Q-PRM: Adaptive Query Rewriting for Retrieval-Augmented Generation via Step-level Process Supervision

Xiaopeng Ye[1], Chen Xu[1], Chaoliang Zhang[2], Zhaocheng Du[2], Jun Xu[1]*, Gang Wang[2], Zhenhua Dong[2],
[1]Gaoling School of Artificial Intelligence, Renmin University of China
[2]Huawei Noah's Ark Lab
{xpye, xc_chen, junxu}@ruc.edu.cn
{zhangchaoliang, zhaochengdu, wanggang110, dongzhenhua}@huawei.com

Abstract

Query rewriting plays a pivotal role in Retrieval-Augmented Generation (RAG) by refining real-world queries of varying complexity. Existing approaches typically rely on outcome-supervised training or heuristic rules to guide the rewriting process. However, these paradigms often struggle to handle queries with varying levels of complexity, posing over- and under-refinement problems. We identify the root cause of these issues as the absence of supervision signals for intermediate steps. To fully construct and utilize such signals, we propose Q-PRM, a novel query rewriting framework. Q-PRM reformulates the rewriting process as a Markov Decision Process (MDP) composed of atomic rewriting steps. In this way, Q-PRM can apply process-level supervision to each atomic step according to the query type, offering more targeted and effective guidance. Q-PRM comprises three key stages: (1) applying Monte Carlo Tree Search to generate step-level process supervision signals; (2) performing reinforced self-training for progressive process refinement; and (3) employing PRM-guided decoding during inference. Experiments on several open-domain QA benchmarks demonstrate that Q-PRM consistently outperforms baselines across different levels of query complexity.

1 Introduction

In real-world applications, Retrieval-Augmented Generation (RAG) systems need to handle user queries that vary widely in complexity (e.g., multi-hop reasoning, ambiguity) (Jeong et al., 2024; Gao et al., 2023; Zhang et al., 2025a). This need arises from the users' diverse expression styles, intents, and background knowledge (Kharitonov et al., 2013; Spatharioti et al., 2023). To improve retrieval effectiveness and answer generation quality, it is essential to transform raw queries into syntactically clear and retrieval-friendly forms. This process, termed *query rewriting*, has emerged as a crucial pre-processing step in RAG (Song and Zheng, 2024; Ma et al., 2023).

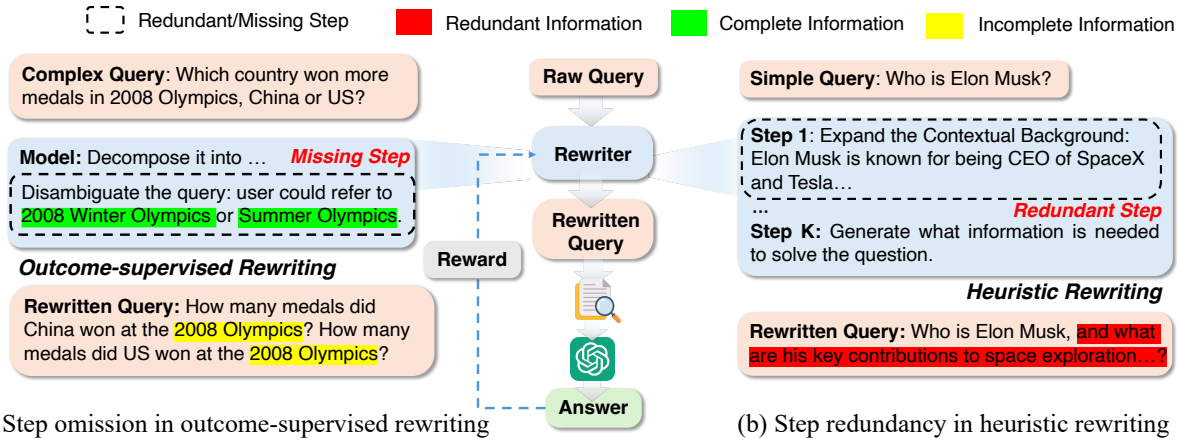
Existing query rewriting methods primarily rely on either heuristic rules (Jagerman et al., 2023) or outcome-based training (Ma et al., 2023) to perform single-step or multi-step rewriting (Song and Zheng, 2024). However, these approaches often fail to adapt to queries of varying complexity, posing over- and under-refinement challenges for real-world deployment. We conduct experiments to better illustrate these challenges.

As shown in Figure 1 (a), outcome-supervised approaches utilize downstream answer quality as the reward to train the rewriter. However, our experiment finds they tend to omit critical steps when handling complex queries (e.g., ambiguous, multi-hop), causing the under-refinement challenge. For example, the ambiguous query "`Which country won more medals in the 2008 Olympics, China or the US?`" requires disambiguation (e.g., distinguishing between the Summer and Winter Olympics). However, the outcome-supervised rewriter merely decomposes the query without resolving the ambiguity, resulting in sub-queries that lack essential context (i.e., under-refinement). On the other hand, heuristic-based methods refine queries using fixed information expansion strategies (Chan et al., 2024; Baek et al., 2024). However, such approaches often cause step redundancy when rewriting simple queries, leading to over-refinement in the final rewritten query. As in Figure 1 (b), the example simple query "`Who is Elon Musk?`" is unnecessarily expanded with excessive background information, introducing redundant information into the rewritten version.

The main reason for such over- and under-refinement issues stems from the lack of supervision signals under each intermediate rewriting step (He et al., 2021). Instead of relying solely on the final answer, the process supervision paradigm

* Corresponding author

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 15113–15128
November 4-9, 2025



(a) Step omission in outcome-supervised rewriting

(b) Step redundancy in heuristic rewriting

Figure 1: Existing query rewriting methods often struggle to adapt to queries of varying complexity, leading to two key issues: (a) For complex queries, they tend to overlook critical intermediate reasoning steps, resulting in under-refinement; (b) For simple queries, they frequently introduce unnecessary steps, leading to over-refinement;

provides step-level guidance using the Process Reward Model (PRM) (Song et al., 2025; Chen et al., 2024). PRM serves as a valuable reminder for the query rewriter to pay attention to essential steps when handling queries of varying complexity. In this way, the rewriter can better preserve essential steps when handling complex queries, while avoiding unnecessary ones for simpler queries.

However, two primary challenges emerge when applying process supervision to the task: (1) the absence of process supervision signals to train the PRM (Mao et al., 2024); and (2) the lack of multi-step reasoning ability tailored for rewriting tasks, which prevents rewriters from effective incorporation of PRM signals. To address these challenges, we propose our method **Q-PRM**. Specifically, Q-PRM consists of three key steps: (1) Data Collection: to address the high cost of manually annotating process supervision signals (challenge 1), we propose an uncertainty-aware Monte Carlo Tree Search (MCTS) approach to automatically collect robust supervision signals for training the PRM; (2) Model Training: we apply reinforced self-training (Gulcehre et al., 2023) to enhance the rewriter's reasoning ability to handle queries of varying complexity (challenge 2); and (3) Inference: by combining the PRM with the rewriter, we propose a decoding method that leverages the PRM for fine-grained guidance, enhancing generalization to queries of varying complexity.

Our contribution can be summarized as follows:

- We emphasize the importance of process-level supervision to handle queries with varying complexity in RAG system.
- We propose Q-PRM, a novel query rewriting framework that leverages a process reward model to guide the rewriting steps in a more adaptive and interpretable manner.
- Extensive experiments demonstrate that our approach outperforms existing baselines on varying datasets with different query complexity.

2 Related Works

Query Rewriting in RAG As a crucial step in the pre-retrieval stage of RAG process (Gao et al., 2023; Zhang et al., 2025b), existing works can be categorized into training-free (heuristic) and training-based (Song and Zheng, 2024).

Training-free methods leverage LLMs' world knowledge to heuristically expand queries for better retrieval. Some works (Gao et al., 2022; Jagerman et al., 2023) (e.g., Query2Doc) converted raw queries into pseudo-documents using the internal knowledge of LLMs. Other works (Baek et al., 2024; Chan et al., 2024) utilize heuristic rules to expand the query in a multi-step way. Although efficient, these methods will inevitably introduce unnecessary and redundant information to the query due to hallucinations of LLMs (Huang et al., 2025).

For training-based methods, existing works mainly use outcome supervision to fine-tune the rewriter model. Some works (Ma et al., 2023; Cong et al., 2024; Wang et al., 2024b) utilize reinforcement learning (e.g., PPO (Schulman et al., 2017)) to train the rewriter model with downstream task quality (i.e., generation, retrieval, re-ranking (Mao et al., 2024)) as the rewards. However, since downstream task performance depends not only on the quality of query rewriting but also on other factors

(e.g., query difficulty (Jeong et al., 2024), retrieval quality (Wang et al., 2024b), and LLM internal knowledge (Zhao et al., 2023)), the reward is often unstable and sparse (Wang et al., 2024b), making it difficult for the rewrite model to effectively learn the optimal actions.

Process Supervision for RAG. RAG is widely adopted for knowledge-intensive tasks (Press et al., 2022; Li et al., 2024b; Zhang et al., 2025a), but it still faces limitations in complex multi-step reasoning. Existing methods integrate RAG with Chain-of-Thought (CoT) reasoning (Trivedi et al., 2022; Press et al., 2022). Since direct CoT often falters on challenging tasks (e.g., competitive programming), CR-Planner (Li et al., 2024b) attempts to use PRM to assist search and improve the reasoning ability of RAG systems through test-time scaling. AutoPRM (Chen et al., 2024) decomposed complex problems into subproblems while leveraging a self-supervisedly trained PRM to provide feedback, thereby enhancing multi-step logical reasoning tasks. To improve the trustworthiness in the reasoning process, ReARTeR (Sun et al., 2025) introduces the Process Explanation Model to provide explanations for the reasoning process. Currently, most works use process supervision to enhance the in-retrieval and post-retrieval stages (Gao et al., 2023), but its potential in the pre-retrieval stage to refine user queries has not yet been explored.

3 Preliminary

Retrieval-Augmented Generation. Consider a knowledge-intensive task (e.g., open-domain question answering), where the dataset comprises multiple query-answer instances. Each instance consists of a raw query q paired with its corresponding ground-truth answer y. A RAG system $f(\cdot)$ takes a query q as input and returns an answer $\hat{y}_q = f(q)$. The RAG process begins with a retriever that identifies a set of relevant documents $\mathcal{D}$ based on q. These documents are then passed to an LLM reader, which generates the final response. The effectiveness of a RAG system is typically assessed by evaluating the quality of the generated response $\hat{y}_q$ using metrics such as Exact Match and F1 Score (Rajpurkar et al., 2016; Yu et al., 2024), which compare the generated answer $\hat{y}_q$ to the ground-truth answer y. In this paper, we denote this evaluation function as $\text{sim}(\hat{y}_q, y)$.

Query Rewriting as Markov Decision Process. Given a query q, the goal of query rewriting is to refine q into a rewritten query $q' = \mathcal{M}(q)$, where $\mathcal{M}$ is the rewriter. We formulate the rewriting process as a multi-step MDP $(\mathcal{S}, \mathcal{Q}, \mathcal{A}, \mathcal{R}, \mathcal{P})$, where each MDP trajectory starts with the query q and ends with final rewritten sub-queries q'. Each state s_t is related to the current rewritten query $q'_t \in \mathcal{Q}$ and a response $\hat{y}_{q'_t}$). At each rewriting state s_t , the rewriter $\mathcal{M}$ can apply an atomic rewriting action $a_t \in \mathcal{A}$, where $\mathcal{A} = \{a_{\texttt{decomp}}, a_{\texttt{disamb}}, a_{\texttt{abst}}, a_{\texttt{exp}}\}$ contains four atom actions (decomposition, disambiguation, abstraction, expansion) as defined by Song and Zheng (2024). Then s_t transitions to a new state s_{t+1} until the final state. A process reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ evaluates each decision step individually by assigning a fine-grained reward to each state s_t.

4 Our Method: Q-PRM

In this section, we present our proposed method, Q-PRM. The overall workflow of Q-PRM is illustrated in Figure 2. Specifically, Q-PRM consists of three key steps. (1) **Data Collection**: leveraging MCTS to gather process supervision signal for query rewriting; (2) **Model Training**: using the collected data to train the rewriting model $\mathcal{M}$ and the PRM; (3) **PRM-Guided Inference**: introducing a PRM-guided search decoding mechanism for inference.

4.1 Tree Search for Data Collection

To implement the idea of guiding the query rewriting process through process supervision, an essential step is to train a PRM, which can provide rewards tailored to the rewriting steps of queries with different levels of complexity. However, the lack of publicly available process reward data for query rewriting poses a significant challenge. To address this issue, we draw inspiration from the success of existing mathematical and code generation tasks (Li et al., 2024a) and adopt MCTS to collect high-quality process reward signals for the rewriting process, which are then used to train the PRM (see Section 4.2).

As illustrated in Figure 2, MCTS iteratively performs four key stages: *selection*, *expansion*, *evaluation*, and *backpropagation* [1]. To adapt MCTS to query rewriting tasks with varying complexity, we introduce targeted designs for the selection and evaluation stage.

[1]Please refer to Appendix A for the details of the four stages in the traditional MCTS algorithm.

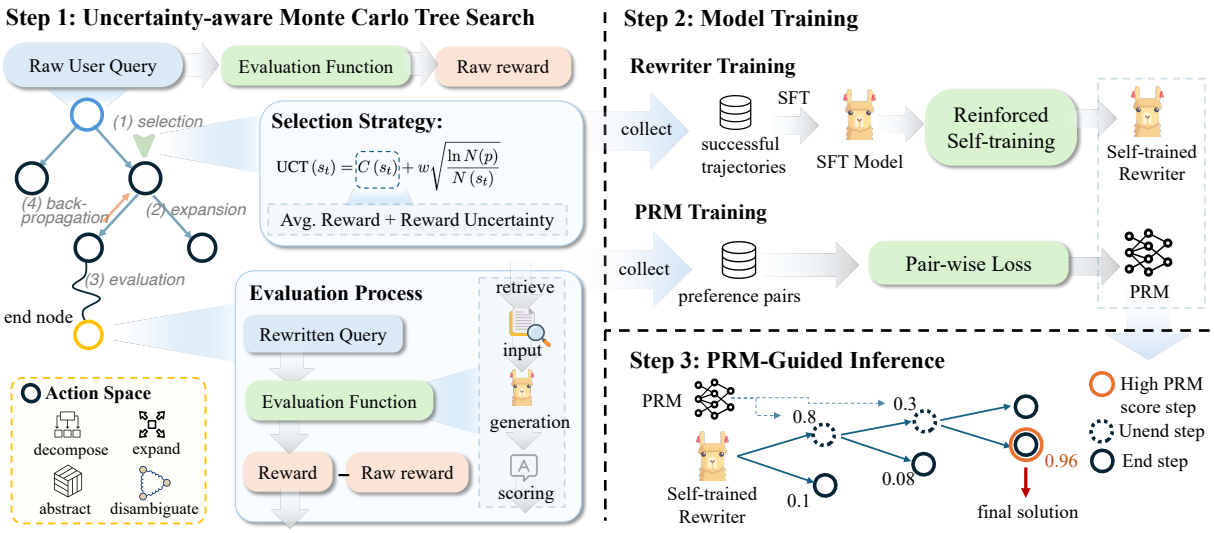


Figure 2: The overall workflow of the proposed Q-PRM method consists of three steps: (1) we perform uncertainty-aware MCTS to generate process-level supervision signals and collect trajectory data for query rewriting; (2) we use the collected data to train both the PRM and the rewriter; (3) during inference, the trained PRM is used to guide the multi-step rewriting process.

Evaluation Stage. During the evaluation stage, we adopted a random rollout strategy (Zhang et al., 2024), where actions were randomly sampled from a given state until a terminal state was reached. A state s_t is considered terminal when one of the following conditions is met: (1) the rewriter $\mathcal{M}$ determines that no further rewriting is possible; (2) the maximum rewriting depth is reached; or (3) the evaluation score of the current state exceeds a predefined threshold, θ, i.e., $\text{sim}(\hat{y}_{q'_t}, y) \geq \theta$. To improve reward estimation, multiple rollouts are conducted for each simulation. Specifically, the reward of node s_t is the average result of N rollouts, i.e., $r = \frac{1}{N}\sum_{n=1}^{N} r^{(n)}$. To eliminate the influence of query complexity from the reward, we use the difference between the evaluation score of the rewritten query $q'^{(n)}$ and that of the raw query q as the reward.

$$r^{(n)} = \text{sim}(y_{q'^{(n)}}, y) - \text{sim}(y_q, y), \quad (1)$$

Selection Stage. During the MCTS process, the early reward estimate of the node is usually inaccurate and biased due to random rollout (Oren et al., 2024). To obtain a more accurate reward estimation, we employ an uncertainty-aware UCT criterion in the selection phase:

$$\text{UCT}(s_t) = C(s_t) + w\sqrt{\frac{\ln N(p)}{N(s_t)}} \quad (2)$$

where w is a hyperparameter that balances the exploitation (i.e., node value $C(s_t)$) and exploration (i.e., visit count $N(s_t)$), and p denotes the parent node of s_t. $C(s_t)$ is the uncertainty-aware node value, which combines the value (average node reward) $V(s_t)$ and its standard deviation:

$$C(s_t) = V(s_t) + \beta\text{Std}(\{V^{(i)}(s_t)\}_{i=1}^{N(s_t)}) \quad (3)$$

Intuitively, some nodes exhibit large reward fluctuations due to unstable reward estimation, which will lead to higher $C(s_t)$. These nodes are more likely to be selected, increasing the chances of discovering successful nodes rather than being biased toward high but potentially inaccurate values. As iteration progresses, value estimates $V(s_t)$ stabilize, and $C(s_t)$ gradually converges to the true value of node.

4.2 Model Training

After MCTS simulations, we collect the process supervision signals and trajectories to fine-tune the rewriter and PRM, respectively.

4.2.1 Rewriter Training

Since general-purpose LLMs cannot often perform query rewriting (Ye et al., 2023), we train a specialized rewriter to acquire the ability to decompose and refine queries through multiple rewriting steps. We first collect the high-reward trajectories from MCTS simulation to supervised fine-tune (SFT) an LLM $\mathcal{M}_\theta$ into $\mathcal{M}_{\text{SFT}}$. Then, we further enhance its multi-step rewriting ability through Reinforced Self-training (Gulcehre et al., 2023), as illustrated

Algorithm 1 PRM-Guided CoT Decoding

Require: User query q, initial state s_0, beam width k, trained PRM $R(\cdot)$
Ensure: Top sequence/path

Initialize beam $B \leftarrow \{s_0\}$
for $t = 1$ to N **do**
  Sample m next-step candidates $\{s_t^i\}_{i=1}^m$ based on each existing step $s_{t-1} \in B$ and store their decoding confidence $\Delta(s_t^i)$.
  Predict process reward for each candidate using PRM: $R(s_t^i|q)$.
  Choose top-k candidates of s_t^i from all mk samples according to score (5).
  Update beam $B \leftarrow$ top-k candidates.
end for
return highest-reward steps in B

in Figure 2. In each self-training iteration, we sample M rewritten outputs from $\mathcal{M}_{\text{SFT}}$. We evaluate each one and retain those with rewards above a threshold ($r^{(n)} \geq \theta$). The model $\mathcal{M}_{\text{SFT}}$ is then further fine-tuned on the retained samples in an iterative manner.

4.2.2 PRM Training

Due to the influence of the internal knowledge embedded in LLMs (Zhao et al., 2023; Ma et al., 2023), the absolute reward values of RAG can fluctuate significantly across queries of different complexity. To ensure the robustness of the PRM's output scores, we do not estimate absolute reward values as in previous work (Sun et al., 2025). Instead, we adopt a pairwise loss to train the PRM to capture the relative differences between good and bad rewriting steps. For each query of different complexity, we perform K MCTS iterations and obtain multiple intermediate states s_t. We assume that a state $s_{t,1}$ is preferred over another state $s_{t,2}$ (denoted $s_{t,1} \succ s_{t,2}$) if its estimated value $V(s_{t,1})$ is greater than $V(s_{t,2})$. These preference pairs are then collected into a dataset: $\mathcal{D}_{\text{pref}} = \{(s_{t,1}, s_{t,2}|q) \mid s_{t,1} \succ s_{t,1}\}_{i=1}^N$. The PRM is then trained by minimizing the following pairwise loss:

$$\mathcal{L}_{\text{BT}} = -\mathbb{E}\left[\log \sigma\left(\mathcal{R}\left(s_{t,1}|q\right) - \mathcal{R}\left(s_{t,2}|q\right)\right)\right], \quad (4)$$

where σ is the sigmoid function, $\mathcal{R}(s_t, q)$ denotes the predicted reward for state s_t given query q.

4.3 PRM-Guided Decoding for Inference

After obtaining the self-trained rewriter $\mathcal{M}_{\text{Self-Train}}$ and the trained PRM, we propose a PRM-Guided CoT decoding strategy during the inference stage. Specifically, for each user query q, this strategy selects the candidate sub-steps via process reward-weighted CoT-decoding probability (Wang and Zhou, 2024; Chen et al., 2024) as the new score. Mathematically, we have:

$$s_t = \arg\max_{s_t} \Delta\left(s_t \mid s_{<t}, q\right) \cdot \mathcal{R}\left(s_t|q\right), \quad (5)$$

where $R(s_t|q_i)$ is the process reward predicted by PRM R, and $\Delta(\cdot)$ is the CoT decoding confidence function (Wang and Zhou, 2024), which measures the probability disparity between the top and secondary tokens in the sub-step:

$$\Delta(s_t \mid s_{<t}, q) = \frac{1}{|s_t|} \sum_{y_n \in s_t} [P_{\mathcal{M}}\left(y_i^1 \mid y_{<i}, q\right) - P_{\mathcal{M}}\left(y_i^2 \mid y_{<i}, q\right)]. \quad (6)$$

Here y_i^1 and y_i^2 represent the top two tokens at the i-th decoding step in query-rewriting sub-step s_t. $|s_t|$ means the number of tokens contained in sub-step s_t.

5 Experiment

5.1 Experimental Details

Datasets. We conduct extensive experiments on four English open-domain QA datasets containing queries of varying complexities to validate the effectiveness of our method, including multi-hop query (i.e., HotpotQA(Yang et al., 2018), StrategyQA (Geva et al., 2021)), ambiguous query (i.e., AmbigQA (Min et al., 2020)) and single-hop query (i.e., PopQA (Mallen et al., 2022)). To validate the effectiveness of our method in Chinese, we conduct an online A/B test on a commercial Chinese RAG system, which handles hundreds of thousands of real-world user queries every day.

Baselines. We compare Q-PRM with a wide range of SOTA models, which include both heuristic methods and outcome-supervised methods. For heuristic methods, **LLM Rewrite** directly enables the LLMs to rewrite the original query with a few-shot prompt. **HyDE** (Gao et al., 2022) and **Query2Doc** (Jagerman et al., 2023) convert queries into pseudo-documents to improve retrieval efficiency. **Crafting** (Baek et al., 2024) rewrites the query through multiple predefined rewriting steps.

Table 1: Evaluation results on queries of varying complexity, assessed using multi-hop, ambiguous, and single-hop open-domain QA datasets. For all three metrics reported, higher values indicate better performance. Both LLaMA3-8B and Qwen2.5-7B are used as rewriters. Bold indicates the best result, and underline denotes the second-best.

	Method	Multi-Hop						Ambiguous			Single-Hop		
		HotpotQA			StrategyQA			AmbigQA			PopQA		
		EM	F1	Acc	EM	F1	Acc	EM	F1	Acc	EM	F1	Acc
	No Retrieval	17.40	24.57	19.60	68.20	68.20	70.00	15.20	25.55	29.60	9.80	13.67	17.20
	Standard	30.40	41.16	37.00	63.00	63.00	67.40	46.80	59.43	64.20	10.60	28.40	63.00
LLama3-8B-Instruct	LLM Rewrite	23.80	33.44	34.60	39.20	39.20	53.80	40.00	53.67	61.20	9.20	24.77	58.60
	HyDE	31.80	42.87	41.00	61.40	61.40	70.60	45.40	59.85	66.60	9.60	25.13	51.20
	Query2Doc	30.60	40.28	37.00	64.60	64.60	68.80	45.60	58.92	64.00	6.20	15.12	35.80
	Crafting	30.00	38.95	35.20	65.60	65.60	68.20	44.40	56.99	62.60	6.40	16.16	39.20
	SFT	30.00	39.99	36.00	64.00	64.00	67.20	44.80	56.99	61.80	9.40	29.04	62.40
	RaFe$_{(DPO)}$	31.00	40.60	37.60	62.40	62.40	67.20	46.20	59.08	66.20	9.80	26.10	58.40
	RRR$_{(PPO)}$	31.60	42.90	37.60	66.20	66.20	68.80	48.80	**61.97**	67.40	10.40	27.98	62.20
	Q-PRM	**33.60**	**43.65**	**44.80**	**66.40**	**66.40**	**72.80**	**49.40**	61.62	**71.60**	**10.80**	**28.60**	**63.80**
Qwen2.5-7B-Instruct	LLM Rewrite	25.80	34.34	30.60	61.40	61.40	65.40	33.40	44.02	48.60	9.20	23.22	52.80
	HyDE	32.80	43.81	38.60	67.20	67.20	71.00	47.20	60.35	64.60	9.00	22.65	52.80
	Query2Doc	30.80	41.25	36.80	63.60	63.60	67.60	48.60	60.77	65.20	8.20	19.52	44.20
	Crafting	28.80	39.29	35.40	65.20	65.20	69.00	42.80	55.13	60.80	7.60	18.36	44.80
	SFT	29.80	39.39	36.20	63.60	63.60	67.60	43.20	55.99	62.20	9.00	22.32	54.00
	RaFe$_{(DPO)}$	32.60	43.24	38.40	68.80	68.80	71.80	48.00	60.76	64.40	11.40	26.67	58.00
	RRR$_{(PPO)}$	33.60	44.19	39.20	68.60	68.60	71.20	47.00	59.89	64.40	12.20	28.93	62.60
	Q-PRM	**34.20**	**44.46**	**39.40**	**70.20**	**70.20**	**72.60**	**51.00**	**63.90**	**69.20**	12.20	**29.75**	**63.60**

For outcome-supervised baselines, Following (Mao et al., 2024), **SFT** uses the pre-generated rewrites to directly train the rewrite model. **RaFe** utilizes re-ranker feedback to construct good-bad rewriting pairs to train the rewriter via DPO (Rafailov et al., 2023). **RRR** (Ma et al., 2023) trains the rewriter using the quality of generated answers and answer hit rate in the retrieved documents as the reward. For the detailed specifics of the baseline, please refer to Appendix B.

Evaluation Metrics. For offline evaluation metrics, following previous works (Sun et al., 2025; Cong et al., 2024), we use Exact Match (EM), F1 score (F1), and accuracy (Acc). For online test evaluation, since the real-world RAG system typically lacks explicit feedback (e.g., click-through rate) and ground-truth answer, we rely on a professional annotation team to conduct human assessments on rewritten queries. The evaluation criteria are detailed in the Appendix C.

Implementation Details. Our model is developed and evaluated using the FlashRAG (Jin et al., 2024) framework. We employ the widely-used dense retriever E5 [2] to collect relevant documents from the English Wikipedia dump from December 20, 2018. The experiments are implemented on Qwen-max [3] as the LLM reader and Qwen 2.5-0.5B as the PRM. To mimic real-world queries of varying complexity, we randomly selected 10,000 samples from the training set and 500 samples from the test (dev) set of each dataset to construct new training and evaluation sets for our experiments. This combined training dataset is then used to train both the outcome-supervised and our process-supervised rewriting methods to achieve their best performances.

5.2 Main Results

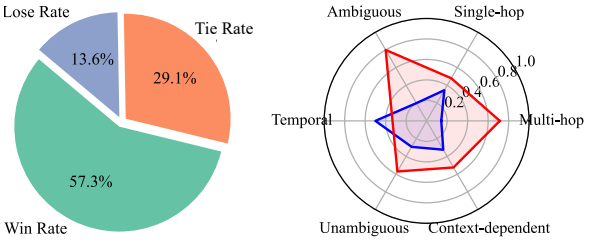
Offline Experiment (EN). Table 1 reports the performance of Q-PRM compared to baselines on four datasets comprising real-world queries with diverse levels of complexity. Results indicate that Q-PRM achieves the best performance compared with other baselines on both multi-hop, ambiguous, and single-hop tasks, which means Q-PRM can adaptively refine queries of varying complexity. Specifically, we find that both heuristic (e.g., HyDE and Query2Doc) and outcome-supervised rewriting methods (e.g., RRR) exhibited instability performance across different complexities. For instance, although heuristic methods like HyDE perform well on more complex queries

[2] https://huggingface.co/intfloat/e5-large-v2

[3] https://www.aliyun.com

Table 2: Ablation Study of Q-PRM across different datasets using Qwen2.5-7B-Instruct as the rewriter.

	HotpotQA			StrategyQA			AmbigQA			PopQA		
	EM	F1	Acc	EM	F1	Acc	EM	F1	Acc	EM	F1	Acc
Q-PRM	**33.60**	**44.19**	**39.20**	**70.20**	**70.20**	**72.60**	**51.00**	**63.90**	**69.20**	**12.20**	**29.75**	**63.60**
w/o PRM	30.40	40.87	37.60	67.40	67.40	70.20	45.20	56.48	64.40	11.60	28.23	57.00
w/o CoT decoding	33.00	43.64	39.00	68.00	68.00	70.20	46.20	58.87	62.80	10.80	24.86	51.40
w/o Self-Training	30.80	41.09	36.00	63.80	63.80	67.20	47.00	59.00	63.80	9.40	25.00	53.20
w/o SFT	26.60	35.74	31.80	62.00	62.00	64.80	37.80	49.29	52.60	8.60	22.01	49.00



(a) Online A/B test result (b) WinRate on different types of query

Figure 3: Online A/B Test Results of Q-PRM. (a) Win/Tie/Loss rates of Q-PRM compared to the existing production system. (b) Win rates of Q-PRM and the baseline system across queries of varying complexity.

(e.g., AmbigQA), they fail to effectively rewrite simple queries, such as those in the PopQA dataset. Moreover, when using Qwen2.5-7B as the rewriter, outcome-supervised methods (e.g., RRR) perform even worse than heuristic baselines on the AmbigQA dataset. We attribute this to the instability of outcome-supervised rewards across queries of varying complexity. In contrast, Q-PRM leverages process supervision signals and achieves superior performance across multiple tasks.

Online A/B Test (ZH). We conducted an online A/B test in a real-world Chinese RAG system by deploying our Q-PRM (7B) alongside the original production model (referred to as the old system) of the same size. Both models were exposed to real user traffic via shadow deployment.

Based on human evaluations of the rewritten queries, we report the Win/Tie/Lose rates, which represent the proportions of cases where our model's output was preferred, rated as equivalent, or less preferred compared to the old system. As shown in Figure 3 (a), Q-PRM achieved a win rate of 57.3% with only a 13.6% Lose Rate, demonstrating its strong overall performance. Furthermore, as illustrated in Figure 3 (b), our model consistently outperforms the old system across user queries of varying complexity, showing significant advantages in WinRate across all categories of query.

5.3 Analysis Experiment

5.3.1 Ablation Study

We conduct ablation studies by systematically removing each component at a time from Q-PRM to investigate the effectiveness of each component. The results presented in Table 2 highlight the significance of each element. Specifically, we evaluate the following configurations: (1) **w/o PRM**: removing the guidance of PRM and directly using CoT decoding (Wang and Zhou, 2024) when inference to analyze its effect on the rewriting process of Q-PRM. (2) **w/o CoT decoding**: utilizing the beam search when inferencing. (3) **w/o Self-Training**: Removing the Reinforced Self-Training Phase to directly use the SFT model to conduct query rewriting. (4) **w/o SFT**: directly using the original LLM without SFT and self-training to conduct query rewriting.

The experimental results, presented in Table 2, demonstrate that removing any of these components negatively impacts the overall performance of Q-PRM. This highlights the importance of each component in enhancing the query rewriting of the RAG system. Moreover, the results in the table demonstrate the importance of process supervision from PRM for query rewriting. During inference with process supervision, Q-PRM performs significantly better than without process supervision, especially for multi-hop and ambiguous queries. For example, in the StrategyQA dataset, the EM score with the guidance of PRM is 4.15% higher than the EM score without PRM.

5.3.2 Generalization Across Tasks

To evaluate the generalization capability of PRM, we conduct experiments across a diverse set of RAG scenarios. Specifically, we consider four representative settings: (a) Open-Domain QA using the MRQA-NQ subset (Fisch et al., 2019), (b) Close-Domain QA with DomainRAG (Wang et al., 2024a), (c) Conversational QA on CORAL

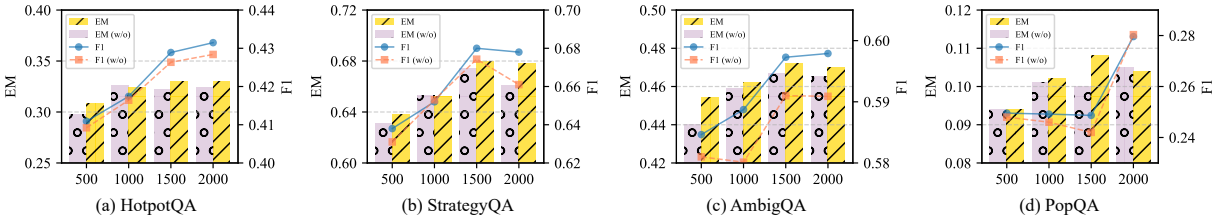
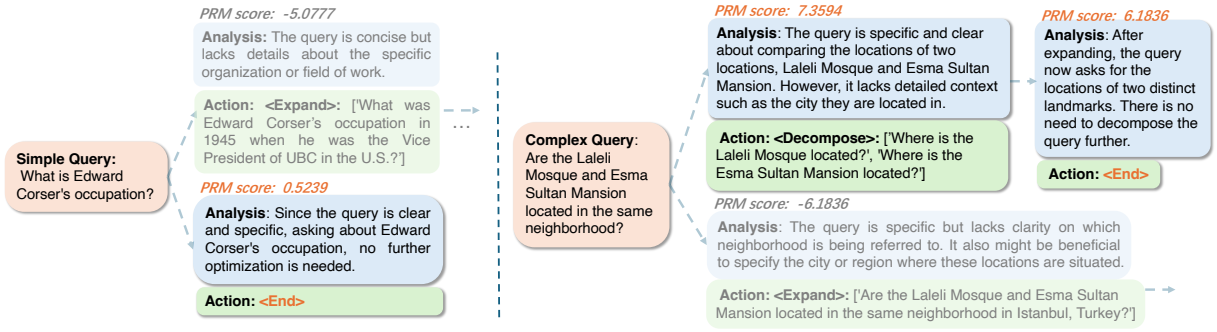


Figure 4: Rewriting performance of Q-PRM under varying MCTS simulation steps. We report EM and F1 scores with and without uncertainty guidance, denoted as EM, EM (w/o), and F1, F1 (w/o), respectively.



(a) PRM-guided simple query rewriting

(b) PRM-guided complex query rewriting

Figure 5: Case study of Q-PRM in rewriting simple and complex queries. The simple query is selected from the PopQA dataset, while the complex query is chosen from the HotpotQA dataset.

(Cheng et al., 2024b), and (d) Long-form QA using ASQA (Stelmakh et al., 2022). Following prior works (Jin et al., 2024; Cheng et al., 2024b; Stelmakh et al., 2022), we adopt F1 as the evaluation metric for MRQA and DomainRAG, and ROUGE-L for CORAL and ASQA. We select the best-performed baselines for comparison: Query2Doc, RaFe, and RRR. Notably, neither our method nor the baselines are further fine-tuned on these datasets.

As shown in Figure 6, our method consistently delivers strong performance across all four RAG tasks. While on MRQA (Figure 6 (a)) our results are comparable to RRR, Q-PRM outperforms all baselines in the remaining settings. These findings indicate that process-supervision-guided query rewriting enables robust generalization across diverse RAG tasks.

5.3.3 Impact of MCTS Simulation.

Figure 4 illustrates how Q-PRM's performance changes across different datasets as we increase the total number of MCTS simulation steps. Specifically, we vary the number of uncertainty-aware MCTS simulations from 500 to 2000, using these simulations to train both the query rewriter and the PRM. For comparison, we also include the performance of models trained without uncertainty guidance, highlighting the benefit of incorporating uncertainty-aware exploration.

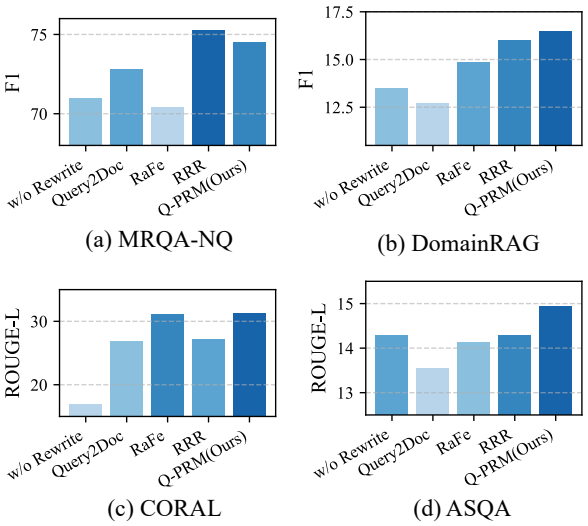


Figure 6: Comparison of Q-PRM and baseline methods on query rewriting performance across different RAG tasks. Evaluation scenarios across different QA settings: (a) Open-Domain: MRQA-NQ (Fisch et al., 2019), (b) Close-Domain: DomainRAG (Wang et al., 2024a), (c) Conversational: CORAL (Cheng et al., 2024b), (d) Long-form: ASQA (Stelmakh et al., 2022).

As shown in Figure 4, the performance of query rewriting improves as the number of MCTS simulation steps increases. This indicates that the trajectories generated by MCTS not only help the

rewriter learn how to perform multi-step rewrites, but also that the process supervision signals collected through MCTS enable the PRM to better adapt the rewriter to queries of varying complexity. In contrast, without considering uncertainty during MCTS, the efficiency of the search may be reduced due to excessive exploration of low-value nodes (Oren et al., 2024; Hayes et al., 2023).

5.3.4 Case Study on Process Supervision.

To demonstrate how PRM helps Q-PRM both generalize to queries of different complexity levels and improve the interpretability of query rewriting (Cheng et al., 2024a), we provide case studies on HotpotQA (complex queries) and PopQA (simpler queries).

As shown in Figure 5, when faced with a single-hop query (i.e., `What is Edward Corser's occupation?`), our method generates two paths. One path suggests expanding the query with background information about `Edward Corser`, while the other deems the query sufficiently clear and proceeds directly to retrieval. In this case, the PRM assigns a higher score (i.e., 0.5239) to the second path (i.e., direct retrieval), and thus, this path is ultimately selected. Similarly, when faced with a multi-hop query (i.e., `Are the Laleli Mosque and Esma Sultan Mansion located in the same neighborhood?`), Q-PRM first selects the path with the higher PRM score, which aims to decompose it into two sub-queries, and then terminates the rewriting process. From these examples, we can see that Q-PRM not only effectively adapts to queries of varying complexity but also addresses the issue of limited interpretability in the rewriting process (He et al., 2021; Kim and Lee, 2024).

6 Conclusion

In this paper, we identify a key challenge in query rewriting across varying levels of complexity: existing methods often overlook critical intermediate steps, leading to incorrect rewrites. We attribute this problem to the absence of process supervision signals. To bridge this gap, we emphasize the importance of process supervision signals and propose Q-PRM, a method that leverages a PRM to guide the rewriting process. Since such supervision signals are not readily available for this task, Q-PRM leverages uncertainty-aware MCTS to alleviate reward instability in the RAG process and to collect process signals for training the PRM. Offline experiments on four open-domain QA datasets and online evaluations show that Q-PRM effectively handles queries with diverse complexities.

Limitations

We acknowledge several limitations and opportunities for future work.

• Recent studies have proposed iterative RAG approaches, which perform multi-turn retrieval and rewriting during the inference stage to better handle complex tasks such as multi-hop question answering. In contrast, our work focuses on improving query rewriting in the pre-retrieval stage. These two directions are not mutually exclusive and could potentially be integrated to further enhance performance.

• Recently, the emergence of Large Reasoning Models (e.g., DeepSeek-R1 (Guo et al., 2025)) has shown promising capabilities in multi-step reasoning. These models often adopt end-to-end reinforcement learning methods (e.g., GRPO (Shao et al., 2024)) to directly enhance reasoning ability, rather than relying on an intermediate reward model. Although our approach mitigates reward sparsity and instability to some extent, future research may investigate the use of rule-based reward functions within RL frameworks to further enhance stability and effectiveness.

• Although multi-step reasoning at test time is increasingly adopted in both industrial systems (e.g., DeepSeek-R1 (Guo et al., 2025)) and academic works (e.g., iterative RAG), where moderate rewriting latency is acceptable due to improved response quality, introducing a PRM for multi-step query rewriting still incurs additional inference overhead. In future work, we aim to mitigate this overhead while preserving the benefits of multi-step reasoning.

Acknowledgements

This work was funded by the National Natural Science Foundation of China (62472426), fund for building world-class universities (disciplines) of Renmin University of China. Work partially done at Beijing Key Laboratory of Research on Large Models and Intelligent Governance, and Engineering Research Center of Next-Generation Intelligent Search and Recommendation, Ministry of Education.

References

Ingeol Baek, Jimin Lee, Joonho Yang, and Hwanhee Lee. 2024. Crafting the path: Robust query rewriting for information retrieval. *Preprint*, arXiv:2407.12529.

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

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

Zhaorun Chen, Zhuokai Zhao, Zhihong Zhu, Ruiqi Zhang, Xiang Li, Bhiksha Raj, and Huaxiu Yao. 2024. Autoprm: Automating procedural supervision for multi-step reasoning via controllable question decomposition. *arXiv preprint arXiv:2402.11452*.

Yiruo Cheng, Kelong Mao, and Zhicheng Dou. 2024a. Interpreting conversational dense retrieval by rewriting-enhanced inversion of session embedding. *arXiv preprint arXiv:2402.12774*.

Yiruo Cheng, Kelong Mao, Ziliang Zhao, Guanting Dong, Hongjin Qian, Yongkang Wu, Tetsuya Sakai, Ji-Rong Wen, and Zhicheng Dou. 2024b. Coral: Benchmarking multi-turn conversational retrieval-augmentation generation. *arXiv preprint arXiv:2410.23090*.

Youan Cong, Cheng Wang, Pritom Saha Akash, and Kevin Chen-Chuan Chang. 2024. Query optimization for parametric knowledge refinement in retrieval-augmented large language models. *arXiv preprint arXiv:2411.07820*.

Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. 2019. Mrqa 2019 shared task: Evaluating generalization in reading comprehension. *arXiv preprint arXiv:1910.09753*.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2022. Precise zero-shot dense retrieval without relevance labels. *arXiv preprint arXiv:2212.10496*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

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 (TACL)*.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. 2023. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*.

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

Conor F Hayes, Mathieu Reymond, Diederik M Roijers, Enda Howley, and Patrick Mannion. 2023. Monte carlo tree search algorithms for risk-aware and multi-objective reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 37(2):26.

Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Improving multi-hop knowledge base question answering by learning intermediate supervision signals. In *Proceedings of the 14th ACM international conference on web search and data mining*, pages 553–561.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55.

Rolf Jagerman, Honglei Zhuang, Zhen Qin, Xuanhui Wang, and Michael Bendersky. 2023. Query expansion by prompting large language models. *arXiv preprint arXiv:2305.03653*.

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. *arXiv preprint arXiv:2403.14403*.

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*.

Eugene Kharitonov, Craig Macdonald, Pavel Serdyukov, and Iadh Ounis. 2013. Intent models for contextualising and diversifying query suggestions. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 2303–2308.

Kiseung Kim and Jay-Yoon Lee. 2024. Re-rag: Improving open-domain qa performance and interpretability with relevance estimator in retrieval-augmented generation. *arXiv preprint arXiv:2406.05794*.

Qingyao Li, Wei Xia, Kounianhua Du, Xinyi Dai, Ruiming Tang, Yasheng Wang, Yong Yu, and Weinan Zhang. 2024a. Rethinkmcts: Refining erroneous thoughts in monte carlo tree search for code generation. *arXiv preprint arXiv:2409.09584*.

Xingxuan Li, Weiwen Xu, Ruochen Zhao, Fangkai Jiao, Shafiq Joty, and Lidong Bing. 2024b. Can we further elicit reasoning in llms? critic-guided planning with retrieval-augmentation for solving challenging tasks. *arXiv preprint arXiv:2410.01428*.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting for retrieval-augmented large language models. *arXiv preprint arXiv:2305.14283*.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Hannaneh Hajishirzi, and Daniel Khashabi. 2022. When not to trust language models: Investigating effectiveness and limitations of parametric and non-parametric memories. *arXiv preprint*.

Shengyu Mao, Yong Jiang, Boli Chen, Xiao Li, Peng Wang, Xinyu Wang, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. 2024. Rafe: ranking feedback improves query rewriting for rag. *arXiv preprint arXiv:2405.14431*.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. Ambigqa: Answering ambiguous open-domain questions. *arXiv preprint arXiv:2004.10645*.

Yaniv Oren, Villiam Vadocz, Matthijs T. J. Spaan, and Wendelin Böhmer. 2024. Epistemic monte carlo tree search. *Preprint*, arXiv:2210.13455.

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*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Preprint*, arXiv:2305.18290.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Mingyang Song, Zhaochen Su, Xiaoye Qu, Jiawei Zhou, and Yu Cheng. 2025. Prmbench: A fine-grained and challenging benchmark for process-level reward models. *arXiv preprint arXiv:2501.03124*.

Mingyang Song and Mao Zheng. 2024. A survey of query optimization in large language models. *arXiv preprint arXiv:2412.17558*.

Sofia Eleni Spatharioti, David M Rothschild, Daniel G Goldstein, and Jake M Hofman. 2023. Comparing traditional and llm-based search for consumer choice: A randomized experiment. *arXiv preprint arXiv:2307.03744*.

Ivan Stelmakh, Yi Luan, Bhuwan Dhingra, and Ming-Wei Chang. 2022. Asqa: Factoid questions meet long-form answers. *arXiv preprint arXiv:2204.06092*.

Zhongxiang Sun, Qipeng Wang, Weijie Yu, Xiaoxue Zang, Kai Zheng, Jun Xu, Xiao Zhang, Song Yang, and Han Li. 2025. Rearter: Retrieval-augmented reasoning with trustworthy process rewarding. *arXiv preprint arXiv:2501.07861*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *arXiv preprint arXiv:2212.10509*.

Shuting Wang, Jiongnan Liu, Shiren Song, Jiehan Cheng, Yuqi Fu, Peidong Guo, Kun Fang, Yutao Zhu, and Zhicheng Dou. 2024a. Domainrag: A chinese benchmark for evaluating domain-specific retrieval-augmented generation. *Preprint*, arXiv:2406.05654.

Xuezhi Wang and Denny Zhou. 2024. Chain-of-thought reasoning without prompting. In *Advances in Neural Information Processing Systems*, volume 37, pages 66383–66409. Curran Associates, Inc.

Yujing Wang, Hainan Zhang, Liang Pang, Hongwei Zheng, and Zhiming Zheng. 2024b. Maferw: Query rewriting with multi-aspect feedbacks for retrieval-augmented large language models. *arXiv preprint arXiv:2408.17072*.

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*.

Fanghua Ye, Meng Fang, Shenghui Li, and Emine Yilmaz. 2023. Enhancing conversational search: Large language model-aided informative query rewriting. *arXiv preprint arXiv:2310.09716*.

Hao Yu, Aoran Gan, Kai Zhang, Shiwei Tong, Qi Liu, and Zhaofeng Liu. 2024. Evaluation of retrieval-augmented generation: A survey. In *CCF Conference on Big Data*, pages 102–120. Springer.

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*.

Kepu Zhang, Zhongxiang Sun, Weijie Yu, Xiaoxue Zang, Kai Zheng, Yang Song, Han Li, and Jun Xu. 2025a. Qe-rag: A robust retrieval-augmented generation benchmark for query entry errors. *arXiv preprint arXiv:2504.04062*.

Kepu Zhang, Zhongxiang Sun, Xiao Zhang, Xiaoxue Zang, Kai Zheng, Yang Song, and Jun Xu. 2025b. Trigger3: Refining query correction via adaptive model selector. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 13260–13268.

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*, 1(2).

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. *Preprint*, arXiv:2403.13372.

Appendix

A Monte Carlo Tree Search

In existing literature (Zhang et al., 2024; Browne et al., 2012), MCTS builds a search tree T based on a policy model π_θ, which is usually the target LLM $\mathcal{M}_\theta$. Each node $s_t = [q_t, N(s_t), V(s_t)]$ represents a state comprising the optimization action a_t, optimized sub-queries q_t, the number of visits $N(s_t)$, and the value function (expected reward) $V(s_t)$ for downstream RAG task. The root node $s_0 = [q_0]$ only contains the original input query q_0, and each edge represents an action aimed at generating the next sub-query. During the search process, MCTS runs for multiple simulations. For the i-th simulation, it conducts four operations to expand the tree:

• *Selection*: selects the leaf node with the highest exploration potential, determined by the UCT (Upper Confidence Bounds applied to Trees) score. The UCT score is calculated as follows:

$$\mathrm{UCT}_i(s_t) = V_i(s_t) + w\sqrt{\frac{\ln N_i(p)}{N_i(s_t)}} \tag{7}$$

where w is a hyper-parameter that balances the exploitation (i.e., node value $N_i(s_t)$) and exploration (i.e., visit count $N_i(s_t)$), and p denotes the parent node of s_t.

• *Expansion*: explores multiple child nodes $\{s_{t+1}\}$ from the selected node s_t by repeatedly sampling actions based on the policy model π_θ. Note that the expansion prompt for query rewriting is shown in Table 6.

• *Evaluation*: aims to perform rollout for each expanded child node s_{t+1} until the task is solved and obtain a reward r based on the rollout results.

• *Backpropagation*: leverages the reward r of the child node to update the value $V_i(s_t)$ of nodes s_t along the path from the root node s_0 to the current node s_t:

$$\begin{aligned} N_{i+1}(s_t) &= N_i(s_t) + 1 \\ V_{i+1}(s_t) &= \frac{V_i(s_t)\, N_i(s_t) + r}{N_i(s_t)} \end{aligned} \tag{8}$$

where N_i and V_i are the number of visits and value function at the i-th iteration, respectively.

B Baseline Implementation Details

All model training is completed on a single machine with 2×NVIDIA A6000 GPUs. For heuristic query rewriting methods, we compare the following baselines:

• **LLM-Rewrite**: directly enable the LLMs to rewrite the original query with a few-shot prompt. The few-shot prompt is shown in Table 5.

• **Query2Doc** (Jagerman et al., 2023) expanded query by generating pseudo-documents through the few-shot prompting of LLMs. The prompt used in this paper is identical to the original work (Jagerman et al., 2023).

• **HyDE** (Gao et al., 2022) generated hypothetical documents for each query, and then encoded the query and documents to retrieve similar real documents.

• **Crafting** (Baek et al., 2024): implements fixed numbers of steps to conduct multi-step query rewriting using few-shot prompting. The input prompt for Crafting is shown in Appendix D.1, which is identical to the original paper.

For outcome-supervised training-based query rewriting methods, we compare the following baselines:

• **SFT**: uses the pre-generated high-quality rewrites to directly train the rewriter, following (Ma et al., 2023).

• **RaFe** (Mao et al., 2024): utilizes feedback of re-ranker to construct good-bad rewriting pairs to train the rewriter via DPO (Rafailov et al., 2023). The DPO fintuing is implemented via LLamaFactory (Zheng et al., 2024)

• **RRR** (Ma et al., 2023): conduct reinforcement learning (e.g., PPO (Schulman et al., 2017)) to train the rewriter using the quality of generated answers and answer hit rate in the retrieved documents as the reward. Following the original paper, we use an indicator to reward if the retrieved content hits the answer and penalize if it misses the answer, denoted as Hit. The total reward is a weighted sum of

EM, F1, and Hit. The DPO fintuing is implemented via LLamaFactory (Zheng et al., 2024)

C Human Evaluation Criteria

In our online experiment, the evaluation conducted by the experiment team broadly followed the guidelines outlined below:

Clarity and Completeness - **Referential Disambiguation**: Rewritten queries should resolve ambiguous pronouns or omitted subjects/objects using context from the current or previous dialogue. *Example*: User: "What are its use cases?" Previous Turn: "Can you tell me what a large language model is?" Rewritten: "What are the use cases of large language models?" Incorrect: "What are its use cases?" (retains ambiguity) - **Intent Completion**: Incomplete or underspecified queries should be supplemented to express a full and meaningful user intent. *Example*: User: "iPhone 13 Pro Max" Good Rewritten: "Can you introduce the iPhone 13 Pro Max?" Bad Rewritten: "iPhone 13 Pro Max?" (still incomplete)
Fluency and Factuality - **Fluency**: Rewritten or decomposed queries must be grammatically correct and natural in standard English. No awkward phrasing or unnatural syntax should remain. - **Factual Accuracy**: The rewritten query must not introduce hallucinated or incorrect facts. It should reflect only what is present or inferable from the input. *Example*: User: "What's the battery life of the iPhone 15?" Goold Rewritten: "How long does the battery of the iPhone 15 last?" Bad Rewritten: "Does the iPhone 15 have a 24-hour battery?" (adds unfounded claim)
Semantic Alignment - **Semantic Correctness**: The meaning of the rewritten query must be faithful to the original. Key information and focus must not shift. *Example*: User: "Is it bad to sleep late every night?" Good Rewritten: "Are there any health risks to sleeping late regularly?" Bad Rewritten: "What are the benefits of staying up late?" (reverses meaning) - **Intent Consistency**: The question type (e.g., "how", "why", "what") must align with the user's original intent. Misinterpreting intent leads to poor query decomposition. *Example*: User: "I get nervous and blush easily. It's distressing. What can I do?" Good Rewritten: "How to cope with frequent blushing?" Bad Rewritten: "Why does nervousness cause blushing?" (shifts from solution-seeking to explanation)
Coverage and Completeness - **Comprehensiveness**: For complex queries, especially those involving multi-step processes, the decomposition should result in a set of sub-queries that collectively cover all critical aspects. *Example*: User: "How do I apply for a credit card?" Good Rewritten: "What are the steps to apply for a credit card?", "What documents are needed to apply for a credit card?", "What are the eligibility criteria for credit card applications?" Bad Rewritten Incomplete: "What are the steps to apply for a credit card?" (misses documents and criteria).

Table 3: Human evaluation criteria for rewritten query.

D Prompt Details

D.1 Input Prompts for Baselines

By following the requirements, write 3 steps related to the Query and answer in the same format as the example.
Requirements: 1. In Step 1, generate the contextual background extracted from the existing query.
2. In Step 2, generate what information is needed to solve the question.
3. In Step 3, generate the expected answer based on the query, Step 1, and Step 2.
4. If you think there is no suitable answer, end with "None".
Example: Query 1: What is the number one Formula One car?
Step 1: Formula One (F1) is the highest class of international automobile racing competition held by the FIA.
Step 2: To know the best car, you have to look at the race records.
Step 3: Red Bull Racing's RB20 is the best car.

(4-shot examples) ...
Query: <query> Output:

Table 4: Input prompt for Crafting

D.2 Prompt for Few-Shot Method

Instruction: You are an expert at rewriting user query for the retrieval generative generation process.
Requirement Given a user query, your task is to give the rewritten query. Please refer to the query rewriting examples.
Example: Input Query: What are the best deep learning models?
Action: <Rewrite>
Output: ["What are the latest state-of-the-art deep learning models in 2024, particularly in NLP, computer vision, and recommendation systems?"]
Input Query: What profession do Nicholas Ray and Elia Kazan have in common?
Action: <Decompose>
Output: ["What was Nicholas Ray's profession?", "What was Elia Kazan's profession?"]
Input Query: How to improve LLM inference efficiency?
Action: <Expand>
Output: ["What are optimization techniques like quantization, distillation, and pruning?"]
Input Query: Who is the 2024 Summer Olympics table tennis singles champion?
Action: <Disambiguate>
Output: ["Who is the women's singles champion in table tennis at the 2024 Summer Olympics?", "Who is the men's singles champion in table tennis at the 2024 Summer Olympics?"]
Input Query: How many times has China hosted the Olympic Games?
Action: <Abstract>
Output: ["The history of hosting the Olympic Games."]

Input Query: <query>
Output: [...]

Table 5: Input prompt for Few-Shot

D.3 Prompt for MCTS Expansion

Instruction: You are an expert at optimizing user query for the retrieval generative generation process.
Given a user query and an existing partial solution (not a complete answer),
your task is to give the correct next query optimization step.
Your goal is to optimize the query, making it can retrieve relevant documents to answer the user's question.
You can only choose from the following actions:

<Decompose>, when the query is too complex or contains multiple sub-questions,
requires retrieving information from different sources or perspectives,
or involves reasoning that can be broken into simpler steps,
<Decompose> the query into separate sub-queries.
Example: Input Query: `'What profession does Nicholas Ray and Elia Kazan have in common?'`
<Decompose>: ["What was Nicholas Ray's profession?", "What was Elia Kazan's profession?"]
<Expand>, when the query is too short and may not return enough relevant documents, lacks key background details,
or could benefit from additional context or related concepts, <Expand> the query.
Example: Input Query: `'How to improve LLM inference efficiency?'`
<Expand>: ['What are optimization techniques like quantization, distillation, and pruning?']
<Disambiguate>, when the query is ambiguous or has multiple potential interpretations, [Disambiguate] the query.
Example: Input Query: `'Who is the 2024 Summer Olympics table tennis singles champion?'`
<Disambiguate>: ['Who is the womenś singles champion in table tennis at the 2024 Summer Olympics?',
'Who is the menś singles champion in table tennis at the 2024 Summer Olympics?']
<Abstract>, when the query requires not only an understanding of the facts but also the ability to comprehend
and apply domain-specific reasoning integral to the context of the data, <Abstract> the query.
Example: Input Query: `'How many times has China hosted the Olympic Games?'`
<Abstract>: ['The history of hosting the Olympic Games.']
Format:
Assuming the input is n-steps, then the format of the input is:
Query: ...
Existing Steps:
Step 1: Analysis: ...
Action: <...>: [...]
Step 2: Analysis: ...
Action: <...>: [...]
...
Step n: ...
where ... denotes omitted input information.
If no existing steps are provided, you need to briefly analyze the user query
and then output the first step to optimize it.
Otherwise, you need to output the next step (step $n+1$) based on the optimized query in the previous step (n).
Your output must be strictly in this format:
`Next step: Analysis: ... Action: <...>: [...]`
Here is the input, please follow the restricted output format.
`Query:`

Table 6: Prompt for MCTS Expansion