HRM: Enhancing Agentic Retrieval Augmented Generation With Hierarchical Reward Modeling

Anonymous ACL submission

Abstract

Retrieval-Augmented Generation (RAG) systems have emerged as a pivotal methodology for enhancing Large Language Models (LLMs) through the dynamic integration of external knowledge. To further improve RAG's flexibility, Agentic RAG introduces autonomous agents into the workflow. However, Agentic RAG faces several challenges: (1) the success of each step depends on both high-quality planning and accurate searching, (2) the lack of supervision for intermediate reasoning steps, and (3) the exponentially large candidate space for planning and searching. To address these challenges, we propose **HRM**, a novel framework that decouples planning and search processes using dual value models, enabling independent optimization of planning reasoning and search grounding. Our approach constructs a reasoning tree, where each node represents planning and searching steps. We leverage Monte Carlo Tree Search to efficiently assess the quality of each step. During inference, Hierarchical Beam Search iteratively refines plan and search candidates through reward-guided optimization. Extensive experiments on five datasets, across policy models of varying parameter sizes, demonstrate the effectiveness of our method.

1 Introduction

Large Language Models (LLMs) (Taylor et al., 2022; Chowdhery et al., 2022; Zhao et al., 2023) have demonstrated remarkable performance across a wide range of downstream tasks (Xia et al., 2024; Yamauchi et al., 2023; Imani et al., 2023; Lewkowycz et al., 2022). Despite these advancements, LLMs remain susceptible to generating responses that include hallucinated facts (Ji et al., 2023; Shuster et al., 2021; Zhang et al., 2023), undermining their reliability. To address this challenge, Retrieval-Augmented Generation (RAG) has been proposed, integrating external knowledge to enhance the generation process (Ram et al., 2023; Shi et al., 2023; Rashkin et al., 2021; Gao et al., 2022; Bohnet et al., 2022; Menick et al., 2022).

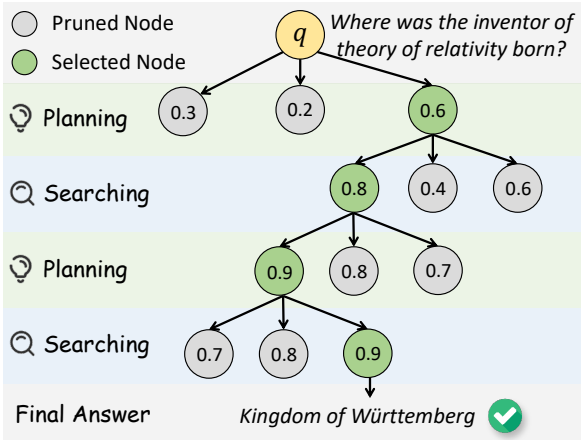


Figure 1: The illustration of Hierarchical Beam Search. During planning, the policy model generates and prunes candidate plans using the planning value model. During searching, it generates queries, retrieves documents, and prunes results using the search value model. This process iterates until the final answer is reached.

While RAG systems have led to significant improvements, they still face important limitations. These systems rely on static workflows and struggle to effectively handle multi-step reasoning or complex tasks. A promising solution to these limitations is Agentic Retrieval-Augmented Generation (Agentic RAG), which introduces autonomous AI agents into the RAG pipeline (Asai et al., 2023; Yu et al., 2024; Chen et al., 2024c; Li et al., 2025). In this framework, the reasoning process typically involves two phases: planning and searching. During the planning phase, the agent analyzes the current reasoning process and determines which information is still required. In the search phase, the agent generates search queries to retrieve relevant external documents. These phases alternate iteratively until a final answer is produced.

Although Agentic RAG shows superior performance, it faces several inherent challenges: (1) The

success of the reasoning process depends not only on high-quality planning but also on the accuracy of the retrieved information. While planning can be improved with high-quality training data and sophisticated pipelines, ensuring accurate retrieval remains challenging, as it depends on both the quality of the generated queries and the retrieval system's performance. (2) Evaluating the quality of each reasoning step is difficult due to the lack of explicit supervision signals. Most RAG datasets only provide final answers without feedback on intermediate steps, making it hard to assess and improve the quality of individual reasoning stages. (3) The expansive candidate space for planning and searching creates a large, computationally intensive searching space, making it challenging to efficiently identify optimal paths.

To address these challenges, we propose **HRM** a novel Agentic RAG framework with **H**ierarchical **R**eward **M**odeling. **To enhance the success probability of each reasoning step**, we introduce planning exploration and searching exploration phases. The policy model generates multiple potential plans, which are evaluated by a plan value model to select the most promising options. Based on these plans, the policy model generates multiple queries to retrieve relevant documents. These search results are then ranked by the search value model to ensure the reliability of the retrieval process. **To efficiently assess the quality of each reasoning step**, we introduce Monte Carlo Tree Search (MCTS) (Silver et al., 2017) to guide the exploration of potential reasoning paths. During MCTS simulations, the LLM acts as a Judge to evaluate the quality of both the planning and search results, separately. Through iterative MCTS simulations, the rewards derived from final answer correctness are back-propagated to update the LLM scores, refining the LLM's scores and correcting potential inaccuracies. **To combat the exponential search space**, we propose pruning the planning and search spaces using a planning value model and a search value model. These models are trained on reward signals derived from the reasoning tree constructed through MCTS annotation. During inference, we employ Hierarchical Beam Search. At each step, the policy model generates multiple plans, which are evaluated by the planning value model to retain only the most promising ones. Based on these plans, the policy model generates search queries to retrieve relevant documents. The search value model then evaluates the retrieved results, preserving only the most valuable ones. This iterative process continues until either the maximum depth is reached or no further nodes can be expanded, ensuring efficient and effective reasoning.

To summarize, our contributions can be summarized as follows:

- We introduce HRM, a novel Agentic RAG framework that decouples planning-search processes with dual value models, enabling independent optimization of planning reasoning and search grounding.
- We propose improving the success rate of each step by fully exploring the planning and search spaces. We utilize MCTS to accurately assess planning and search quality, while Hierarchical Beam Search is employed to efficiently prune the exponential candidate space.
- Extensive experiments on five datasets across policy models of different parameter sizes demonstrate the effectiveness of our method.

2 Background

In Agentic RAG, given a user query q, the policy model conducts multi-step reasoning and retrieves external knowledge to produce the final answer. Each step typically involves two stages: planning and searching. In the planning stage, the policy model $\mathcal{M}$ reasons based on the interaction history τ_{t-1} and generates a plan p_t:

$$p_t = \mathcal{M}(\tau_{t-1}),$$

where $\tau_{t-1} = \{q, p_1, q_1, d_1, \ldots, p_{t-1}, q_{t-1}, d_{t-1}\}$ represents the previous reasoning path.

In the searching stage, the policy model generates search queries and retrieves external documents using an off-the-shelf search engine:

$$q_t = \mathcal{M}(\tau_{t-1}, p_t), \quad (1)$$

$$d_t = \text{Retrieve}(q_t), \quad (2)$$

where d_t denotes the retrieved documents.

The success of each step depends on two interdependent factors: the quality of the planning and the precision of the searching. While planning quality can be improved through high-quality training data, retrieval accuracy is subject to uncertainties due to challenges in query formulation and retriever performance.

To enhance the success rate of each step, we encourage the policy model to fully explore both the planning and search spaces through sampling.

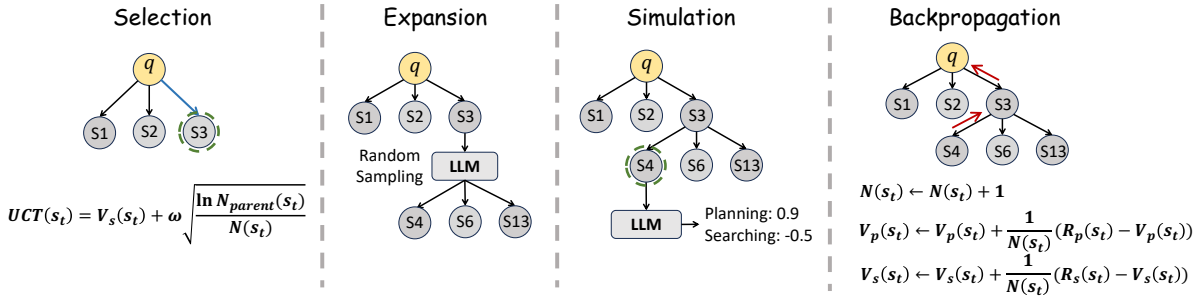


Figure 2: Single iteration of MCTS Annotation. The iteration is repeated until the maximum number of iterations is reached or no further nodes in the tree can be expanded.

These sampled paths are then refined using a planning value model and a searching value model, respectively, ensuring more accurate outcomes.

3 Approach

3.1 Overview

Figure 3 presents an overview of our framework. The process begins with the application of Monte Carlo Tree Search (MCTS) to construct a reasoning tree for the queries in the training dataset. Each node in the tree represents a reasoning step, encompassing both planning and search results. From these trees, we extract both correct and incorrect paths, which are subsequently utilized to train the policy model and the value models, respectively. During the inference phase, we introduce a hierarchical beam search algorithm, where at each layer, the policy model fully explores the planning and search spaces, and the value models select the best candidate for further refinement.

3.2 MCTS Annotation

During MCTS annotation, we prompt the LLM to generate plans and search queries, interactively collaborating with the retriever to iteratively expand the reasoning tree. The process runs for multiple simulations and terminates when the maximum iteration number is reached, or no further paths can be expanded. For the i-th simulation, MCTS conducts four operations to expand the tree:

Selection The i-th simulation begins with s_0, representing the input query. The algorithm selects nodes according to the Upper Confidence Bound for Trees (UCT) criterion (Rosin, 2011):

$$UCT(s_t) = V_s(s_t) + w\sqrt{\frac{\ln N_{parent}(s_t)}{N(s_t)}} \tag{3}$$

where $V_s(s_t)$ represents the reward of the search result, and w controls the balance between exploration and exploitation. The reason we choose $V_s(s_t)$ to calculate the UCT score is that the quality of the search results serves as a more reliable indicator of a step's potential to arrive at the correct answer.

Expansion After selecting the node to be expanded, the LLM generates the next plan and query based on the reasoning status. For simplicity, assume the chosen node s_t corresponds to the intermediate reasoning trajectory τ_{t-1}. The expansion process is as follows:

$$p_t, q_t = \text{LLM}(\tau_{t-1}) \tag{4}$$

$$d_t = \text{Retrieve}(q_t) \tag{5}$$

To ensure diversity, we employ sampling generation with a higher temperature.

Simulation The simulation evaluates the quality of planning and search at each step and assigns reward values. For intermediate nodes, the LLM assesses the quality of planning and searching, assigning a value between -1 and 1, where 1 indicates high quality and -1 indicates low quality:

$$R_p(s_t), R_s(s_t) = \text{LLM}(\tau_{t-1}, p_t, q_t) \tag{6}$$

For terminal nodes, if the final answer is correct, both planning and search rewards are set to 1; otherwise, they are set to -1.

Backpropagation At the end of the i-th simulation, each edge along the path from the leaf node s_t to the root undergoes a backward pass update. The updates to their values and visiting counts are executed as follows:

$$\begin{aligned} N(s_t) &\leftarrow N(s_t) + 1 \\ V_p(s_t) &\leftarrow V_p(s_t) + \frac{1}{N(s_t)}(R_p(s_t) - V_p(s_t)) \\ V_s(s_t) &\leftarrow V_s(s_t) + \frac{1}{N(s_t)}(R_s(s_t) - V_s(s_t)) \end{aligned} \tag{7}$$

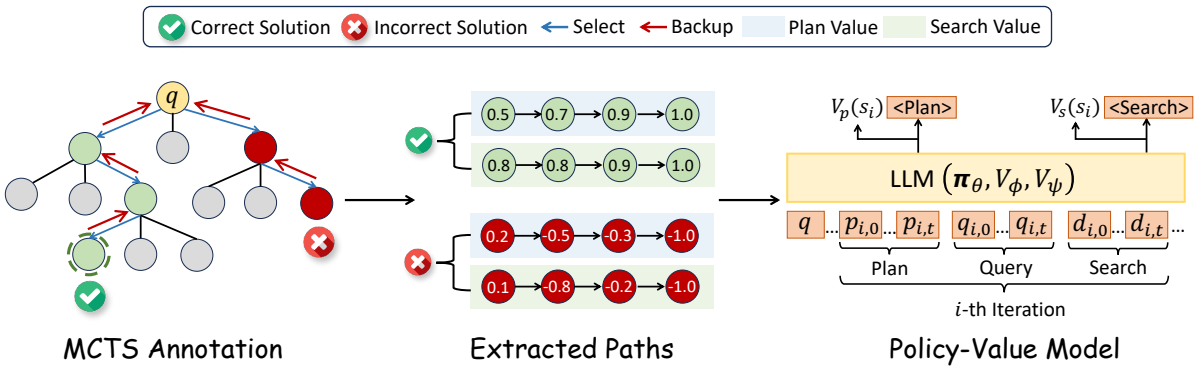


Figure 3: Overview of the proposed method: The MCTS algorithm constructs reasoning trees for training queries, from which correct and incorrect paths are extracted to train the policy and value models. During inference, Hierarchical Beam Search iteratively refines planning and search candidates, ensuring high-quality reasoning and retrieval outcomes.

This backpropagation process is crucial because it corrects potential inaccuracies in the LLM's generated scores using the answer correctness signal.

3.3 Model Training

In our framework, the policy model π_θ is initialized with a pre-trained LLM. We extend this model to derive the planning value model V_ϕ and search value model V_ψ by adding two auxiliary linear layers with a Tanh activation function. These layers operate alongside the traditional softmax layer responsible for token prediction, as illustrated in the rightmost panel of Figure 3. This design ensures that the policy model and the value models share the majority of their parameters, promoting parameter efficiency and joint optimization.

To construct the training signals for the policy model and the value models, we sample solution paths from the tree constructed through multiple rounds of MCTS. These paths are denoted as $\mathbf{x}^+$ (correct solutions) and $\mathbf{x}^-$ (incorrect solutions). We then apply a multi-task loss function to jointly update all the models:

$$\mathcal{L} = -\log \pi_\theta(\mathbf{x}^+|\mathbf{q}) + \beta \cdot \sum_{t=1}^{T(\mathbf{x})} \Big(\|V_\phi(\mathbf{s}_t) - V_p(\mathbf{s}_t)\|^2 + \|V_\psi(\mathbf{s}_t) - V_s(\mathbf{s}_t)\|^2 \Big) \quad (8)$$

Here, the first term represents the negative log-likelihood loss for next-token prediction in correct solutions, guiding the policy model to generate accurate predictions. The second term captures the loss in value prediction for both correct and incorrect solutions, ensuring the value models provide reliable estimates of expected rewards at each node. $T(\mathbf{x})$ denotes the number of steps in the solution path $\mathbf{x}$, and β is a tunable hyperparameter that controls the weight of the value loss term.

3.4 Model Inference

After obtaining the trained policy model, it can be directly used to conduct reasoning. However, this greedy decoding process fails to fully explore the planning and search spaces, limiting its ability to identify optimal reasoning paths. To address this issue, we propose a hierarchical beam search algorithm to encourage the policy model to thoroughly explore both the planning and search spaces.

Hierarchical Beam Inference At each step, the policy model first samples multiple possible plans, which are ranked and filtered by the planning value model. Based on the most promising plan, the policy model generates multiple search queries, which are used to retrieve relevant documents. The retrieved documents are then evaluated by the search value model to select the most valuable result. This iterative process continues until the maximum depth is reached or no further paths can be expanded. Finally, the answers are evaluated by the planning value model, and the answer with the highest value is selected as the output. This approach ensures a more comprehensive exploration of both the planning and search spaces, leading to higher quality and more reliable results.

4 Experiments

4.1 Datasets and Metrics

We conduct experiments on five datasets spanning both single-hop and multi-hop question-answering (QA) tasks. Specifically, the multi-hop QA tasks include the 2WikiMultiHopQA dataset (Ho et al.,

Method	HotpotQA		2WikiMulti		MusiQue		Bamboogle		TriviaQA		AVG	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Qwen-2.5-7B-Instruct												
Direct	18.20	24.10	25.80	27.61	5.80	10.96	16.94	19.00	43.20	41.84	21.99	24.70
CoT	20.64	26.64	24.00	26.44	7.80	13.69	15.32	18.81	46.28	46.03	22.81	26.32
Standard	26.00	30.04	15.00	17.31	7.20	11.57	19.35	21.77	58.80	56.42	25.27	27.42
Iterative	13.20	16.40	7.80	9.59	3.80	6.39	20.97	24.60	40.00	37.66	17.15	18.93
Gen-Retrieve	24.40	28.15	18.80	19.68	8.20	12.95	19.35	21.41	55.20	52.93	25.19	27.02
Judge-Retrieve	25.80	30.88	17.40	19.96	7.60	12.18	19.35	21.77	55.20	52.92	25.07	27.54
RAgent	26.40	30.95	26.00	28.70	9.00	14.57	30.65	35.01	57.20	50.89	29.85	32.02
Search-o1	29.80	32.37	29.60	31.33	12.40	16.67	31.45	35.95	53.60	50.70	31.37	33.40
HRM (Greedy)	34.94	32.86	34.00	34.79	12.40	17.01	36.59	39.03	61.40	53.02	35.87	35.34
HRM (HBS)	**38.62**	**36.60**	**35.87**	**35.03**	**17.20**	**17.73**	**42.28**	**46.52**	**65.66**	**58.08**	**39.93**	**38.79**
Qwen-2.5-14B-Instruct												
Direct	22.00	27.37	25.60	27.24	6.20	12.63	12.90	15.85	54.00	51.47	24.14	26.91
CoT	26.00	30.98	25.60	27.52	9.40	15.30	33.06	37.58	60.40	58.82	30.89	34.04
Standard	27.40	28.51	34.00	20.75	9.40	12.91	16.94	19.95	60.40	54.70	29.63	27.36
Iterative	15.00	16.24	5.40	6.74	5.21	8.70	10.48	15.16	42.20	36.30	15.66	16.63
Gen-Retrieve	26.80	27.10	33.20	21.23	8.40	11.70	19.35	21.41	61.60	57.30	29.87	27.75
Jud-Retrieve	27.40	28.38	33.40	20.16	9.00	11.49	18.55	21.11	60.40	55.32	29.75	27.29
RAgent	37.40	38.18	33.87	34.19	16.20	18.72	37.90	43.28	65.80	59.92	38.23	38.86
Search-o1	36.80	37.18	34.20	35.86	16.40	20.54	35.48	43.41	64.40	60.46	37.46	39.49
HRM (Greedy)	38.96	37.31	37.80	36.15	16.06	19.39	45.53	48.18	63.45	57.13	40.36	39.63
HRM (HBS)	**43.35**	**39.56**	**41.84**	**38.37**	**18.38**	**21.82**	**47.15**	**49.77**	**72.44**	**62.98**	**44.63**	**42.50**

Table 1: Evaluation results on five representative QA tasks. The **bold** fonts denote the best results in each dataset.

2020), the HotpotQA dataset (Yang et al., 2018), the Bamboogle dataset (Press et al., 2022) and the MuSiQue dataset (Trivedi et al., 2022), while the single-hop QA task is represented by the TriviaQA dataset (Joshi et al., 2017).

To evaluate performance, we employ two key metrics: Exact Match (EM) and F1 Score. Under the EM metric, a predicted answer is deemed correct if its normalized form exactly matches any of the normalized versions of the reference answers in the provided answer list. The F1 score, on the other hand, quantifies the word-level overlap between the normalized predicted answer and the reference answers, providing a measure of the answer's precision and recall.

4.2 Baselines

We compare HRM with the following three categories of methods:

Vanilla Prompting Methods This category includes direct prompting, Chain-of-Thought (CoT), and standard Retrieval-Augmented Generation (RAG). Direct prompting instructs the model to generate answers directly without retrieving external resources. Chain-of-Thought guides the model to reason step by step before arriving at the final answer. Standard RAG first retrieves relevant documents from an external corpus and then generates the answer based on the retrieved information.

Advanced RAG Methods This category includes Iterative RAG (Xu et al., 2024), Judge-then-retrieve (Asai et al., 2023), and Generate-then-retrieve (Wang et al., 2023). We implement all these baselines in our experiments. Iterative RAG decomposes the query into sub-queries, retrieves and generates answers for each, and then combines them to produce the final answer. Judge-then-retrieve first determines whether retrieval is necessary and then generates the final answer using either internal knowledge or retrieved documents. Generate-then-retrieve directly generates an answer, concatenates the answer with the question, and then retrieves and generates a refined answer.

Agentic RAG Methods This category includes RAgent (Li et al., 2025) and Search-o1 (Li et al., 2025). These methods operate by iteratively searching for the necessary information to answer the question. At each step, the policy model au-

tonomously decides when and what to retrieve. Search-o1 enhances this approach by incorporating a Reason-in-Document module, which condenses retrieved documents into reasoning steps while preserving the logical flow of the reasoning chain.

4.3 Implementation Details

To demonstrate the generality of our method, we initialize the policy with two large language models (LLMs) of different parameter sizes: Qwen2.5-7B-Instruct[1](Team, 2024) and Qwen2.5-14B-Instruct[2](Team, 2024). During Monte Carlo Tree Search (MCTS) annotation, we employ Qwen-Turbo to predict the next action and evaluate the scores for planning and searching. The policy model and value models are fine-tuned over 10 epochs with a batch size of 4 and a learning rate of 1e-6, utilizing 8 NVIDIA A100 80GB GPUs. For retrieval, we use the Wikipedia dump from January 27, 2020, as our corpus and employ DPR (Karpukhin et al., 2020) as our dense retriever. For each query, we retrieve the top-5 most relevant documents from the retrieval corpus. Additional training details can be found in Appendix B. To promote reproducibility, we plan to open-source the code upon acceptance of this work.

4.4 Main Results

In this section, we present the results of experiments conducted on five QA datasets using two model backbones, respectively. Based on the results in Table 1, several observations can be made:

First, our method achieves superior performance on all datasets across different readers, verifying the effectiveness of our approach. Notably, when using Qwen2.5-7B-Instruct-1M as the policy model, HRM achieves a 30% relative average improvement over the best-performing baseline. This improvement is attributed to the application of plan beam search and search beam search, which enables the policy model to thoroughly explore the plan and search spaces, significantly increasing the likelihood of identifying the correct reasoning path.

Second, among the baselines, agentic-RAG methods outperform both prompting methods and advanced RAG methods. This is primarily due to the flexibility agentic-RAG provides, allowing the policy model to dynamically decide what to retrieve and when to retrieve. This capability is particularly crucial for handling complex queries that require multi-step reasoning, as evidenced by the substantial performance gains on multi-hop datasets such as Bamboogle.

Third, when comparing policy models of different sizes, larger models (e.g., Qwen2.5-14B-Instruct) generally yield better performance, as expected, due to their higher model capacity. However, after applying Search Beam Search (SBS), the performance of HRM with the 7B policy model becomes comparable to that of the 14B model, highlighting the potential for smaller models to achieve competitive performance through inference-time scaling techniques.

Methods	HotpotQA		TriviaQA	
	EM	F1	EM	F1
HRM	38.62	36.60	65.66	58.08
-w/o Planning	35.35	32.84	62.10	54.78
-w/o Searching	36.75	35.05	62.58	55.63
-w/o Both	34.94	32.86	61.40	53.02

Table 2: Ablation Study. We experiment by gradually removing all model components.

5 Analysis

5.1 Ablation Study

In this section, we analyze the effectiveness of plan expansion and search expansion by removing these components and observing the resulting performance changes, as shown in Table 2.

The results demonstrate that removing either plan expansion or search expansion leads to a decline in performance, underscoring the importance of thoroughly exploring both the plan space and the search space. Notably, the removal of plan expansion results in a more significant performance drop. This is because the plan typically defines the search space; if the plan is suboptimal, it becomes challenging to retrieve high-quality results. Therefore, plan expansion plays a critical role in ensuring robust performance.

5.2 Scaling with Planning and Searching

During inference, we employ hierarchical beam search, which involves two key hyperparameters: the plan expansion size B_1 and the search expansion size B_2. To investigate their impact on model performance, we conduct experiments on the HotpotQA, 2WikiMultihopQA, and MusiQue datasets, varying these parameters within the range of 1 to 5.

[1] https://huggingface.co/Qwen/Qwen2.5-7B-Instruct
[2] https://huggingface.co/Qwen/Qwen2.5-14B-Instruct

Size	HotpotQA		2WikiMulti		MusiQue	
	EM	F1	EM	F1	EM	F1
Plan Expansion Size						
1	35.35	32.84	34.74	34.68	14.20	16.66
2	37.83	37.48	36.40	35.68	15.43	18.53
3	38.62	36.60	35.87	35.03	17.20	17.73
4	37.78	36.00	35.60	36.14	15.23	19.20
5	35.29	34.62	33.87	34.19	14.63	18.42
Search Expansion Size						
1	36.75	35.05	33.40	34.04	14.80	17.78
2	38.87	37.68	34.07	33.85	15.03	18.19
3	38.18	36.85	33.87	34.38	16.23	18.92
4	37.78	35.36	35.34	34.79	15.43	17.78
5	40.20	37.46	35.34	35.99	17.60	19.59

Table 3: We vary the expansion sizes within the range of 1 to 5 and observe the performance changes.

Based on the results presented in Table 3, several observations can be made.

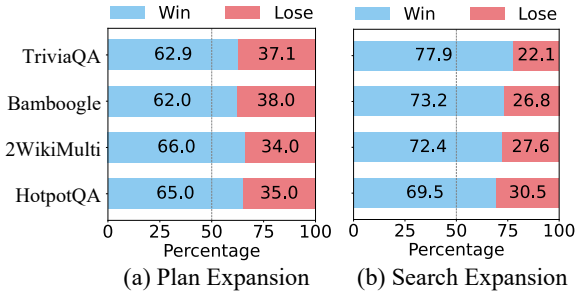
First, for the plan expansion size, model performance peaks when the expansion size is set to approximately 3. Values smaller or larger than this threshold result in a decline in performance. This is primarily because a larger plan expansion size provides the model with more opportunities to identify the optimal plan. However, when the expansion size becomes too large, the plan value model struggles to effectively rank and select the best plan due to increased complexity.

Second, **for the search expansion size, we observe that larger expansion sizes generally lead to improved performance**. This is because a larger search expansion size increases the likelihood of retrieving optimal evidence that can lead to the correct answer. Compared to the plan value model, the search value model faces relatively less difficulty in ranking search results, as it can directly evaluate the retrieved evidence, whereas the plan value model must rely on complex patterns learned from training data to make decisions.

5.3 Effectiveness of Value Models

In this section, we analyze the accuracy of the plan value model and the search value model. Specifically, during the beam ranking stage, instead of using our value model to rank, we randomly select one plan or search result and compare the performance of random selection with ranking by the value model. Based on the results shown in Figure 4, several observations can be made:

First, for both plan expansion and search expansion, ranking by the learned value model achieves better performance compared to random selection. This verifies that both value models can accurately measure the quality of plans and search results. Second, the performance superiority is more pronounced for search expansion. **This is because determining the value of search results is relatively easier than evaluating plans**. Typically, the value of a search result can be directly assessed by checking whether it contains the answer to the search query. In contrast, evaluating the quality of a plan involves greater uncertainty, as there are no obvious patterns to determine its effectiveness. Therefore, when resources are constrained, allocating more resources to search expansion may be a more robust strategy.



(a) Plan Expansion (b) Search Expansion

Figure 4: We analyze the effectiveness of value models by replacing the value ranking with random sampling.

5.4 Case Study

In this section, we present a case study from the MusiQue dataset in Figure 5.

Given the query "Who is the father-in-law of Gulcicek Hatun?", the policy model generates plans, such as searching for Gulcicek Hatun's spouse or lineage. The plan value model assigns a higher reward to the spouse search plan, pruning the lower-value alternative. The policy model then creates search queries like "Who is Gulcicek Hatun's husband?" and "Gulcicek Hatun Spouse." The first query retrieves direct information about her husband, Murad I, and receives a high positive reward, while the less relevant result is pruned. Next, the policy model searches for Murad I's father, generating queries like "Who was Murad I's father?" The result identifying Orhan Ghazi as his father receives a high reward, while irrelevant results are pruned. The final answer, Orhan Ghazi, is output. This case study illustrates how plan and search expansion broaden the candidate space, while the value models identify the most valuable candidates, validating our framework's effectiveness in handling complex queries.

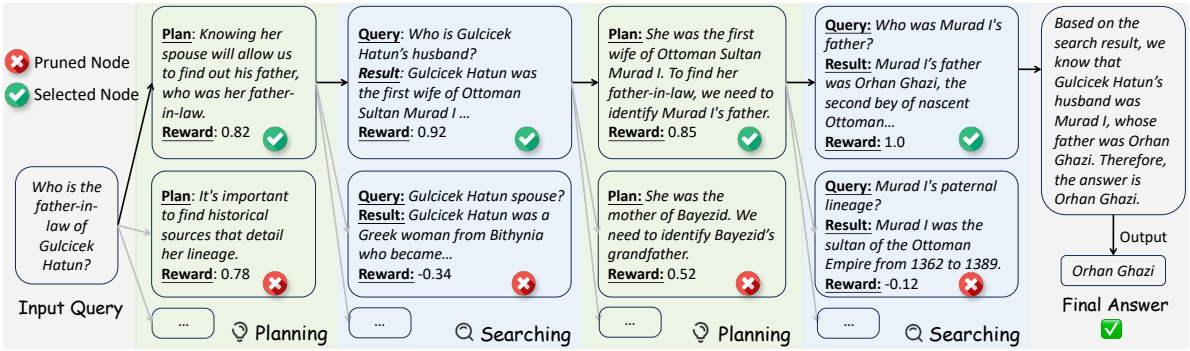


Figure 5: Case Study of HRM in Hierarchical Agentic RAG. The example illustrates how HRM optimizes reasoning by decoupling planning and search using hierarchical reward modeling. The reasoning tree dynamically selects high-reward paths while pruning suboptimal ones, demonstrating the effectiveness of Hierarchical Beam Search.

6 Related Work

6.1 Agentic Retrieval-Augmented Generation

Despite significant advancements, Large Language Models (LLMs) often generate responses that include hallucinated facts and inaccurate information (Ji et al., 2023; Shuster et al., 2021; Zhang et al., 2023), which compromises their reliability and limits their practical applicability in real-world scenarios. To mitigate this issue, researchers have turned to Retrieval-Augmented Generation (RAG), which integrates external knowledge to improve the accuracy of responses (Ram et al., 2023; Shi et al., 2023; Rashkin et al., 2021; Gao et al., 2022; Bohnet et al., 2022; Menick et al., 2022). By dynamically retrieving relevant information from external databases or documents, RAG enables LLMs to ground their outputs in verifiable evidence.

While RAG offers substantial improvements, it remains limited by its reliance on static workflows. Agentic RAG presents a more promising approach by incorporating agents into the RAG pipeline. For instance, Self-RAG (Asai et al., 2023) employs a self-reflection mechanism to iteratively predict reflection tokens during training. Auto-RAG (Yu et al., 2024) systematically plans retrievals and refines queries to acquire valuable knowledge through multi-turn iterations. MindSearch (Chen et al., 2024c) mimics human cognitive processes in web information seeking and integrates them with an LLM-based multi-agent framework. PlanxRAG (Verma et al., 2024) isolates the reasoning plan as a directed acyclic graph (DAG) outside the LM's working memory. Search-o1 (Li et al., 2025) incorporates an agentic search process into reasoning, allowing for the dynamic retrieval of information whenever LLMs face uncertain knowledge points.

6.2 Enhancing LLMs with Search

The application of search techniques to enhance LLMs has garnered considerable attention. Numerous studies have demonstrated that Monte Carlo Tree Search (MCTS) can significantly improve the reasoning capabilities of LLMs by generating diverse reasoning paths. For example, AlphaMATH (Chen et al., 2024a) utilizes MCTS to eliminate the need for process annotations from humans or GPTs. Similarly, SVPO (Chen et al., 2024b) employs MCTS to automatically annotate step-level preferences for multi-step reasoning. Llama-berry (Zhang et al., 2024b) leverages MCTS to facilitate more efficient exploration of solution spaces.

Other notable works include CoAT (Pan et al., 2025), which integrates MCTS with associative memory for structured reasoning, and MCTSr (Zhang et al., 2024a), which applies MCTS to self-refine mathematical solutions through tree-search iterations. AirRAG (Feng et al., 2025) activates intrinsic reasoning capabilities and expands the solution space for specific tasks using MCTS.

7 Conclusion

In this paper, we propose HRM, a novel framework that decouples planning and search processes using dual value models, enabling independent optimization of planning reasoning and search grounding. Our approach constructs a reasoning tree, where each node represents planning and searching steps. We leverage Monte Carlo Tree Search (MCTS) to efficiently assess the quality of each step. During inference, hierarchical beam search iteratively refines plan and search candidates through reward-guided optimization. Extensive experiments on five datasets, across policy models of varying parameter sizes, demonstrate the effectiveness of our method.

Limitations

In this paper, we propose an agentic RAG framework that fully explores the planning and searching spaces. We acknowledge two limitations of our method. First, the MCTS annotation process requires multiple simulations, which can lead to additional labeling costs. Second, our current approach focuses on retaining only the single most promising plan and search result at each step. The exploration of retaining multiple promising plans and search results is left for future work.

Ethics Statement

This work complies with the ACL Ethics Policy. All datasets and LLMs used are publicly available. Our research focuses on improving the performance of agentic RAG, and we do not anticipate any negative ethical impacts.

References

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

Bernd Bohnet, Vinh Q Tran, Pat Verga, Roee Aharoni, Daniel Andor, Livio Baldini Soares, Jacob Eisenstein, Kuzman Ganchev, Jonathan Herzig, Kai Hui, et al. 2022. Attributed question answering: Evaluation and modeling for attributed large language models. *arXiv preprint arXiv:2212.08037*.

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. 2024a. Alphamath almost zero: Process supervision without process. In *Advances in Neural Information Processing Systems*, volume 37, pages 27689–27724. Curran Associates, Inc.

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. 2024b. Step-level value preference optimization for mathematical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7889–7903, Miami, Florida, USA. Association for Computational Linguistics.

Zehui Chen, Kuikun Liu, Qiuchen Wang, Jiangning Liu, Wenwei Zhang, Kai Chen, and Feng Zhao. 2024c. Mindsearch: Mimicking human minds elicits deep ai searcher. *arXiv preprint arXiv:2407.20183*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Wenfeng Feng, Chuzhan Hao, Yuewei Zhang, Jingyi Song, and Hao Wang. 2025. Airrag: Activating intrinsic reasoning for retrieval augmented generation via tree-based search. *arXiv preprint arXiv:2501.10053*.

Luyu Gao, Zhuyun Dai, Panupong Pasupat, Anthony Chen, Arun Tejasvi Chaganty, Yicheng Fan, Vincent Y Zhao, Ni Lao, Hongrae Lee, Da-Cheng Juan, et al. 2022. Rarr: Researching and revising what language models say, using language models. *arXiv preprint arXiv:2210.08726*.

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, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 6609–6625. International Committee on Computational Linguistics.

Shima Imani, Liang Du, and Harsh Shrivastava. 2023. Mathprompter: Mathematical reasoning using large language models. *arXiv preprint arXiv:2303.05398*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL 2017*, pages 1601–1611.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. 2022. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857.

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

Jacob Menick, Maja Trebacz, Vladimir Mikulik, John Aslanides, Francis Song, Martin Chadwick, Mia Glaese, Susannah Young, Lucy Campbell-Gillingham, Geoffrey Irving, et al. 2022. Teaching language models to support answers with verified quotes. *arXiv preprint arXiv:2203.11147*.

Jianfeng Pan, Senyou Deng, and Shaomang Huang. 2025. Coat: Chain-of-associated-thoughts framework for enhancing large language models reasoning. *arXiv preprint arXiv:2502.02390*.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2022. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *arXiv preprint arXiv:2302.00083*.

Hannah Rashkin, Vitaly Nikolaev, Matthew Lamm, Lora Aroyo, Michael Collins, Dipanjan Das, Slav Petrov, Gaurav Singh Tomar, Iulia Turc, and David Reitter. 2021. Measuring attribution in natural language generation models. *arXiv preprint arXiv:2112.12870*.

Christopher D Rosin. 2011. Multi-armed bandits with episode context. *Annals of Mathematics and Artificial Intelligence*, 61(3):203–230.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2023. Replug: Retrieval-augmented black-box language models. *arXiv preprint arXiv:2301.12652*.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation. *arXiv preprint arXiv:2104.07567*.

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*.

Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. 2022. Galactica: A large language model for science. *CoRR*, abs/2211.09085.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

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

Prakhar Verma, Sukruta Prakash Midigeshi, Gaurav Sinha, Arno Solin, Nagarajan Natarajan, and Amit Sharma. 2024. Plan x rag: Planning-guided retrieval augmented generation. *arXiv preprint arXiv:2410.20753*.

Liang Wang, Nan Yang, and Furu Wei. 2023. Query2doc: Query expansion with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 9414–9423. Association for Computational Linguistics.

Shijie Xia, Xuefeng Li, Yixin Liu, Tongshuang Wu, and Pengfei Liu. 2024. Evaluating mathematical reasoning beyond accuracy. *arXiv preprint arXiv:2404.05692*.

Shicheng Xu, Liang Pang, Huawei Shen, Xueqi Cheng, and Tat-Seng Chua. 2024. Search-in-the-chain: Interactively enhancing large language models with search for knowledge-intensive tasks. In *Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, May 13-17, 2024*, pages 1362–1373. ACM.

Ryutaro Yamauchi, Sho Sonoda, Akiyoshi Sannai, and Wataru Kumagai. 2023. Lpml: llm-prompting markup language for mathematical reasoning. *arXiv preprint arXiv:2309.13078*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Tian Yu, Shaolei Zhang, and Yang Feng. 2024. Auto-rag: Autonomous retrieval-augmented generation for large language models. *arXiv preprint arXiv:2411.19443*.

Di Zhang, Xiaoshui Huang, Dongzhan Zhou, Yuqiang Li, and Wanli Ouyang. 2024a. 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. 2024b. Llama-berry: Pairwise optimization for o1-like olympiad-level mathematical reasoning. *arXiv preprint arXiv:2410.02884*.

Jiaxin Zhang, Zhuohang Li, Kamalika Das, Bradley Malin, and Sricharan Kumar. 2023. Sac3: Reliable hallucination detection in black-box language models via semantic-aware cross-check consistency: Reliable hallucination detection in black-box language models via semantic-aware cross-check consistency. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15445–15458.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

A Dataset Statistics

The dataset statistics used in this paper are shown in Table 4.

Settings	TrivaQA (Joshi et al., 2017)	Bamboogle (Press et al., 2022)	HotpotQA (Yang et al., 2018)	2WikiMultiHopQA (Ho et al., 2020)	MuSiQue (Trivedi et al., 2022)
			Dataset statistics		
Task	Single-Hop QA	Multi-Hop QA	Multi-Hop QA	Multi-Hop QA	Multi-Hop QA
Train Data	3,000	0	5,000	3,000	5,000
Test Data	500	125	500	500	500
			Evaluation settings		
Metrics	EM, F1	EM, F1	EM, F1	EM, F1	EM, F1
			Retrieval settings		
Corpus	Wikipedia	Wikipedia	Wikipedia	Wikipedia	Wikipedia
Retriever	DPR	DPR	DPR	DPR	DPR

Table 4: Statistics and experimental settings of different tasks/datasets.

B Training Details

Dataset Construction We sample 3000, 5000, 3000, 5000 queries from the training data of TriviaQA, HotpotQA, 2WikiMultihopQA and MuSiQue datasets and conduct the MCTS annotations. During MCTS annotation, following Chen et al. (2024a), the parameter w, β is set to 1.4 and 0.1, respectively. The maximum number of iterations is configured to 20. we employ Qwen-Turbo to predict the next action and evaluate the scores for planning and searching. We then sample 10,000 correct paths and 10,000 incorrect paths and the correct paths are used to train the policy model, while both paths are used to train the value models.

Training Process The policy model and value models are fine-tuned over 10 epochs with a batch size of 4 and a learning rate of 1e-6, utilizing 8 NVIDIA A100 80GB GPUs.

Inference Process During inference, both the plan expansion size and the search expansion size are set to 3.

C Prompts

The prompts used in MCTS annotations are list below:

LLM Sample Prompt

You are a highly capable web agent. Your task is to engage in multi-step reasoning and propose plans to reach a final answer for the given question.

For each step, please include the following elements:

Thought: Offer a comprehensive and detailed analysis. This section should cover:
- An analysis of the specific information required to address the question effectively and the information currently available.
- If the information is enough to answer the question, you should conduct deep analysis based on the information and then answer the question.
- If the information is not enough to answer the question, you should analyze whether the current plan progresses well.
- If yes, predict the next action.
- If no, reflect on why the progress is not good and then propose a new plan.

Action: Provide the next action. This section should cover:
- If the information is enough to answer the question, you should output the final answer in format of Finish(put the answer here) without extra content.
- If the information is not enough to answer the question, you should clearly specify the exact query for the next search in the format Search([List of Queries]) without extra content. Ensure the queries convey the same semantic information but are expressed differently to enhance the likelihood of finding the necessary information.

For the question: query, here is the reasoning process so far:
history

The Output Format:
- **Thought:** [Detailed analysis of the needed information, existing information, identifies whether information is enough. If enough, conduct analysis to obtain the final answer, else, identify what still needs to be searched]
- **Action:** [Finish(put the answer here) or Search([List of Queries])]

Please provide the plan for the next step:

LLM Evaluation Prompt

Task: Assess the effectiveness of the thought and the search result in the last reasoning step.
As an advanced web search agent, your role is to systematically evaluate the current step step.
For the question: query, here is the reasoning process so far:
history

As an expert in web search, your tasks are as follows:
1. Analyze the thought in the last step: Evaluate the thought and determine its effectiveness in reaching the final answer. Assign a score between -1 and 1, where -1 means the thought is useless and 1 means the thought is very effective.
2. Analyze the search result in the last step: Evaluate the search result and determine its effectiveness in reaching the final answer. Assign a score between -1 and 1, where -1 means the search result was ineffective, and 1 means the search results were highly useful.

You should output the following elements
Analysis of the thought:
- Analyze whether the thought from the last step were helpful in progressing toward the final answer.
- Assign a score between -1 and 1, where -1 means the step was ineffective, and 1 indicates high usefulness.
- You must conclude the analysis with the format of "the value of the thought is ***x***", where x represent the value and * is the identifier. Remember that you must output the value x with identifier ***.

Analysis of the search result:
- Analyze whether the search query and search results from the last step were helpful in progressing toward the final answer.
- Assign a score between -1 and 1, where -1 means the step was ineffective, and 1 indicates high usefulness.
- You must conclude the analysis with the format of "the value of the search result is ***x***", where x represent the value and * is the identifier. Remember that you must output the value x with identifier ***.

Please begin by analyzing the previous step: **Analysis of the thought:**