instruct-70B as backbone models. Specially, on the 2WikiMultiHopQA dataset, GRAT achieves improvements of 4.8%, 7.3%, and 17.5% over the second-best model across the three different backbone settings. However, on the Bamboogle dataset, our method does not surpass all baseline models, which may be due to the limited dataset size (only 125 test samples). Nevertheless, GRAT still demonstrates competitive performance.

Additionally, we compare our approach with the latest LLMs accessible via their APIs, such as GPT3.5-Turbo[2] and DeepSeek-R1[3]. Notably, GRAT only based on Llama3-Instruct-8B can significantly outperform GPT-3.5-Turbo (both No Retrieval and With Retrieval) and achieves performance comparable to DeepSeek-R1. Using a backbone model with a larger number of parameters will achieve better results. These results demonstrate that our method is highly effective in tackling complex multi-hop reasoning tasks.

[2] https://platform.openai.com

[3] https://platform.deepseek.com

Table 1: Performance of GRAT and other baselines on the four datasets, with the best values highlighted in **bold**.

Method	Model	Datasets			
		2MultiHopQA	HotPotQA	Bamboogle	MuSiQue
No Retrieval	DeepSeek-R1	53.2	45.4	**61.6**	22.2
No Retrieval	GPT-3.5	32.8	33.6	33.6	9.4
With Retrieval		38.0	42.4	15.2	9.6
No Retrieval	Llama3-Instruct-8B	30.4	22.8	15.2	4.8
With Retrieval		36.2	38.4	15.2	7.4
ToT		57.8	49.4	32.8	18.4
InstructRAG-ICL		51.4	51.4	32.0	15.2
ActiveRAG		53.2	51.2	45.0	15.8
GRAT		60.6	50.2	32.8	20.2
No Retrieval	Qwen2.5-Instruct-14B	30.0	24.8	28.8	6.8
With Retrieval		46.4	46.2	24.0	12.4
ToT		60.2	57.8	46.4	22.2
InstructRAG-ICL		53.4	53.0	45.6	18.8
ActiveRAG		58.8	56.8	52.0	22.6
GRAT		64.6	60.0	40.8	25.8
No Retrieval	Llama3-Instruct-70B	33.2	31.8	33.6	8.0
With Retrieval		34.8	41.8	27.2	9.8
InstructRAG-ICL		60.4	59.6	50.4	24.2
ActiveRAG		64.0	58.0	59.0	26.6
GRAT		**75.2**	**61.8**	52.0	**29.2**

3.3 Self-Training

As mentioned in Section 2.5, we use GRAT to perform reasoning and constructing a search tree, and score each sub-path, which can generate reasoning paths with process rewards. We then select the highest-scoring path among all correct paths into training dataset while filtering out noisy paths (e.g., those where the output contain "No relevant information found in the document"). We generate training data using the train set of 2WikiMultiHopQA, ensuring that the data for generating does not overlap with the test set. In our experiments, we generate a total of 11,459 training samples and use Llama3-Instruct-8B as the reasoning model, applying LoRA for instruction fine-tuning. We use InstructRAG as baseline, which includes both an In-Context Learning version (No Training) and a Fine-Tuning version (With Training). For the fine-tuning version, we used the publicly available model weights provided by Wei et al. (2024) for testing. The final results are shown in Table 2.

From the results in Table 2, we can observe that both InstructRAG and GRAT show significant improvement after training. GRAT achieves an increase of 10.2% compared to the performance of its No-training version, indicating that self-training can significantly enhance the model's reasoning and comprehension abilities. Additionally, GRAT (With Training) outperforms InstructRAG (With Training) by 12.5%, demonstrating that our model still achieves better performance than baselines after training.

Table 2: Performance of GRAT and InstructRAG before and after self-training on the 2MultiHopQA dataset. The backbone model is Llama3-Instruct-8B.

Method	Datasets
	2MultiHopQA
InstructRAG(No Training)	51.4
GRAT(No Training)	60.6
InstructRAG(With Training)	59.4
GRAT(With Training)	**66.8**

3.4 Influence of Computational Budget

Next, we experimented to find the impact of the search iterations on the accuracy of the responses. One search iteration means starting from the root node and sequentially completing selection, expansion, simulation, and back-propagation. We set the computational budget from 1 to 10 and used Llama3-Instruct-8B as both the reasoning and eval-

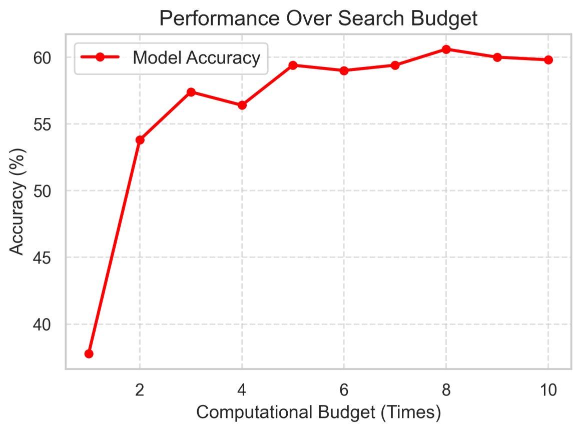

Figure 3: Performance over Different Computational Budget

uation model to conduct experiments on the 2WikiMultiHopQA dataset. The results are shown in Figure 3. It can be observed that the accuracy increases as the number of iterations increases. Specifically, the accuracy rises rapidly when the number of searches increases from 1 to 3. This is because most questions in the dataset require two- or three-hop reasoning, making the accuracy highly sensitive to the number of searches. The accuracy peaks at 60.6 when the number of searches reaches 8, likely due to the fact that more complex questions require multiple iterations of reasoning attempts to ultimately derive the correct answer.

3.5 Ablation

Table 3: Results of various ablation experiments.

Method	**Model**	**Datasets**
		2MultiHopQA
Base(No Retrieval)		30.4
w/o Simulate		57.6
w/o Retrieval	Llama3-Instruct-8B	38.8
w. Gold-docs		**70.4**
GRAT		60.6

In the ablation experiments, we made the following adjustments to GRAT to evaluate the impact of each module on the final performance:

• **w/o Simulation**: This indicates the ablation of the original simulation module in GRAT, meaning the model's ability to evaluate the future has been removed. It can be observed that after removing the **Simulation** module, the accuracy of GRAT on 2WikiMultiHopQA drops from 60.6 to 57.6, indicating that estimating future reasoning steps helps in better solving complex problems.

• **w/o Retrieval**: This removes the retrieval module in GRAT, meaning the model can rely solely on its reasoning ability and internal knowledge during reasoning. For comparison, we also present a baseline, **Base(No Retrieval)**, in Table 3, where the base model directly answers the original question without using external retrieval. We can observe that **w/o Retrieval** shows a significant performance drop compared to the full GRAT, indicating that external retrieval plays a crucial role in answering complex questions. Meanwhile, **w/o Retrieval** achieves an 8.4 accuracy improvement over **Base (No Retrieval)** under the same condition of no external retrieval, demonstrating the performance gain brought by our tree-based search method.

• **w. Gold-docs**: This replaces the retrieved documents in GRAT with gold documents, which contain all the necessary information to answer the original question. This ensures that the model's performance is not constrained by missing external information, demonstrating the upper bound of our model's potential performance. We can observe that under this condition, our model achieves an accuracy of 70.4. This demonstrates the great potential of GRAT in solving complex multi-hop problems.

3.6 Case Study

Figure 4 illustrates a search tree constructed using GRAT (Qwen2.5-Instruct_14B). The numbers indicate the order of visits. Node 0 represents the question and its correct answer. Starting from this root node, nodes 1 and 2 are expanded first. Notably, an error occurs when answering the sub-question at node 1. However, in the next search step, the model identifies this factual error and corrects it at nodes 3 and 4. As a result, in the subsequent exploration, the model selects the higher-value path: 0-2-5. Ultimately it arrives at the correct answer. In addition, the Appendix A presents the templates of the prompts we used.

4 Related Work

Retrieval-Augmented Generation. Retrieval-augmented generation (RAG) is a widely used technique across multiple areas that retrieves external knowledge to obtain the latest and up-to-date information. By providing LLMs with relevant information, RAG helps LLMs generate more accurate and useful content(Asai et al., 2023a; Chen et al., 2023; Asai et al., 2023b; Jiang et al., 2023; Shao et al.,

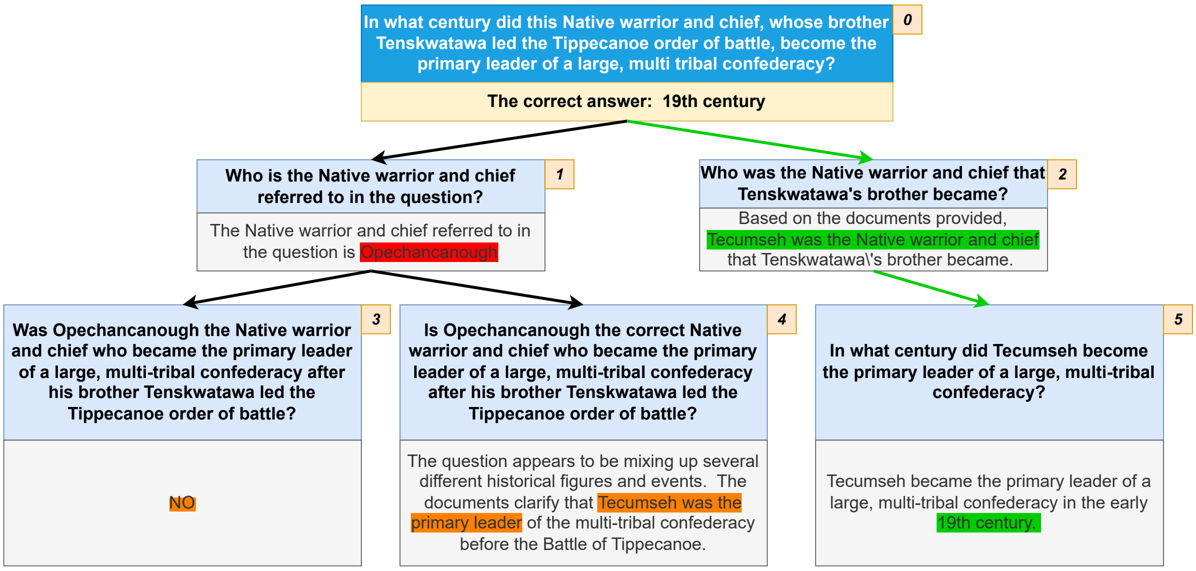

Figure 4: The detailed reasoning process by GRAT. The root node (Node 0) includes the original question and the correct answer. In other nodes, the upper part of the node represents the generated sub-questions, and the lower part shows the corresponding answers. The numerical labels indicate the order of visits.

2023). At the same time, RAG can also help mitigate the hallucination problem commonly found in LLMs(Achiam et al., 2023; Guu et al., 2020; Lewis et al., 2020). Many works have attempted to optimize different stages of this process,for example, (Yoran et al., 2023; Wang et al., 2023; Yu et al., 2023) enhance model performance by reducing noise in relevant documents and improving the model's robustness to irrelevant content. Chen et al. (2023), Jeong et al. (2024) and Asai et al. (2023b) try to avoid irrelevant retrieval by adjusting the granularity and timing of retrieval. Some works also focus on optimizing prompts and queries (Chan et al., 2024; Ma et al., 2023). Press et al. (2023) proposes a Self-ask approach, where the model continuously asks itself (and answers) follow-up questions to analyze complex problems. This essentially serves as an improved strategy for the chain-of-thought method (Wei et al., 2022). Wang et al. (2024) trains the model to parse the question into a fixed template (plan), which can then be further decomposed into sub-questions. The LLM only needs to answer each sub-question in sequence.

Large Language Model Reasoning. To answer complex questions, many reasoning methods have been proposed. For example, Chain of Thought (CoT) (Wei et al., 2022) try to generate intermediate multi-step reasoning steps to provide a step-by-step solution for complex problems. Self-Consistency (Wang et al., 2022) generates multiple reasoning steps using LLMs and selects the one with the highest score, thereby improving the reliability of the answer. Tree of Thoughts(ToT) (Yao et al., 2024) improves upon Chain of Thought by transforming linear reasoning into a tree structure, allowing multiple reasoning paths to be explored simultaneously, leading to more comprehensive thinking. Hao et al. (2023) employed Monte Carlo Tree Search to construct the reasoning tree under the guidance of LLMs and rewards, enhancing the model's ability to select and evaluate paths while balancing exploitation and exploration.

Large Language Model Training. To align the content generated by LLMs with human preferences, instruction-tuning can be performed using datasets that contain instructions and human-written completions(Mishra et al., 2022; Sanh et al., 2022; Chung et al., 2024). However, compared to directly constructing human preference data, it is easier to judge the relative quality of data for humans. Therefore, some works first optimize a neural network reward function and then fine-tune the language model using reinforcement learning (RL) algorithms (Ramamurthy et al., 2022; Kreutzer et al., 2018). Another approach is to use LLMs fine-tuned with human feedback to generate additional synthetic preference data, which is then used to further fine-tune the original model (Bai et al., 2022; Zhang et al., 2024).

5 Conclusion

In this paper, we propose a novel retrieval-augmented generation model called GRAT, which is based on Monte Carlo Tree Search. GRAT possesses the capabilities of multi-path exploration, strategic look-ahead, stepwise evaluation, and global selection, while also balancing exploitation and exploration during the search process. Compared to single-chain RAG methods, it offers significant advantages. Additionally, GRAT can perform self-training using high-quality stepwise reasoning data generated by itself, continuously refining its problem analysis capabilities. We conducted extensive experiments on multiple datasets, demonstrating the effectiveness and superiority of our model.

Limitations

Our work has some limitations. For example, during training, we only used correct data and did not utilize the lower-quality generated data. In future approaches, we will explore using this data for preference optimization. This is because, even though incorrect sub-questions may not directly contribute to the solution, they can help the model recognize which actions lead to lower-value outcomes.

Additionally, for self-training, we conducted experiments only on the 8B model. In the future, we will explore the impact of self-training on performance with larger-scale models.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant No.62276110, No.62172039 and in part by the fund of Joint Laboratory of HUST and Pingan Property & Casualty Research (HPL). The authors would also like to thank anonymous reviewers for their comments on improving the quality of this paper.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

AI@Meta. 2024. Llama 3 model card.

Akari Asai, Sewon Min, Zexuan Zhong, and Danqi Chen. 2023a. Retrieval-based language models and applications. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 6: Tutorial Abstracts)*, pages 41–46.

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

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. 2022. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.

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*.

D Chen. 2017. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

Tong Chen, Hongwei Wang, Sihao Chen, Wenhao Yu, Kaixin Ma, Xinran Zhao, Hongming Zhang, and Dong Yu. 2023. Dense x retrieval: What retrieval granularity should we use? *arXiv preprint arXiv:2312.06648*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao

Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

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.

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.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. 2024. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7029–7043.

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

Julia Kreutzer, Joshua Uyheng, and Stefan Riezler. 2018. Reliability and learnability of human bandit feedback for sequence-to-sequence reinforcement learning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1777–1788.

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

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.

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

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. Factscore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3470–3487.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, et al. 2021. Kilt: a benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2523–2544.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711.

Rajkumar Ramamurthy, Prithviraj Ammanabrolu, Kianté Brantley, Jack Hessel, Rafet Sifa, Christian Bauckhage, Hannaneh Hajishirzi, and Yejin Choi. 2022. Is reinforcement learning (not) for natural language processing: Benchmarks, baselines, and building blocks for natural language policy optimization. *arXiv preprint arXiv:2210.01241*.

Stephen E Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR'94: Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, organised by Dublin City University*, pages 232–241. Springer.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2022. Multitask prompted training enables zero-shot task generalization. In *ICLR 2022-Tenth International Conference on Learning Representations*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9248–9274.

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

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

Junjie Wang, Mingyang Chen, Binbin Hu, Dan Yang, Ziqi Liu, Yue Shen, Peng Wei, Zhiqiang Zhang, Jinjie Gu, Jun Zhou, et al. 2024. Learning to plan for retrieval-augmented large language models from knowledge graphs. *arXiv preprint arXiv:2406.14282*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Zhiruo Wang, Jun Araki, Zhengbao Jiang, Md Rizwan Parvez, and Graham Neubig. 2023. Learning to filter context for retrieval-augmented generation. *arXiv preprint arXiv:2311.08377*.

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.

Zhepei Wei, Wei-Lin Chen, and Yu Meng. 2024. Instructrag: Instructing retrieval-augmented generation via self-synthesized rationales. *Preprint*, arXiv:2406.13629.

Zhipeng Xu, Zhenghao Liu, Yukun Yan, Shuo Wang, Shi Yu, Zheni Zeng, Chaojun Xiao, Zhiyuan Liu, Ge Yu, and Chenyan Xiong. 2024. Activerag: Autonomously knowledge assimilation and accommodation through retrieval-augmented agents. *Preprint*, arXiv:2402.13547.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

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. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

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.

Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. 2023. Making retrieval-augmented language models robust to irrelevant context. *arXiv preprint arXiv:2310.01558*.

Wenhao Yu, Hongming Zhang, Xiaoman Pan, Kaixin Ma, Hongwei Wang, and Dong Yu. 2023. Chain-of-note: Enhancing robustness in retrieval-augmented language models. *arXiv preprint arXiv:2311.09210*.

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*.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning acting and planning in language models. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*.

A Prompting Template

We present the prompt templates for reasoning processes, subquestion answering, and self-evaluation in Figures 5 , 6, and 7, respectively. These prompts include example cases to help the model generate more effectively.

PROMPT FOR REASONING STEPS

INSTRUCTION
Given the following complex multi-hop question, you need to step through and determine what questions to ask at each step. The input may already include some steps of sub-questions, and you need to continue generating the next sub-question.

EXAMPLES
Question:
Which film came out first, Brudebuketten or Vibes (Film)?
Reasoning steps:
Step 1: when did film Brudebuketten come out?
Answer 1: 1953
Output: <when did film Vibes (Film) come out?>

Question:
Which film has the director who was born later, Glamour Boy (Film) or Night By The Seashore?
Reasoning steps:
Step 1: Who is the director of Glamour Boy (Film)?
Answer 1: Ralph Murphy
Step 2: when did director Ralph Murphy born?
Answer 2: May 1, 1895
Step 3: Who is the director of Night By The Seashore?
Answer 3: Erkko Kivikoski
Step 4: when did director Erkko Kivikoski born?
Answer 4: 2 July 1936
Output: <Which film has the director who was born later, Glamour Boy (Film) or Night By The Seashore?>

INPUT
Next, please continue to generate the next sub-question for the following question. Note that, you should put your output in <>, like: < your sub-question>
Question:
{##QUESTION}
Reasoning steps:
{##HISTORY REASONING STEPS}
Output:

Figure 5: Prompt for reasoning

PROMPT FOR SUB-QUESTION ANSWERING

INSTRUCTION
Given an original complex multi-hop question, your task is to refer to the original reasoning steps and relevant document content to answer the current sub-question. Note that you do not need to answer the original question or sub-questions that have already been answered.

EXAMPLES
Original question: When did Charles Mathew's father die?
Reasoning steps:
Let's think step by step about the sub-questions that need to be queried.
Step 1: Who is Charles Mathew's father ?
Answer 1: James Mathews
Step 2 :When did James Mathews die?
Documents: Charles was born to James Mathews (died 1804), a Wesleyan Methodist bookseller, printer, and pharmacist on the Strand, who also served as minister in one of the Countess of Huntingdon's chapels. Charles was educated at Merchant Taylors' School in London, which had some openings for common boys. He was next apprenticed to his father. For religious reasons, the father forbade his children from visiting theatres.
Current sub-question: When did James Mathews die?
Output: 1804

INPUT
Note: Please try to use the content from the original documents to answer, and provide a concise response without any analysis. Output no more than 15 words.
Original question:{##QUESTION}
Reasoning steps:
{##HISTROY REASONING STEPS}
Documents:
{##DOCUMENTS}
Current sub-question:{##SUB-QUESTION}
Output:

Figure 6: Prompt for sub-question answering

PROMPT FOR SELF-ESTIMATION

INSTRUCTION
Given a complex multi-hop question, answering this question may require answering multiple sub-questions. Your task is to determine, based on the given complex question and existing reasoning steps, whether these existing reasoning steps can help to solve some part of the problem and provide a score. The score should be a decimal between 0 and 1. If all sub-questions are fully raised and answered, the score is 1. If no relevant sub-questions are resolved, the score is 0. The more sub-questions that help answer the original question are resolved, the closer the score is to 1; the fewer are resolved, the closer the score is to 0. First, provide a relevant analysis, then give the score. Do not generate additional solving steps, only output scores to evaluate the current steps. Please learn from the following example:

EXAMPLES
question 1:
Which film came out first, Brudebuketten or Vibes (Film)?
reason steps:
Let's think step by step about the sub-questions that need to be queried.
Step 1: when did film Brudebuketten come out?
Answer 1: 1953
Step 2: when did film Vibes (Film) come out?
Answer 2: 1988
analysis:
To solve this problem, the first step is to answer when the movie Brudebuketten was released, the second step is to answer when the movie Vibes (Film) was released, and the third step should be to compare the two release dates and determine which movie was released first, need 3 steps. However, in the reasoning steps above, only the first two steps were completed, and the reasoning for the final answer was not finished. Therefore, the final score is <0.66>.

INPUT
Attention, you must put your score in <>, like <score>. Next, you need to analyze the following reasoning steps:

question:
{##QUESTION},
reason steps:
{##HISTORY REASONING STEPS},
analysis:

Figure 7: Prompt for self-estimation