ReARTeR: Retrieval-Augmented Reasoning with Trustworthy Process Rewarding

Zhongxiang Sun
Qipeng Wang
Gaoling School of Artificial Intelligence
Renmin University of China
Beijing, China
{sunzhongxiang,wqp}@ruc.edu.cn

Weijie Yu
School of Information Technology and Management
University of International Business and Economics
Beijing, China
yu@uibe.edu.cn

Xiaoxue Zang
Kai Zheng
Kuaishou Technology Co., Ltd.
Beijing, China
xxic666@126.com
zhengkai@kuaishou.com

Jun Xu*
Xiao Zhang
Gaoling School of Artificial Intelligence
Renmin University of China
Beijing, China
{junxu,zhangx89}@ruc.edu.cn

Yang Song
Han Li
Kuaishou Technology Co., Ltd.
Beijing, China
lihan08@kuaishou.com
ys@sonyis.me

Abstract

Retrieval-Augmented Generation (RAG) systems for Large Language Models (LLMs) have shown promise in knowledge-intensive tasks, yet their reasoning capabilities, particularly for complex multi-step reasoning, remain limited. Although recent approaches have explored integrating RAG with chain-of-thought reasoning or incorporating test-time search with process reward model (PRM), these methods face several untrustworthy challenges, including lack of explanations, bias in PRM training data, early-step bias in PRM scores, and ignoring post-training that fails to fully optimize reasoning potential. To address these issues, we propose **Re**trieval-**A**ugmented **R**easoning through **T**rustworthy Proc**e**ss **R**ewarding (**ReARTeR**), a framework that enhances RAG systems' reasoning capabilities through both post-training and test-time scaling. At test time, ReARTeR introduces Trustworthy Process Rewarding via a Process Reward Model for accurate scalar scoring and a Process Explanation Model (PEM) for generating natural language explanations, enabling step refinement. During post-training, we leverage Monte Carlo Tree Search guided by Trustworthy Process Rewarding to collect high-quality step-level preference data, which is used to optimize the model through Iterative Preference Optimization. ReARTeR tackles three key challenges: (1) misalignment between PRM and PEM, addressed through off-policy preference learning; (2) bias in PRM training data, mitigated by a balanced annotation method and incorporating stronger annotations for difficult examples; and (3) early-step bias in PRM, resolved via a temporal-difference-based look-ahead search strategy. Experimental results on multi-step reasoning benchmarks demonstrate that ReARTeR significantly improves reasoning performance, highlighting its potential to advance the reasoning capability of RAG systems.

CCS Concepts

• **Information systems → Question answering**.

Keywords

Retrieval Augment Generation, Reasoning, Trustworthy

ACM Reference Format:
Zhongxiang Sun, Qipeng Wang, Weijie Yu, Xiaoxue Zang, Kai Zheng, Jun Xu, Xiao Zhang, Yang Song, and Han Li. 2025. ReARTeR: Retrieval-Augmented Reasoning with Trustworthy Process Rewarding. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '25), July 13–18, 2025, Padua, Italy*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3726302.3730102

1 Introduction

Retrieval-augmented generation (RAG) for Large Language Models (LLMs) is widely utilized to address knowledge-intensive tasks, typically comprising a generator (LLM) and a retriever (for external knowledge retrieval) [6, 11, 16, 50]. Though studies have explored integrating RAG with chain-of-thought (CoT) reasoning [42, 52], complex multi-step reasoning tasks remain challenging even for the most advanced RAG systems.

Recently, Process Reward Models (PRMs) have been introduced to enhance the reasoning capability of RAG systems through test-time scaling [3, 18], where PRM assigns a score to each step in the reasoning process, providing more fine-grained feedback [20]. However, these methods often face untrustworthy challenges: **C1: Lack of Explanations:** Existing PRMs often generate unexplainable scalar scores and cannot incorporate natural language critiques,

*Corresponding author.
Work partially done at Engineering Research Center of Next-Generation Intelligent Search and Recommendation, Ministry of Education.

SIGIR '25, July 13–18, 2025, Padua, Italy

ACM ISBN 979-8-4007-1592-1/25/07
https://doi.org/10.1145/3726302.3730102

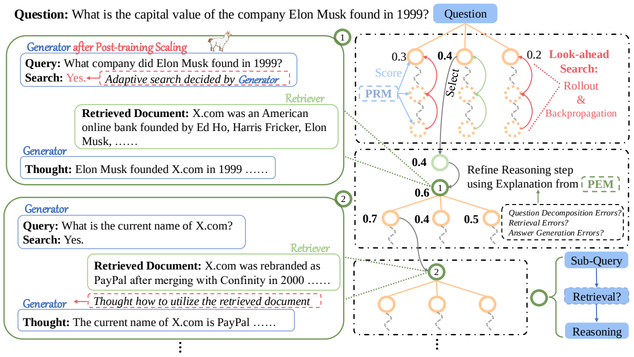

Question: What is the capital value of the company Elon Musk found in 1999?
Question
Generator after Post-training Scaling
Query: What company did Elon Musk found in 1999?
Search: Yes.
Adaptive search decided by Generator
Retriever
Retrieved Document: X.com was an American online bank founded by Ed Ho, Harris Fricker, Elon Musk, ……
Generator
Thought: Elon Musk founded X.com in 1999 ……
Generator
Query: What is the current name of X.com?
Search: Yes.
Retriever
Retrieved Document: X.com was rebranded as PayPal after merging with Confinity in 2000 ……
Generator
Thought how to utilize the retrieved document
Thought: The current name of X.com is PayPal ……
1
2
0.3
0.4
0.2
Score
PRM
Select
Look-ahead Search:
Rollout & Backpropagation
0.4
0.6
Refine Reasoning step using Explanation from PEM
0.7
0.4
0.5
Question Decomposition Errors?
Retrieval Errors?
Answer Generation Errors?
Sub-Query
Retrieval?
Reasoning

Figure 1: An example of how RARTPR tackles complex multi-step questions. The right part highlights test-time search with PRM and refinement via PEM explanations, while the left part details the reasoning step, including sub-query, adaptive retrieval, and reasoning thought.

which limits interpretability and hinders their effectiveness in enhancing refinement during test-time reasoning [22, 45]; **C2: Bias in PRM training data:** Traditional Monte Carlo methods for collecting Process Supervision Datasets often result in a distributional bias, where some questions receive disproportionately high scores [18, 45, 47]. Consequently, the PRM struggles to identify erroneous steps and fails to provide meaningful feedback on difficult examples; **C3: Early-Step Bias in PRM:** PRMs exhibit reduced accuracy in predicting rewards for earlier reasoning steps compared to those closer to the reasoning endpoint, due to the increased randomness and uncertainty in earlier steps; **C4: Lack of Reasoning Optimization:** Additionally, these approaches rely on off-the-shelf LLMs as generators without incorporating reasoning-specific optimization during the post-training phase [23, 49, 55].

To address the above challenges and improve the reasoning capabilities of RAG systems, we explore enhancing **R**etrieval-**A**ugmented **R**easoning through **T**rustworthy **P**rocess **R**ewarding (**ReARTeR**) in both test-time and post-training scaling. Specifically, as shown in Figure 1, the **testing phase** is guided by the Trustworthy Process Rewarding which is implemented through two models: (1) a Process Reward Model (**PRM**), which provides scalar scores for reasoning path selection; and (2) a Process Explanation Model (**PEM**), which generates natural language explanations for the process reward model's scores, facilitating refinement of steps with lower scores **(C1)**. During the **post-training phase**, we introduce step-level offline reinforcement fine-tuning to enhance the reasoning capabilities of the RAG system **(C4)**. Specifically, recognizing the dynamic interaction between the generator and retriever in RAG, on each iteration we employ Monte Carlo Tree Search (MCTS) [2] guided by Trustworthy Process Rewarding to generate high-quality, step-level preference data. This data is subsequently utilized to optimize the model, resulting in a substantial improvement in the system's reasoning performance.

As the core component of ReARTeR, the Trustworthy Process Rewarding solving the following challenges: **(1) Misalignment between the PEM and PRM:** Off-the-shelf LLMs used as PEM often generate explanations that are not aligned with the PRM's scores, hindering the RAG system's ability to refine outputs based on external feedback. To address this issue, we propose aligning the PEM with the PRM through Off-policy Preference Learning, which leverages preference labels derived from PRM scores before and after the RAG system refines the reasoning step based on PEM explanations. If the explanation improves the PRM score, it is treated as a positive example; otherwise, it is treated as a negative example; **(2) Bias in PRM training data:** To mitigate this **(C2)**, we leverage OmegaPRM [22], which emphasizes identifying errors in reasoning steps and balances positive and negative examples. For challenging samples, we incorporate annotations from stronger generators or human experts to provide accurate reasoning steps, thereby enhancing the PRM's ability to discern correct reasoning paths in difficult scenarios; **(3) Early-Step Bias in PRM:** To resolve this **(C3)**, we propose a temporal-difference (TD) based look-ahead search strategy (as shown in Figure 1), where simulated future reasoning steps are used to compute expected rewards, enabling updates to the current step's reward estimation. Compared to previous approaches [35], this method effectively achieves a balance between bias and variance.

We summarize the major contributions of this paper as follows:

(1) We pioneer the exploration of combining post-training and test-time scaling to enhance the multi-step reasoning capabilities of RAG systems. By integrating Trustworthy Process Rewarding, ReARTeR improves the quality of reasoning paths explored during the post-training phase, as well as the accuracy of PRM and the refinement ability of RAG systems during the test phase.

(2) We tackle key challenges in implementing Trustworthy Process Rewarding by aligning the PEM and PRM through off-policy preference learning, balancing the training data of PRM, and employing a TD-based look-ahead search strategy to reduce Early-Step Bias of PRM.

(3) Experimental results demonstrate that ReARTeR achieves significant improvements on multiple public multi-step reasoning RAG datasets, validating the feasibility of enhancing RAG systems' reasoning capabilities through post-training and inference-time scaling with ReARTeR.

2 Related Work

2.1 Learning and Search for Reasoning

Advanced reasoning models often follow the *learning and search* principle [38] to enhance reasoning capabilities through post-training and test-time scaling strategies.

Post-training Scaling. ReFT [23] employs reinforcement fine-tuning, where LLMs explore reasoning paths and optimize based on feedback, using PPO for training. While PPO achieves better results than DPO due to interactive updates, it suffers from instability. Iterative training methods [34] offer more stability and efficiency, with Iterative Preference Optimization [27] improving reasoning by constructing preference CoT data and using iterative DPO. However, these approaches face challenges in collecting step-level reasoning preferences and rely on difficult-to-collect pairwise data. To address these limitations, we propose using MCTS to collect step-level preference data and employ KTO [5] for stable optimization, leveraging process supervision to enhance reasoning.

Test-time Scaling. Test-time scaling typically relies on (1) **Self-Refinement**, where models iteratively improve outputs [24], but

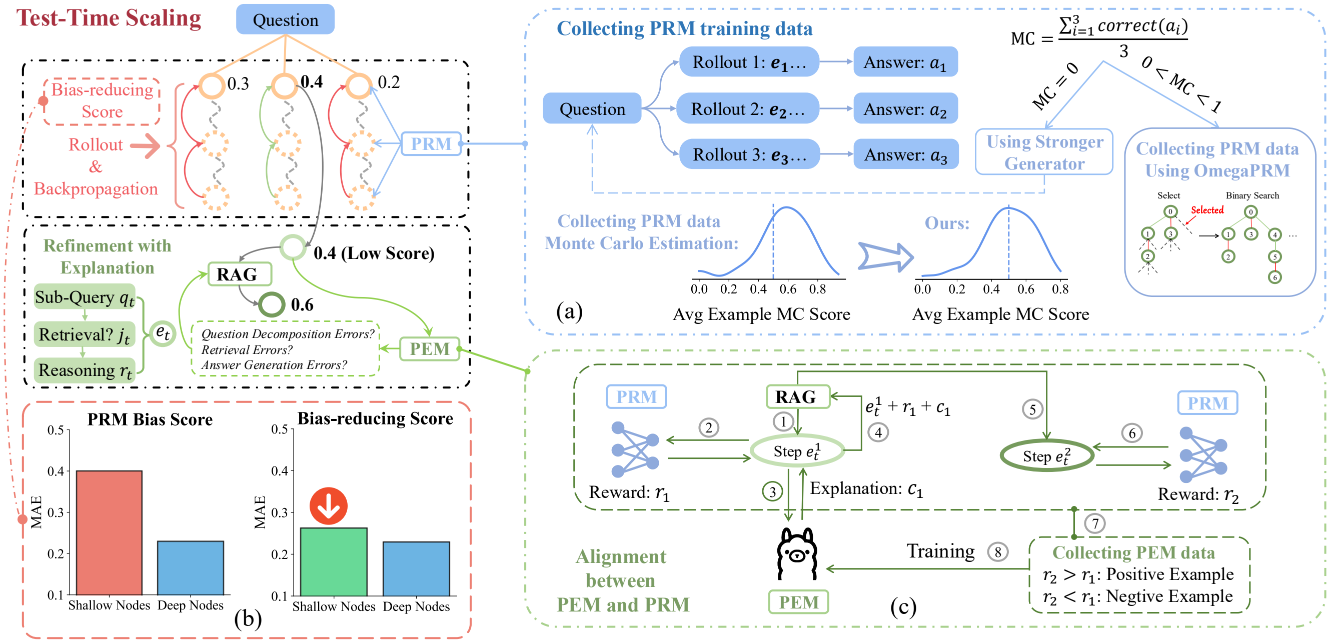


Figure 2: Test-Time Scaling of ReARTeR, which includes collecting unbiased PRM training data for PRM (a), reducing early-step bias in PRM (b), and alignment between PEM and PRM (c).

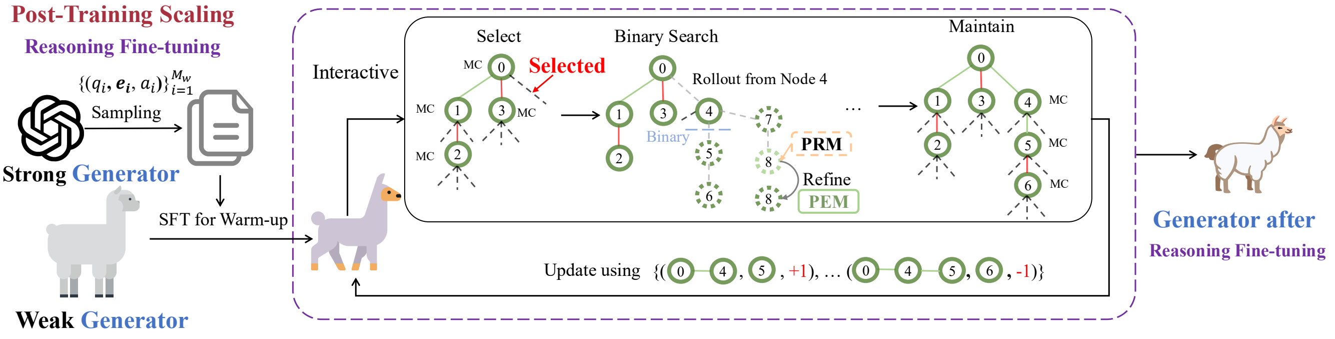


Figure 3: Post-Training Scaling of ReARTeR, which includes Warm-Up and Step-Level Offline Reinforcement Learning Stages.

this approach is limited by the lack of external feedback [10]; and (2) **Search with Verifier**, which generates multiple outputs and selects the best using a verifier, such as a Process Reward Model (PRM). While PRM scores have been used as feedback for Self-Refinement [48], they often fail to guide effective improvements in RAG scenarios. To overcome this, we combine PRM-aligned PEM explanations with step-level Self-Refinement for better reasoning performance. PRMs are critical during search, but their training data collection significantly affects performance. Existing methods [3, 18] use Monte Carlo methods to generate process supervision signals, discarding reasoning steps after rollouts, resulting in inefficiency. OmegaPRM [22] improves this by storing rollouts for reuse and using binary search to identify errors, balancing positive and negative examples. Building on OmegaPRM, we incorporate stronger generators for difficult problems and propose a TD-based look-ahead search strategy to enhance PRM accuracy for shallow reasoning nodes, achieving trustworthy process rewards.

2.2 Retrieval-Augmented Reasoning

Retrieval-augmented generation for Large Language Models is widely used for knowledge-intensive tasks [6, 11, 16, 32, 36, 37, 50], but remains limited in handling complex multi-step reasoning. Facing this challenge, existing works integrate RAG with CoT reasoning [42, 52]. For instance, Self-Ask [28] uses CoT to explicitly reason through follow-up questions before addressing the query, while IRCoT [42] interleaves retrieval with reasoning steps to iteratively refine reasoning using CoT and retrieved results. Recently, Yue et al. [53] proposed an iterative demonstration-based RAG method that performs multiple iterations to achieve test-time scaling. However, these approaches primarily leverage the long context capabilities of

LLMs and directly combine CoT with retrieval without effectively utilizing learning and search to enhance the reasoning capabilities of RAG systems. CR-Planner [18] attempts to directly use Process Reward Models to assist search and improve the reasoning capability of RAG systems through test-time scaling. However, it fails to address the untrustworthy challenges inherent in PRMs. We pioneer the use of trustworthy process rewarding to guide both post-training scaling and test-time scaling, significantly enhancing the multi-step reasoning capabilities of RAG systems.

3 Method

3.1 Overview

In this section, we present the overview of ReARTeR, which enhances Retrieval-Augmented Reasoning through Trustworthy Process Rewarding in both test-time and post-training scaling. The policy model π_θ of ReARTeR includes a generator G, which can either be an off-the-shelf LLM such as the proprietary model GPT4-o [1] or an open-source model such as LLaMA3 [4] which can be post-trained for enhancing reasoning, and a retriever E. Additionally, ReARTeR incorporates a Process Reward Model (PRM) R and a Process Explanation Model (PEM) C.

Given a complex multi-step question q and a retrieval corpus $\mathcal{D}$, ReARTeR generates a reasoning process (CoT) $\mathbf{e}$ before producing an answer a to q. The CoT of ReARTeR consists of a sequence of reasoning steps:

$$\mathbf{e} = [e_1, e_2, \dots, e_T], \tag{1}$$

where T represents the maximum length of the reasoning steps.

As illustrated in Figure 2, each reasoning step e_t comprises a subquery q_t, a retrieval indicator j_t, external knowledge d_t retrieved by E from the corpus $\mathcal{D}$ if j_t = "Yes", and a thought r_t generated by the generator based on the context:

$$e_t = [q_t, j_t, d_t, r_t]. \tag{2}$$

At timestep t, the reasoning step e_t is sampled from the policy $\pi_\theta(s_t)$, where the state s_t represents the combination of the question q and the sequence of reasoning steps up to e_{t-1}.

At each sampling process of e_t, we first sample M different reasoning steps:

$$\mathcal{E}_t = [e_t^1, e_t^2, \dots, e_t^M].$$

Subsequently, the PRM R predicts scores to the reasoning steps in $\mathcal{E}_t$:

$$r_t = R(s_t, e_t), \quad r_t \in (0, 1).$$

The reasoning step with the highest reward score is selected:

$$\hat{e}_t = \arg\max_{e_t^m \in \mathcal{E}_t} R(s_t, e_t^m),$$

if $\hat{r}_t > \tau$, then $e_t \leftarrow \hat{e}_t$ is directly added to s_t. Otherwise, a **refinement phase** is initiated, where the process critic model C provides an explanation c_t for the low process reward score of $\hat{e}_t$:

$$c_t = C(s_t, \hat{e}_t, \hat{r}_t).$$

The policy model then utilizes external feedback to correct $\hat{e}_t$:

$$e_t = \pi_\theta(s_t | \hat{e}_t, c_t, \hat{r}_t).$$

By employing this refinement phase, ReARTeR significantly improves the quality of reasoning steps sampled during beam search. Compared to the reasoning steps generated by MCTS [30], our approach strikes a better balance between accuracy and computational efficiency.

Finally, e_t is added to the reasoning process:

$$s_{t+1} = [s_t, e_t].$$

Ultimately, the policy π_θ generates the final answer a based on the question q and the complete reasoning process $\mathbf{e}$.

In the following sections, we will introduce the implementation of the PRM (**§ 3.2**), including its training process and the method for reducing early-step bias for PRM. Additionally, we describe the training process of the PEM (**§ 3.4**) and the post-training scaling strategy for ReARTeR (**§ 3.5**).

3.2 Process Reward Model Training

The Process Reward Model of ReARTeR is trained to truthfully predict the process reward score of each intermediate step e_t.

Training data collection: Considering the training data requires process supervision labels which are hard to annotate, to reduce human annotation costs, existing methods propose an automatic annotation approach using the Monte Carlo method to generate process supervision signals [3, 18]. For each step of a CoT $\mathbf{e}$, multiple complete reasoning paths and final answers are obtained via rollouts. By evaluating the accuracy of the final answers, process supervision signals for the current reasoning step can be derived.

However, as shown in Figure 2(a), we observed that this method often introduces distributional bias, where most questions receive disproportionately high scores. Additionally, for difficult questions, the sampled process supervision signals frequently result in a value of zero, leaving the PRM unable to identify erroneous steps or provide meaningful feedback on challenging examples.

To address this issue, as illustrated in Figure 2(a), we first perform N rollouts for the question q to obtain $\{(q_1, \mathbf{e}_1, a_1), \dots, (q_N, \mathbf{e}_N, a_N)\}$. The accuracy of the final answers across all rollouts is used to compute the Monte Carlo (MC) score:

$$\text{MC} = \frac{\sum_{n=1}^{N} \text{correct}(a_n)}{N}. \tag{3}$$

For questions where $0 < \text{MC} < 1$, we employ the OmegaPRM [22] annotation scheme, which efficiently identifies the first error in $\mathbf{e}$ using binary search and balances positive and negative examples, thereby ensuring both efficiency and quality. For questions where MC = 0, we switch to a stronger generator for reasoning. Questions with final MC = 1 or MC = 0 (even when using a stronger generator) are discarded, as they lack discriminative value and do not enable the model to identify correct or incorrect reasoning steps for specific questions.

For the selected questions, following the above process, we construct the process supervision data $\mathcal{D}_{\text{prm}} = \{(s_i, e_i, \text{MC}_i)\}_{i=1}^{M_r}$, where M_r represents the number of samples in $\mathcal{D}_{\text{prm}}$, and MC_i is the MC score computed for $[s_i, e_i]$ after N rollouts using Eq. 3.

PRM Training: For each process supervision data (s_i, e_i, MC_i) in $\mathcal{D}_{\text{prm}}$, we define binary labels $y_i = 1$ if $\text{MC}_i > 0.5$, otherwise $y_i = 0$. We utilize the Cross-Entropy (CE) loss to train the PRM:

$$\mathcal{L}_{\text{prm}} = -\frac{1}{M_r} \sum_{i=1}^{M_r} \left[y_i \log R(s_i, e_i) + (1 - y_i) \log(1 - R(s_i, e_i)) \right],$$

where R denotes the process reward model.

3.3 Reducing Early-Step Bias for PRM

At the inference stage of PRM, we observe that PRMs exhibit reduced accuracy in predicting rewards for earlier reasoning steps (shallow nodes) compared to those closer to the reasoning end-point (deep nodes), as shown in Figure 2(b). This phenomenon, attributed to the increased randomness and uncertainty in earlier steps, is referred to as **early-step bias**.

Some existing works adopt a Lookahead Search strategy [35], which performs a simulation by rolling out up to H steps further, stopping early if the solution end-point is reached. The PRM's score at the end of this rollout is then used to evaluate the current step during beam search. While this approach mitigates bias, it introduces significant variance [39]. To achieve a bias-variance trade-off, inspired by Temporal Difference (TD) learning [40], we propose a TD-based Lookahead Search to update the PRM scores for shallow nodes:

$$r_t \leftarrow r_t + \alpha \Delta_t,$$

where

$$\Delta_t = (r_{t+1} - r_t),$$

$r_t = R(s_t, e_t)$, and α is the discount factor.

In our approach, the termination of the Lookahead Search simulation is adaptively determined by whether Δ_t falls below a threshold β (indicating diminishing returns in further rollouts when reward scores stabilize) or if the predefined step limit H is reached. This adaptive simulation mechanism balances computational efficiency and bias reduction, saving resources while maintaining performance.

3.4 Process Explanation Model

In this section, we introduce the training procedure of the Process Explanation Model. After training the PRM, the PRM can effectively score the reasoning process; however, the PRM score is an unexplainable scalar and cannot provide natural language critiques. To address this limitation, we designed PEM, a generative model specifically aimed at producing explanations for refinement.

However, directly using off-the-shelf LLMs as PEM often results in explanations misaligned with the PRM's scalar scores, hindering the generator's ability to refine reasoning steps based on external feedback. To address this issue, as shown in Figure 2(c), we propose aligning the PEM with the PRM through **Off-policy Preference Learning**. This method uses the PRM as a verifier to provide feedback for the PEM-generated explanations, yielding preference data $\mathcal{D}_{\text{pem}}$. The PEM is then updated to align its explanations with the PRM's scoring, facilitating the generation of explanations that enhance the policy model's reasoning step through refinement.

Given the state s_t and reasoning step e_t^1, the process is as follows:

1. The PRM provides an initial score:

$$r_t^1 = R(s_t, e_t^1),$$

and the PEM generates an explanation:

$$c_t = C(s_t, e_t^1, r_t^1).$$

2. Using external feedback, the policy model π_θ (i.e., RAG) refines the reasoning step:

$$e_t^2 = \pi_\theta(s_t \mid e_t^1, c_t, r_t^1).$$

3. The PRM re-evaluates the refined step:

$$r_t^2 = R(s_t, e_t^2).$$

If $r_t^2 > r_t^1$, then (s_t, e_t^1, c_t) is labeled as a positive example with preference label $p_t = +1$. Otherwise, it is labeled as a negative example with $p_t = -1$ (cases where $r_t^2 = r_t^1$ are discarded). The collected PEM preference training dataset is denoted as:

$$\mathcal{D}_{\text{pem}} = \left\{ \left(s_t, e_t^1, c_t, r_t^1, r_t^2, p_t \right) \right\}_{i=1}^{M_e},$$

where M_e is the size of $\mathcal{D}_{\text{pem}}$.

During the training phase, since the collected preference data is binary, we employ the KTO Loss [5] to optimize the PEM. This loss is designed for binary preference optimization and is robust to noise in the data. The KTO Loss incorporates a hyperparameter $\lambda_U > 1$ for negative examples, reflecting *loss aversion*. In our dataset, the negative examples include the corresponding PRM scores r_1 and r_2, which can be used to dynamically adjust λ_U, reflecting the degree of loss aversion. Instead of assigning a uniform λ_U for all negative examples as in the original KTO Loss, we introduce a dynamic λ_U:

$$\lambda_U = \lambda_0 \cdot \exp(r_1 - r_2),$$

where λ_0 is the base value, which provides more accurate loss aversion.

3.5 Post-Training Scaling of ReARTeR

In this section, we introduce how ReARTeR enhances the reasoning capabilities of RAG systems through post-training scaling. While test-time scaling can improve the reasoning performance of RAG systems to some extent, for certain weak open-source LLM-based RAG systems π_θ^w, their inherent limitations in reasoning capabilities prevent them from solving complex multi-hop questions solely through test-time scaling. Inspired by [43, 49, 54], we propose a step-level offline reinforcement fine-tuning approach to strengthen the reasoning abilities of the RAG system. As illustrated at Figure 3 (the retriever is omitted for simplicity), this approach comprises two stages: warm-up and step-level offline reinforcement learning.

Warm-Up Stage: In the warm-up stage, we utilize a strong generator-based RAG system (π_θ^s) to generate a dataset containing reasoning steps:

$$\mathcal{D}_w = \{(q_i, \mathbf{e}_i, a_i)\}_{i=1}^{M_w},$$

where $\mathbf{e} = [e_1, e_2, \ldots, e_T]$ and $e_t = [q_t, j_t, d_t, r_t]$.

During fine-tuning of the weak policy π_θ^w using $\mathcal{D}_w$, the retriever-generated content d_t must be masked, as it is not produced by the generator:

$$\mathcal{L}_w = -\sum_{i=1}^{M_w} \sum_{t=1}^{T} \sum_{k=1}^{|e_t|} \mathbf{1}\left[o_{t,k} \notin d_t \right] \log \pi_\theta^w \left(o_{t,k} \mid q_i, o_{<t}, o_{t,<k} \right),$$

where $o_{t,k}$ denotes the k-th token in sequence e_t, $o_{<t}$ represents all tokens generated before step t (i.e., tokens in $e_1, \ldots, e_{t-1}$), and $o_{t,<k}$ denotes tokens from the first to the position $(k-1)$ within e_t.

Step-Level Offline Reinforcement Learning: In the step-level offline reinforcement learning stage, the policy model π_θ^w iteratively collects step-level preference data using MCTS and leverages this data to improve its reasoning capability via KTO Loss. The updated model is then used to collect new data for further policy updates.

Data Collection: To ensure efficient data collection and balance between positive and negative examples, we adopt the OmegaPRM-based MCTS approach [22] introduced in § 3.2. During rollouts, the generated nodes are scored using the PRM and PCM trained with the strong generator π_θ^s. Reasoning steps with low reward scores are refined to gather higher-quality data.

Iterative Updates: In the first iteration, the policy model is initialized as $\pi_{\theta,0}^w$, which is the model fine-tuned during the warm-up stage. Using the collected preference data:

$$\mathcal{D}_0^u = \{(s_i, e_i, MC_i)\}_{i=1}^{M_u},$$

where $MC_i > 0.5$ indicates desirable (positive) reasoning steps and $MC_i \leq 0.5$ indicates undesirable (negative) reasoning steps, $\mathcal{D}_0^u$ is used to update $\pi_{\theta,0}^w$ via KTO Loss [5] (with the retrieved document d_i in reasoning steps e_i masked). This yields the updated model $\pi_{\theta,1}^w$. In subsequent iterations, $\pi_{\theta,1}^w$ is used as the policy model to generate preference data $\mathcal{D}_1^u$, which is then used to update $\pi_{\theta,1}^w$ to $\pi_{\theta,2}^w$. This process is repeated for I iterations, progressively improving the reasoning capability of π_θ^w. Compared to the PPO approach used in ReFT [23], our method sacrifices some exploration but achieves more stable updates. Finally, $\pi_{\theta,I}^w$ represents the policy model after post-training scaling of ReARTeR, which can be combined with test-time scaling to further enhance the reasoning capabilities of RAG systems.

4 Experiments

In this section, we empirically verify the effectiveness of ReARTeR by addressing the following research questions:
RQ1: How does ReARTeR improve the reasoning capabilities of RAG systems in both closed-source and open-source models?
RQ2: How do the components of ReARTeR affect test-time scaling?
RQ3: How does the number of iterations during the post-training process of ReARTeR affect its performance?
RQ4: How effective is ReARTeR in aligning PEM and PRM?

The source code and detailed prompts have been shared at: https://github.com/Jeryi-Sun/ReARTeR.

4.1 Experimental Settings

4.1.1 Datasets.

In this paper, we focus on leveraging ReARTeR to address complex multi-step question-answering (QA) tasks. To this end, we utilize five benchmark datasets: HotpotQA [51], 2WikiMultiHopQA [8], Musique [41], Bamboogle [29], and StrategyQA [7]. Wikipedia passages serve as the retrieval corpus for all datasets [14]. Following the general experimental setup of RAG [12, 14, 15], we sample 500 examples from the development sets of HotpotQA, 2WikiMultiHopQA, and Musique as test sets. For Bamboogle, which has only 125 examples in its test set, we include all of them as the test set. Since StrategyQA lacks dev or test sets, we sample 500 examples from its training set for testing.

For the training data used in PRM and PCM, and for the post-training of ReARTeR, we sample 200 examples from the training sets of each dataset. Using the PRM training data construction strategy described in § 3.2, we generate a total of $M_r = 167,716$ training examples. Similarly, using the PCM training data construction strategy described in § 3.4, we generate $M_e = 769$ training examples. For the post-training phase, the warm-up stage uses $M_w = 548$ examples, and the preference data collected during each iteration averages $M_u = 27,822$ examples.

4.1.2 Evaluation Metrics.

During the evaluation phase, we observed that the outputs of reasoning-optimized RAG systems are typically longer compared to those generated by traditional RAG systems. Specifically, while the model accurately answers the question, it often includes extensive supplementary information. This renders exact-match metrics such as EM unsuitable for our evaluation tasks. Therefore, we adopt accuracy (**ACC**$_R$) as our primary evaluation metric, which determines whether the golden answer is contained within the predicted answer generated by the RAG system. To further refine our evaluation, we employ an LLM-as-Judge approach [17], using GPT4-o [1] as the evaluation model to assess whether the predicted answer is correct. This accuracy metric is referred to as **ACC**$_L$. The evaluation prompt is as follows:

> Given a Question and its Golden Answer, verify whether the Predicted Answer is correct. The prediction is correct if it fully aligns with the meaning and key information of the Golden Answer. Respond with True if the prediction is correct and False otherwise.
> Question: { }
> Golden Answer: { }
> Predicted Answer: { }

Further manual verification confirms the reliability of the **ACC**$_L$ metric.

During the Process Reward Model Training and Post-Training stages, we use ACC$_R$ to determine correctness in Eq. 3, which is more efficient and better suited for collecting large amounts of training data as reward feedback.

4.1.3 Backbone and Baseline Models.

To verify the effectiveness of ReARTeR in enhancing the reasoning capabilities of RAG systems, we selected different generators for evaluation. These include the proprietary GPT4o-mini [1] for test-time scaling and the open-source LLaMA3.1-8B [4] (Llama-3.1-8B-Instruct) for both post-training (warm-up from GPT4o-mini) and test-time scaling.

We compared ReARTeR against several baselines: 1. **Naive Generation:** Directly generating answers using the generator without retrieval. 2. **Standard RAG:** Traditional retrieval-augmented generation systems. *Given that ReARTeR employs multi-path reasoning with CoT processes, which include adaptive retrieval and final answer generation summarized from CoTs, we further compared it with:* 3. **Branching Methods (Branching):** These execute multiple reasoning paths in parallel for a single query, including SuRe [16] and REPLUG [33]. 4. **Summarization-based Methods (Summary):** LongLLMLingua [13], RECOMP-abstractive [50], and Selective-Context [19]. 5. **Adaptive Retrieval Methods (AR):** SKR [46] which adaptively retrieve based on generator's knowledge. 6. **RAG-CoT Methods (RAG-CoT):** These integrate RAG with CoT reasoning, including Self-Ask [28], Iter-RetGen [31], and IRCoT [42]. 7. **Test-time Scaling Methods (Test-Time):** CR-Planner [18], a recently proposed approach for scaling RAG using PRM at test time.

Additionally, we compared ReARTeR with LLaMA3.1-8B as the backbone against recent **Open-source Reasoning Models (Reasoning)**, such as Marco-o1-Qwen-7B [56] and Skywork-o1-Llama-3.1-8B [25], which have been extensively optimized for reasoning through large-scale training in general domains and test-time scaling, both integrated into standard RAG configurations.

4.1.4 Implementation Details. The implementation of ReARTeR and the baseline models is based on the open-source RAG framework FlashRAG [14]. The number of samples M generated per reasoning step for ReARTeR at test-time is set to 3, balancing accuracy and efficiency. The maximum number of reasoning steps T for Chain-of-Thought (CoT) reasoning is set to 5, where shallow nodes are defined as the first 3 reasoning steps and deep nodes are the remaining steps. To ensure a fair comparison, the setup of CR-Planner is consistent with that of ReARTeR. The threshold τ for initiating the refinement phase is set to 0.5. For the lookahead search, the predefined step limit H and stopping threshold β are set to 3 and 0.05, respectively. The number N in PRM training data collection is set to 5. To ensure fairness, we configure the retrieval settings as follows: for iterative retrieval baselines and ReARTeR, the number of external documents retrieved per step is set to Top 1; for single-retrieval baselines, the number of retrieved documents is set to 3. The stronger generator used for collecting PRM training data is GPT4-o. The retriever utilized in all experiments is e5-base-v2 [44]. For the PRM, following [18] we fine-tune skywork-reward-llama-3.1-8b-v0.2 [21] with LoRA [9], which is fine-tuned from the general-purpose LLM and excels at scoring in complex scenarios. For the PEM, we fine-tune the Llama-3.2-3B-Instruct [4], which is efficient and effective in generating the explanation for the policy model to refine error reasoning steps. We run all the experiments on machines equipped with NVIDIA A6000 GPUs and 52-core Intel(R) Xeon(R) Gold 6230R CPUs at 2.10GHz.

4.2 RQ1: Overall Performance

Table 1 presents the experimental results of applying ReARTeR to RAG systems with two different generators: the proprietary GPT4o-mini and the open-source LLaMA3.1-8B, across five multi-step QA datasets. For the RAG system with GPT4o-mini as the generator, where fine-tuning is not feasible, we applied only the Test-Time Scaling component of ReARTeR. Based on the results in Table 1, we observed the following key findings: (1) Compared to baseline models, ReARTeR significantly improves the reasoning capabilities of RAG systems in both closed-source and open-source setups, demonstrating the generalizability of the ReARTeR framework in enhancing RAG systems' reasoning abilities. (2) ReARTeR outperforms Branching methods, indicating that multi-path exploration through CoT reasoning is better suited for complex multi-step QA tasks than probability integration in REPLUG or Best-of-K strategies in SuRe. (3) ReARTeR surpasses summarization-based methods, suggesting that conducting CoT reasoning followed by summarization is superior to directly compressing and summarizing external document knowledge for multi-step reasoning tasks. (4) ReARTeR outperforms adaptive retrieval methods, showing that allowing the generator to dynamically decide whether to retrieve in the CoT process can further unlock the model's reasoning potential and improve its ability to answer complex questions. (5) ReARTeR exceeds the performance of RAG-CoT methods, demonstrating that our approach, which leverages Post-Training and Test-Time Scaling, more effectively enhances reasoning capabilities compared to directly combining RAG and CoT reasoning. (6) ReARTeR outperforms CR-Planner, validating that our proposed Trustworthy Process Rewarding mechanism produces superior reasoning paths for RAG systems, thereby improving their ability to handle complex multi-step reasoning problems. (7) ReARTeR surpasses models extensively optimized for reasoning through large-scale training on general domains and test-time scaling, such as Skywork-o1 and Marco-o1. This result indicates that models optimized for general tasks are less effective in RAG-specific reasoning scenarios compared to our framework, further highlighting the effectiveness of ReARTeR in enhancing the reasoning capabilities of RAG systems.

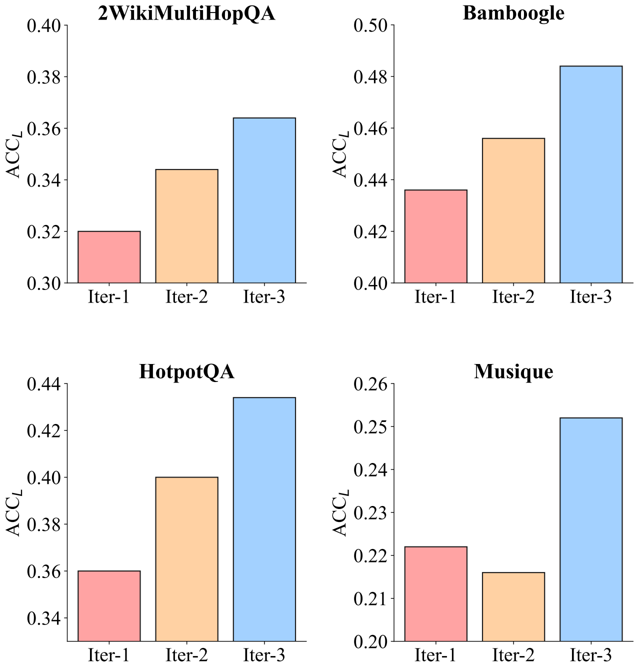


Figure 4: The impact of Post-Training Scaling iterations on ReARTeR using LLaMA-3.1-8B as the generator.

4.3 RQ2: Ablation Study of ReARTeR

In this section, we conduct an ablation study to analyze the impact of different components of ReARTeR on the test-time scaling performance of RAG systems. Specifically, we evaluate the following configurations: (1) **w/o Refinement**: Removing the refinement phase to analyze its effect on the reasoning process of ReARTeR. (2) **w/o PEM**: Replacing the Process Explanation Model (PEM) with the process reward score directly provided by the PRM during the refinement phase, to evaluate the importance of PEM-generated explanations for refinement. (3) **w/o TD-Lookahead**: Removing the TD-based lookahead search to validate its role in mitigating

Table 1: Performance comparisons between ReARTeR and the baselines. The above table shows results with GPT4-o-mini as the generator (Only Test-Time Scaling), while the below table uses LLaMA3.1-8B. The boldface indicates the best performance.

Types	Models	2WikiMultiHopQA		Bamboogle		HotpotQA		Musique		StrategyQA	
		ACC_R	ACC_L	ACC_R	ACC_L	ACC_R	ACC_L	ACC_R	ACC_L	ACC_R	ACC_L
GPT4o-mini	Naive Generation	0.348	0.346	0.240	0.280	0.324	0.404	0.134	0.170	0.724	0.724
	Standard RAG	0.344	0.292	0.272	0.328	0.342	0.450	0.172	0.188	0.674	0.674
Branching	SuRe	0.244	0.264	0.168	0.208	0.270	0.380	0.128	0.146	0.550	0.576
	REPLUG	0.296	0.254	0.224	0.256	0.350	0.428	0.132	0.138	0.654	0.654
Summary	LongLLMLingua	0.324	0.316	0.248	0.288	0.358	0.450	0.150	0.172	0.722	0.722
	RECOMP-abstractive	0.298	0.306	0.136	0.176	0.332	0.398	0.118	0.134	0.628	0.628
	Selective-Context	0.350	0.290	0.240	0.288	0.366	0.442	0.152	0.172	0.688	0.688
Adaptive	SKR	0.364	0.314	0.248	0.288	0.360	0.454	0.162	0.174	0.712	0.712
RAG-CoT	Self-Ask	0.336	0.478	0.336	0.416	0.392	0.462	0.260	0.270	0.556	0.556
	Iter-RetGen	0.326	0.270	0.232	0.256	0.374	0.456	0.178	0.188	0.686	0.686
	IRCoT	0.492	0.114	0.272	0.184	0.434	0.308	0.192	0.214	0.406	0.406
Test-Time	CR-Planner	0.520	0.478	0.488	0.524	0.404	0.416	0.272	0.262	0.744	0.744
Ours	ReARTeR	**0.554**	**0.534**	**0.496**	**0.544**	**0.468**	**0.506**	**0.296**	**0.302**	**0.772**	**0.772**
LLaMA3.1-8B	Naive Generation	0.326	0.254	0.144	0.168	0.208	0.268	0.068	0.096	0.672	0.672
	Standard RAG	0.336	0.212	0.168	0.216	0.334	0.398	0.104	0.098	0.674	0.674
Branching	SuRe	0.122	0.262	0.160	0.192	0.266	0.346	0.106	0.144	0.478	0.498
	REPLUG	0.334	0.204	0.168	0.232	0.290	0.348	0.078	0.090	0.654	0.654
Summary	LongLLMLingua	0.304	0.294	0.168	0.216	0.314	0.382	0.088	0.100	0.584	0.584
	RECOMP-abstractive	0.324	0.322	0.104	0.160	0.318	0.380	0.112	0.126	0.628	0.628
	Selective-Context	0.266	0.204	0.144	0.200	0.296	0.358	0.092	0.104	0.690	0.690
Adaptive	SKR	0.336	0.212	0.176	0.208	0.300	0.372	0.100	0.112	0.662	0.662
RAG-CoT	Self-Ask	0.306	0.322	0.360	0.432	0.316	0.408	0.222	0.226	0.616	0.616
	Iter-RetGen	0.310	0.224	0.144	0.176	0.302	0.362	0.084	0.084	0.642	0.642
	IRCoT	0.338	0.312	0.120	0.104	0.210	0.146	0.060	0.042	0.242	0.242
Test-Time	CR-Planer	0.420	0.350	0.304	0.336	0.332	0.350	0.144	0.098	0.664	0.654
Reasoning	Marco-o1	0.442	0.184	0.224	0.200	0.352	0.348	0.134	0.104	0.654	0.504
	Skywork-o1	0.344	0.190	0.176	0.160	0.306	0.256	0.092	0.060	0.612	0.326
Ours	ReARTeR	**0.470**	**0.364**	**0.438**	**0.484**	**0.424**	**0.434**	**0.244**	**0.252**	**0.724**	**0.724**

early-step bias in the PRM. (4) **w/o PRM Data**: Training the PRM using data collected with traditional Monte Carlo methods instead of the unbiased data collection strategy proposed in ReARTeR, to analyze the quality of the PRM trained with our data collection method. (5) **w/o Beam Search**: Disabling beam search by setting $M = 1$, resulting in only a single reasoning path being sampled to generate the CoT.

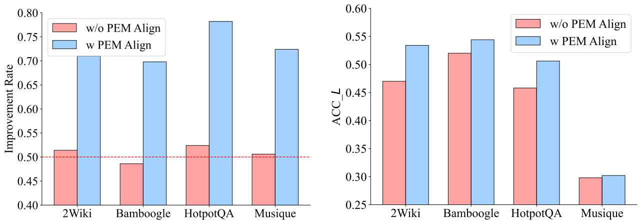
The experimental results, presented in Table 1, demonstrate that removing any of these components negatively impacts the overall performance of ReARTeR. This highlights the importance of each component in enhancing the reasoning capabilities of the RAG system. Moreover, these results validate that the unbiased PRM training data collection strategy designed to address the untrustworthy challenges of process reward models enables the training of a more reliable PRM, which provides accurate reward scores for reasoning steps. Additionally, the combination of a more accurate PRM with the TD-based lookahead search enhances the feedback provided during the refinement stage. By leveraging explanations generated by the PEM during the refinement phase, ReARTeR achieves better reasoning step improvements compared to using PRM scores alone.

4.4 RQ3: Post-training iterations analysis.

In this section, we analyze the impact of the number of iterations in the Step-Level Offline Reinforcement Stage during the post-training scaling of ReARTeR on the reasoning capabilities of RAG systems. In this experiment, we used LLaMA-3.1-8B as the generator and conducted three iterations of Offline Reinforcement, testing the system on four multi-step reasoning datasets. The experimental results in Figure 4 demonstrate that the performance of the RAG system on multi-step reasoning datasets improves significantly as the number of Offline Reinforcement iterations increases. Additionally, the results show that our algorithm achieves stable performance improvements across iterations, validating that the proposed Step-Level Offline Reinforcement method provides effective and consistent updates. Due to resource constraints, we did not verify the scalability of our approach on larger datasets or with additional iterations. However, based on the current experimental results, we observe a promising scaling property, suggesting the potential for even greater improvements under resource-abundant conditions.

Table 2: Ablation Study of ReARTeR across different generators and datasets.

Model	Ablation	2WikiMultiHopQA		Bamboogle		HotpotQA		Musique	
		ACC_R	ACC_L	ACC_R	ACC_L	ACC_R	ACC_L	ACC_R	ACC_L
GPT4o-mini	w/o Refinement	0.522	0.466	0.474	0.522	0.424	0.456	0.282	0.276
	w/o PEM	0.532	0.484	0.486	0.532	0.426	0.462	0.284	0.286
	w/o TD-Lookahead	0.524	0.490	0.488	0.540	0.458	0.494	0.290	0.294
	w/o Beam Search	0.526	0.492	0.482	0.522	0.442	0.474	0.278	0.272
	w/o PRM Data	0.536	0.476	0.474	0.534	0.464	0.504	0.288	0.290
	ReARTeR	**0.554**	**0.534**	**0.496**	**0.544**	**0.468**	**0.506**	**0.296**	**0.302**
Llama-3.1-8B	w/o Refinement	0.444	0.334	0.418	0.440	0.402	0.424	0.230	0.238
	w/o PEM	0.450	0.340	0.420	0.446	0.406	0.416	0.234	0.218
	w/o TD-Lookahead	0.462	0.352	0.428	0.454	0.414	0.438	0.222	0.242
	w/o Beam Search	0.452	0.346	0.424	0.448	0.416	0.420	0.236	0.246
	w/o PRM Data	0.466	0.350	0.416	0.458	0.406	0.400	0.238	0.232
	ReARTeR	**0.470**	**0.364**	**0.438**	**0.484**	**0.424**	**0.434**	**0.244**	**0.252**



(a) The impact of alignment between PEM and PRM on the improvement rate of process reward scores.

(b) The impact of PEM on ReARTeR's final performance before and after alignment with PRM.

Figure 5: The impact of aligning PEM and PRM on ReARTeR's overall performance with GPT4-o-mini as generator.

4.5 RQ4: The effective of RARTPR in aligning PEM and PRM.

To evaluate the effectiveness of the alignment strategy for PEM and PRM proposed in RARTPR, we first calculated the improvement rate of process reward scores for reasoning steps with low initial scores after refinement using explanations generated by PEM, both before and after alignment. As shown in Figure 5(a), before aligning PEM with PRM (*w/o PEM Align*), the improvement rate achieved using explanations from an off-the-shelf LLM-based PEM was only around 50%. This result indicates that PEM struggles to produce accurate explanations aligned with PRM scores, making it difficult for the RAG system to leverage these explanations to refine reasoning steps and improve PRM scores. In contrast, after aligning PEM and PRM (*w PEM Align*), we observed a significant increase in the improvement rate, validating the effectiveness of the alignment strategy for enhancing the refinement process and improving reasoning quality. Furthermore, as shown in Figure 5(b), we directly compared the accuracy of RARTPR in solving complex multi-hop queries before and after aligning PEM with PRM. The results demonstrate consistent improvements across multiple datasets after alignment, further confirming the effectiveness of the proposed alignment strategy for PEM and PRM in RARTPR.

5 Conclusion

We propose **ReARTeR**, a framework that enhances the multi-step reasoning capabilities of RAG systems through both post-training and test-time scaling. ReARTeR integrates Trustworthy Process Rewarding, which combines a Process Reward Model for accurate scoring and a Process Explanation Model for explanation-based refinements. During post-training, step-level offline reinforcement fine-tuning with MCTS generates high-quality preference data to optimize the generator. ReARTeR addresses key reasoning challenges, including misalignment between PEM and PRM, bias in PRM training data, and early-step bias in PRM scores, through off-policy preference learning, balanced annotation strategies, and a temporal-difference-based look-ahead search. Experiments on multi-step reasoning benchmarks show that ReARTeR outperforms existing methods, demonstrating its effectiveness in enhancing RAG systems for knowledge-intensive tasks. Based on the reliable PRM technique and the natural step-wise decomposition characteristic of Deep (Re)search [26], we believe that ReARTeR will have broader applications in future Deep (Re)search scenarios.

Acknowledgments

This work was funded by the National Key R&D Program of China (2023YFA1008704), the National Natural Science Foundation of China (62472426, 62376275), Beijing Key Laboratory of Big Data Management and Analysis Methods, Major Innovation & Planning Interdisciplinary Platform for the "Double-First Class" Initiative, funds for building world-class universities (disciplines) of Renmin University of China. Supported by Kuaishou Technology. Supported by the Outstanding Innovative Talents Cultivation Funded Programs 2025 of Renmin University of China.

References

[1] 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* (2023).

[2] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games* 4, 1 (2012), 1–43.

[3] Guanting Dong, Chenghao Zhang, Mengjie Deng, Yutao Zhu, Zhicheng Dou, and Ji-Rong Wen. 2024. Progressive Multimodal Reasoning via Active Retrieval. *arXiv preprint arXiv:2412.14835* (2024).

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

[5] Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306* (2024).

[6] Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A Survey on RAG Meeting LLMs: Towards Retrieval-Augmented Large Language Models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Barcelona, Spain) *(KDD '24)*. Association for Computing Machinery, New York, NY, USA, 6491–6501. https://doi.org/10.1145/3637528.3671470

[7] Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics* 9 (2021), 346–361.

[8] 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*. 6609–6625.

[9] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. [n. d.]. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

[10] Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2024. Large Language Models Cannot Self-Correct Reasoning Yet. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=IkmD3fKBPQ

[11] Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong 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)*, Kevin Duh, Helena Gomez, and Steven Bethard (Eds.). Association for Computational Linguistics, Mexico City, Mexico, 7036–7050. https://doi.org/10.18653/v1/2024.naacl-long.389

[12] 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)*. 7029–7043.

[13] Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. *arXiv preprint arXiv:2310.06839* (2023).

[14] Jiajie Jin, Yutao Zhu, Xinyu Yang, Chenghao Zhang, and Zhicheng Dou. 2024. FlashRAG: A Modular Toolkit for Efficient Retrieval-Augmented Generation Research. *arXiv preprint arXiv:2405.13576* (2024).

[15] Jaehyung Kim, Jaehyun Nam, Sangwoo Mo, Jongjin Park, Sang-Woo Lee, Minjoon Seo, Jung-Woo Ha, and Jinwoo Shin. [n. d.]. SuRe: Summarizing Retrievals using Answer Candidates for Open-domain QA of LLMs. In *The Twelfth International Conference on Learning Representations*.

[16] Jaehyung Kim, Jaehyun Nam, Sangwoo Mo, Jongjin Park, Sang-Woo Lee, Minjoon Seo, Jung-Woo Ha, and Jinwoo Shin. 2024. SuRe: Summarizing Retrievals using Answer Candidates for Open-domain QA of LLMs. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=w4DW6qkRmt

[17] Haitao Li, Qian Dong, Junjie Chen, Huixue Su, Yujia Zhou, Qingyao Ai, Ziyi Ye, and Yiqun Liu. 2024. LLMs-as-Judges: A Comprehensive Survey on LLM-based Evaluation Methods. *arXiv preprint arXiv:2412.05579* (2024).

[18] Xingxuan Li, Weiwen Xu, Ruochen Zhao, Fangkai Jiao, Shafiq Joty, and Lidong Bing. 2024. Can We Further Elicit Reasoning in LLMs? Critic-Guided Planning with Retrieval-Augmentation for Solving Challenging Tasks. *arXiv preprint arXiv:2410.01428* (2024).

[19] Yucheng Li, Bo Dong, Frank Guerin, and Chenghua Lin. 2023. Compressing Context to Enhance Inference Efficiency of Large Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 6342–6353. https://doi.org/10.18653/v1/2023.emnlp-main.391

[20] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050* (2023).

[21] Chris Yuhao Liu, Liang Zeng, Jiacai Liu, Rui Yan, Jujie He, Chaojie Wang, Shuicheng Yan, Yang Liu, and Yahui Zhou. 2024. Skywork-Reward: Bag of Tricks for Reward Modeling in LLMs. *arXiv preprint arXiv:2410.18451* (2024).

[22] Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, et al. 2024. Improve Mathematical Reasoning in Language Models by Automated Process Supervision. *arXiv preprint arXiv:2406.06592* (2024).

[23] Trung Quoc Luong, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. 2024. Reft: Reasoning with reinforced fine-tuning. *arXiv preprint arXiv:2401.08967* (2024).

[24] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2024. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems* 36 (2024).

[25] Skywork o1 Team. 2024. Skywork-o1 Open Series. https://huggingface.co/Skywork. https://huggingface.co/Skywork

[26] OpenAI. 2024. *Introducing Deep Research.* https://openai.com/index/introducing-deep-research/

[27] Richard Yuanzhe Pang, Weizhe Yuan, Kyunghyun Cho, He He, Sainbayar Sukhbaatar, and Jason Weston. 2024. Iterative reasoning preference optimization. *arXiv preprint arXiv:2404.19733* (2024).

[28] 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* (2022).

[29] 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*. 5687–5711.

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

[31] 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. *arXiv preprint arXiv:2305.15294* (2023).

[32] Chenglei Shen, Xiao Zhang, Teng Shi, Changshuo Zhang, Guofu Xie, and Jun Xu. 2024. A survey of controllable learning: Methods and applications in information retrieval. *arXiv preprint arXiv:2407.06083* (2024).

[33] 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* (2023).

[34] Avi Singh, John D Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J Liu, James Harrison, Jaehoon Lee, Kelvin Xu, et al. 2023. Beyond human data: Scaling self-training for problem-solving with language models. *arXiv preprint arXiv:2312.06585* (2023).

[35] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314* (2024).

[36] ZhongXiang Sun, Zihua Si, Xiaoxue Zang, Kai Zheng, Yang Song, Xiao Zhang, and Jun Xu. 2025. LargePiG for Hallucination-Free Query Generation: Your Large Language Model is Secretly a Pointer Generator. In *THE WEB CONFERENCE 2025*. https://openreview.net/forum?id=MyywdOeyn0

[37] ZhongXiang Sun, Xiaoxue Zang, Kai Zheng, Jun Xu, Xiao Zhang, Weijie Yu, Yang Song, and Han Li. 2025. ReDeEP: Detecting Hallucination in Retrieval-Augmented Generation via Mechanistic Interpretability. In *The Thirteenth International Conference on Learning Representations*. https://openreview.net/forum?id=ztzZDzgfrh

[38] Richard Sutton. 2019. The Bitter Lesson. http://incompleteideas.net/IncIdeas/BitterLesson.html Incomplete Ideas (blog), 13(1):38.

[39] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[40] Gerald Tesauro. 1995. Temporal difference learning and TD-Gammon. *Commun. ACM* 38, 3 (March 1995), 58–68. https://doi.org/10.1145/203330.203343

[41] 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 (2022), 539–554.

[42] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 10014–10037.

[43] Luong Trung, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. 2024. ReFT: Reasoning with Reinforced Fine-Tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long*

Papers), Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 7601–7614. https://doi.org/10.18653/v1/2024.acl-long.410

[44] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533* (2022).

[45] Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 9426–9439.

[46] Yile Wang, Peng Li, Maosong Sun, and Yang Liu. 2023. Self-knowledge guided retrieval augmentation for large language models. *arXiv preprint arXiv:2310.05002* (2023).

[47] Zihan Wang, Yunxuan Li, Yuexin Wu, Liangchen Luo, Le Hou, Hongkun Yu, and Jingbo Shang. 2024. Multi-step problem solving through a verifier: An empirical analysis on model-induced process supervision. *arXiv preprint arXiv:2402.02658* (2024).

[48] Zhenyu Wu, Qingkai Zeng, Zhihan Zhang, Zhaoxuan Tan, Chao Shen, and Meng Jiang. 2024. Enhancing Mathematical Reasoning in LLMs by Stepwise Correction. *arXiv preprint arXiv:2410.12934* (2024).

[49] Yuxi Xie, Anirudh Goyal, Wenyue Zheng, Min-Yen Kan, Timothy P Lillicrap, Kenji Kawaguchi, and Michael Shieh. 2024. Monte Carlo Tree Search Boosts Reasoning via Iterative Preference Learning. *arXiv preprint arXiv:2405.00451* (2024).

[50] Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2024. RECOMP: Improving Retrieval-Augmented LMs with Context Compression and Selective Augmentation. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=mlJLVigNHp

[51] 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*. 2369–2380.

[52] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations (ICLR)*.

[53] Zhenrui Yue, Honglei Zhuang, Aijun Bai, Kai Hui, Rolf Jagerman, Hansi Zeng, Zhen Qin, Dong Wang, Xuanhui Wang, and Michael Bendersky. 2024. Inference scaling for long-context retrieval augmented generation. *arXiv preprint arXiv:2410.04343* (2024).

[54] 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* (2024).

[55] Yuxiang Zhang, Yuqi Yang, Jiangming Shu, Yuhang Wang, Jinlin Xiao, and Jitao Sang. 2024. OpenRFT: Adapting Reasoning Foundation Model for Domain-specific Tasks with Reinforcement Fine-Tuning. *arXiv preprint arXiv:2412.16849* (2024).

[56] Yu Zhao, Huifeng Yin, Bo Zeng, Hao Wang, Tianqi Shi, Chenyang Lyu, Longyue Wang, Weihua Luo, and Kaifu Zhang. 2024. Marco-o1: Towards open reasoning models for open-ended solutions. *arXiv preprint arXiv:2411.14405* (2024).