Optimizing RAG Rerankers with LLM Feedback via Reinforcement Learning

Yuhang Wu[1]* **Xiangqing Shen**[2]* **Fanfan Wang**[1] **Cangqi Zhou**[1]
Zhen Wu[3] **Xinyu Dai**[3] **Rui Xia**[2]†
[1]School of Computer Science and Engineering, Nanjing University of Science and Technology, China
[2]School of Intelligence Science and Technology, Nanjing University, China
[3]School of Artificial Intelligence, Nanjing University, China
{yhangwu, ffwang, cqzhou}@njust.edu.cn
{xqshen, wuz, daixinyu, rxia}@nju.edu.cn

Abstract

Rerankers play a pivotal role in refining retrieval results for Retrieval-Augmented Generation. However, current reranking models are typically optimized on static human annotated relevance labels in isolation, decoupled from the downstream generation process. This isolation leads to a fundamental misalignment: documents identified as topically relevant by information retrieval metrics often fail to provide the actual utility required by the LLM for precise answer generation. To bridge this gap, we introduce ReRanking Preference Optimization (RRPO)[1], a reinforcement learning framework that directly aligns reranking with the LLM's generation quality. By formulating reranking as a sequential decision-making process, RRPO optimizes for context utility using LLM feedback, thereby eliminating the need for expensive human annotations. To ensure training stability, we further introduce a reference-anchored deterministic baseline. Extensive experiments on knowledge-intensive benchmarks demonstrate that RRPO significantly outperforms strong baselines, including the powerful list-wise reranker RankZephyr. Further analysis highlights the versatility of our framework: it generalizes seamlessly to diverse readers (e.g., GPT-4o), integrates orthogonally with query expansion modules like Query2Doc, and remains robust even when trained with noisy supervisors.

1 Introduction

Retrieval-Augmented Generation (RAG) (Lewis et al., 2020; Borgeaud et al., 2022; Izacard and Grave, 2021; Gao et al., 2023) has emerged as a powerful paradigm to enhance LLMs by dynamically incorporating external information, thereby improving factual accuracy and contextual relevance for knowledge-intensive tasks. In a typical pipeline, the system first retrieves documents based on a user query and then passes them to an LLM Reader for response generation. Consequently, the overall performance depends heavily on the quality of these retrieved documents. To optimize this, a reranking model (reranker) is often introduced to refine the initial document set. By prioritizing the most relevant documents, the reranker ensures the LLM receives high-quality context, leading to more precise and coherent answers.

However, a fundamental disconnect persists in current RAG architectures: reranker optimization is typically isolated from the RAG pipeline. Most rerankers are optimized under a supervised paradigm using static Information Retrieval metrics (e.g., NDCG), a process that is entirely decoupled from the downstream LLM. As shown in Figure 1, this creates a critical misalignment: the notion of relevance learned by the reranker does not necessarily translate to the usefulness required by the LLM to generate precise answers (Lee et al., 2025). For instance, given the query "Who discovered penicillin?", a standard reranker might favor a long historical article (high topic relevance), whereas the LLM might benefit more from a concise timeline entry (high LLM usefulness). By treating the reranker as an independent module rather than an integrated component of the RAG pipeline, standard approaches may not explicitly optimize for the system's ultimate objective—generation quality.

To bridge this gap, we introduce ReRanking Preference Optimization (RRPO), a reinforcement learning framework that directly optimizes the reranker within the RAG pipeline using the LLM's generation quality as the reward signal. Unlike supervised approaches that rely on static, human-annotated relevance labels, RRPO leverages the downstream LLM itself to provide scalable supervision, thereby eliminating the need for expensive

*Equal contribution.
†Corresponding author.
[1]Our code is publicly available at https://github.com/NUSTM/RRPO

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 30474–30490

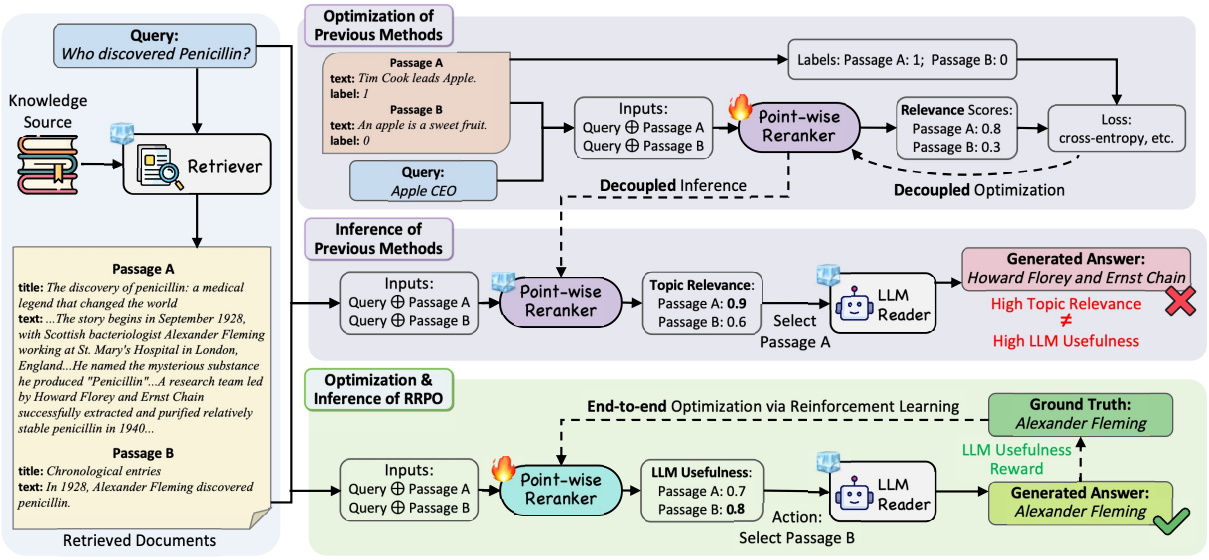

Figure 1: Comparison of previous methods with our proposed RRPO framework.

manual annotation. RRPO reframes reranking as a sequential decision-making process, allowing for end-to-end alignment between retrieval and generation. To address the instability caused by discrete, high-variance LLM rewards, we introduce a reference-anchored deterministic baseline. This component effectively stabilizes training by compelling the reranker to prioritize document "usefulness" over mere topic "relevance", avoiding the computational complexity and instability associated with training a separate critic network.

We validate the effectiveness of RRPO through extensive experiments on standard knowledge-intensive benchmarks. Empirical results demonstrate that RRPO achieves consistent improvements by explicitly optimizing the reranker to align with reader needs, significantly outperforming strong supervised baselines. Notably, our method surpasses state-of-the-art list-wise LLM rerankers, RankZephyr (Pradeep et al., 2023). This confirms that aligning the reranker with the reader's specific needs yields good context utility.

Beyond performance gains, our analysis highlights three critical properties that distinguish RRPO as a robust solution. First, it exhibits strong generalization: the learned policy not only handles complex multi-hop reasoning but also transfers seamlessly to diverse LLM Readers, effectively enhancing even proprietary models like GPT-4o. Second, it offers seamless integration: functioning as a "plug-and-play" module, it complements advanced retrieval strategies and delivers additive gains when combined with query expansion methods like Query2Doc (Wang et al., 2023). Finally, it demonstrates training robustness: RRPO maintains its effectiveness even when optimized using feedback from smaller, noisy supervisors, significantly reducing training costs.

2 Related Work

2.1 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) enhances LLMs by incorporating external information to mitigate hallucinations and knowledge obsolescence (Ji et al., 2023; Guu et al., 2020; Lewis et al., 2020). While early frameworks focused on dynamic knowledge provisioning for QA and dialogue (Gao et al., 2023; Karpukhin et al., 2020; Izacard et al., 2023; Shuster et al., 2021), subsequent research has refined the generation process. Key advancements include instruction-tuning generators to better utilize context (Ma et al., 2024; Zhu et al., 2024; Muennighoff et al., 2024; Liu et al., 2024; Lin et al., 2023) and incorporating self-reflection mechanisms like Self-RAG (Asai et al., 2024). Furthermore, sophisticated retrieval strategies such as iterative, adaptive, or active retrieval have been developed to handle complex queries (Trivedi et al., 2023; Jiang et al., 2023; Jeong et al., 2024; Xu et al., 2024).

2.2 Rerankers in RAG

Rerankers serve as a critical refinement stage between retrieval and generation. Early methods employed BERT-based encoders for pointwise or pairwise relevance estimation (Nogueira and Cho,

2019; Nogueira et al., 2019). Recent approaches explicitly leverage LLM capabilities, ranging from zero-shot listwise prompting (Sun et al., 2023) or discrete prompt optimization (Cho et al., 2023), to fine-tuning open-source LLMs on ranking data, as seen in RankZephyr (Pradeep et al., 2023; Ma et al., 2024). Other works have explored unsupervised risk minimization (Yuan et al., 2024) or utilized attention mechanisms for efficiency (Gangi Reddy et al., 2024; Chen et al., 2024). Most recently, research has expanded into context-aware and agentic paradigms. EBCAR (Yuan et al., 2025) utilizes embeddings to capture cross-passage interactions, while REARANK (Zhang et al., 2025a) formulates reranking as a reasoning agent.

2.3 Reinforcement Learning for RAG

Reinforcement Learning (RL) has been widely adopted to align RAG components with end-task performance. Prior work has applied RL to optimize query rewriting (Wang et al., 2024), incorporate reasoning traces via ReAct (Yao et al., 2023), or facilitate multi-step browsing (Nakano et al., 2021). A growing trend involves aligning retrieval with generation via preference optimization. Methods like DynamicRAG (Sun et al., 2025), KnowPO (Zhang et al., 2025b), and DPA-RAG (Dong et al., 2025) apply preference optimization (e.g., DPO) to fine-tune models, while others explore multi-agent reinforcement learning architectures (Chen et al., 2025). While recent progress has been driven by DPO-based alignments and multi-agent frameworks, RRPO explores an alternative paradigm by employing a sequential decision-making formulation to explicitly capture the combinatorial utility of document sets.

3 Methodology

We propose RRPO, a framework that fundamentally redefines document reranking as a sequential decision-making process. This approach is designed to bridge the misalignment between the relevance scores predicted by traditional rerankers and the actual usefulness of documents for the downstream LLM. Unlike supervised methods that rely on static labels, RRPO leverages Reinforcement Learning (RL) to optimize document selection based on direct feedback from the frozen LLM Reader. This allows for end-to-end alignment of the retrieval pipeline with the final generation quality. In this section, we formulate the reranking problem as a Markov Decision Process (MDP) and detail the components of our training algorithm, highlighting our reference-anchored baseline.

3.1 Task Formulation

Given a user query q and an initial candidate set of N documents $\mathcal{D} = \{d_1, \ldots, d_N\}$ retrieved from a corpus, the goal of the reranker is to select an ordered subset of k documents ($k < N$) that maximizes the quality of the response generated by a fixed LLM Reader.

We formalize this selection process as a finite-horizon Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, P, R)$:

State (s_t): At time step t, where $t \in \{1, \ldots, k\}$, the state s_t is defined as the set of candidate documents that have not yet been selected. The initial state $s_1 = \{d_1, d_2, \ldots, d_N\}$ is the set of top-N documents returned by the initial retrieval module.

Action (a_t): In state s_t, an action a_t denotes selecting one document d_{c_t} from the current set of documents s_t. Here, c_t is the index of this document in the original list of N documents. The action space $\mathcal{A}_t$ varies at each time step:

$$\mathcal{A}_t = \{\text{select } d_i \mid d_i \in s_t\}. \tag{1}$$

Transition Dynamics $(P(s_{t+1} \mid s_t, a_t))$: The state transitions are deterministic. Upon executing action a_t (i.e., selecting document d_{c_t}) in state s_t, the system transitions to the next state s_{t+1}, which is obtained by removing the selected document d_{c_t} from s_t :

$$s_{t+1} = s_t \setminus \{d_{c_t}\}. \tag{2}$$

Reward (r_t): After executing action a_t at each time step t, the agent receives a reward r_t. This reward reflects the contribution of the set of documents selected up to the current time step t, $\{d_{c_1}, \ldots, d_{c_t}\}$, to the generation of a high-quality response by the downstream LLM Reader. The downstream LLM Reader generates a specific reward value based on an evaluation metric R_{lm}.

3.2 ReRanking Preference Optimization

We detail our ReRanking Preference Optimization framework in three aspects: Agent and Policy Network, Optimization objective, and Reference-anchored Deterministic Baseline.

3.2.1 Agent and Policy Network

In this framework, the agent is implemented by a parameterized reranking model f_θ with parameters θ. The agent's behavior is defined by a policy

$\pi_\theta(a_t \mid s_t)$. We realize our policy π_θ through a pointwise reranking model f_θ. For a given query q and each document $d_i (i = 1, \ldots, N)$ in the initial candidate list, the reranker model f_θ computes a relevance score:

$$\text{score}_i = f_\theta(q, d_i), \tag{3}$$

where the scores for all N candidate documents are then converted into an initial probability distribution $\{p_1, \ldots, p_N\}$ using the SoftMax function, where p_i represents the initial probability that document d_i is considered relevant:

$$p_i = \frac{\exp(\text{score}_i)}{\sum_{j=1}^{N} \exp(\text{score}_j)}. \tag{4}$$

At each Reinforcement Learning time step t, when the agent is in state s_t (the set of remaining documents), the probability of selecting action a_t (i.e., choosing document $d_{c_t} \in s_t$), $\pi_\theta(a_t \mid s_t)$ is defined as the proportion of this document's initial probability p_{c_t} to the sum of initial probabilities of the currently remaining documents:

$$\begin{aligned} &\pi_\theta(a_t = \text{select}\, d_{c_t} \mid s_t) \\ &= \frac{p_{c_t}}{\sum_{d_j \in s_t} p_j} = \frac{p_{c_t}}{1 - \sum_{l=1}^{t-1} p_{c_l}}. \end{aligned} \tag{5}$$

Reward Calculation: The sequence of selected documents $\{d_{c_1}, \ldots, d_{c_t}\}$ is combined with the original query q and an instruction INST, and then input to the LLM Reader to generate a response:

$$\text{response}_t = \text{Reader}_{\text{LLM}}(\text{INST}, q, d_{c_1}, \ldots, d_{c_t}). \tag{6}$$

An evaluation metric R_{lm} is used to compute a score between the generated response response_t and the ground truth answer ans, which serves as the reward for that time step:

$$r_t = R_{lm}(\text{ans}, \text{response}_t). \tag{7}$$

The specific settings for the instruction INST and the evaluation metric R_{lm} are dataset-dependent and will be detailed in the Experiments section.

3.2.2 Objective Function

In practice, we found it necessary to constrain the magnitude of updates to the reranker model to prevent excessive oscillations in the training process that can result from large policy gradient updates. Therefore, drawing inspiration from the Proximal Policy Optimization (PPO) algorithm, we define our specific training objective as

$$\begin{aligned} J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \Bigg[\frac{1}{k} \sum_{t=1}^{k} \min \Big(& \rho_t(\theta) \hat{A}_t, \\ & \text{clip}\left(\rho_t(\theta), 1-\epsilon, 1+\epsilon\right) \hat{A}_t \Big) \\ & - \beta \mathbb{D}_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}}) \Bigg], \end{aligned} \tag{8}$$

where $\rho_t(\theta)$ is the importance sampling ratio $\rho_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{ref}(a_t|s_t)}$, π_{ref} is a fixed reference policy and $\hat{A}_t$ is the estimated advantage function at time step t, which will be detailed later.

Here, we employ both the PPO-clip mechanism and a KL divergence penalty to constrain the step size of each model update. We note that this combination of mechanisms have been extensively used and proven successful in RLHF (PPO-like algorithms) for language models (Ouyang et al., 2022; Rafailov et al., 2023; Shao et al., 2024), leading us to believe this setup is well-justified.

3.2.3 Reference-anchored Deterministic Baseline

In order to compute the advantage function $\hat{A}$ required by the PPO objective, we first introduce Generalized Advantage Estimation (GAE) (Schulman et al., 2016) as the framework for advantage calculation. A critical challenge in applying RL to RAG is the high variance and sparsity of LLM-based rewards, which makes training a separate value network (critic) notoriously difficult. To address this, we introduce a reference-anchored deterministic baseline for defining $V(s_t)$.

GAE in Objective Function We use GAE to compute the raw advantage function A_t, which is then normalized to obtain $\hat{A}_t$. GAE is a widely adopted method for effectively estimating the advantage function by balancing bias and variance, which is also used in PPO. The TD residual from time step t to $t+1$ is defined as

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t). \tag{9}$$

A_t is then calculated as (for an episode of length k):

$$A_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + (\gamma\lambda)^{k-t-1}\delta_{k-1} = \sum_{j=t}^{k-1}(\gamma\lambda)^{j-t}\delta_j, \tag{10}$$

where the parameter $\lambda \in [0, 1]$ controls the bias-variance trade-off in the advantage estimation. A value of $\lambda = 0$ would result in using only the single-step TD residual δ_t, which has low bias but high variance. Conversely, $\lambda = 1$ would correspond to a Monte Carlo estimate over the rest of the episode, which has high bias but lower variance. By using a value between 0 and 1, GAE provides a fine-grained control to reduce variance while keeping bias at an acceptable level, which is crucial for stable policy updates. We explain the value of λ in the Appendix B.

Limitations of Standard PPO Critics Our task exhibits certain characteristics that differ from typical RL modeling in RLHF. In our definition, the episode length k is relatively short, and the reward signal computed by R_{lm} can be highly discrete and exhibit large variance. We found in practice that training a separate critic network to fit $V(s_t)$ was ineffective. Critic models based on deep networks tend to predict numerically continuous values, making it difficult for them to accurately capture the value associated with LLM Reader's evaluations, especially for Factual QA tasks. This can lead to a decline in policy performance.

Reference Policy Rollout as a Deterministic Baseline Therefore, departing from traditional PPO and its classic applications in RLHF for LLMs, we adopt a deterministic baseline approach that uses actual observed values. This involves generating a reference trajectory using the reference model, which is then evaluated by the LLM Reader and R_{lm} in the environment. The value of being in state s_t (at step t of the agent's trajectory), denoted $V(s_t)$, is defined as the reward obtained by the reference policy π_{ref} executing a greedy sequence of t actions starting from the initial state s_1. Specifically, for each step $j = 1, \ldots, t$ in this reference rollout, a document is greedily selected according to π_{ref}, which is described as action a'_j:

$$a'_j = \underset{a \in \mathcal{A}(s'_j)}{\operatorname{argmax}}\, \pi_{ref}(a \mid s'_j), (\text{select document } d'_{c_j}) \tag{11}$$

where $s'_1, \ldots, s'_t$ is the path generated by these greedy selections by π_{ref}, with its initial state $s'_1 = s_1 = \{d_1, \ldots, d_N\}$. Let d'_{c_j} be the document selected by the reference model at its j-th step. The response is generated using these selected documents $\{d'_{c_1}, \ldots, d'_{c_t}\}$

$$\text{response}'_t = \text{Reader}_{\text{LLM}}(\text{INST}, q, d'_{c_1}, \ldots, d'_{c_t}). \tag{12}$$

The baseline value is then computed as

$$V(s_t) = R_{lm}(\text{ans}, \text{response}'_t). \tag{13}$$

Advantages of Reference-Anchored Optimization By anchoring the time-dependent baseline $V(s_t)$ to the deterministic greedy performance of the reference policy π_{ref}, we avoid the complexities of training a critic network and the heavy reliance on human preference data typically required in RLHF. This approach artfully integrates off-policy insights into PPO's on-policy framework, leveraging the typically strong reference reranker as a stable anchor for advantage estimation. By incorporating this reference-based value into the advantage calculation—alongside the importance sampling ratio and KL divergence penalty—we ensure that policy updates are robustly guided towards outperforming π_{ref} while mitigating the risk of significant performance degradation during exploration.

3.2.4 Training Stabilization Strategies

Even with the deterministic baseline, the raw advantage values A_t can exhibit significant variance. To further stabilize the policy gradient updates, we apply advantage normalization:

$$\hat{A}_t = \frac{A_t - \text{mean}(\mathbf{A}_{\text{window}})}{\text{std}(\mathbf{A}_{\text{window}}) + \epsilon_{\text{norm}}}, \tag{14}$$

where statistics are computed over a sliding window of recent episodes, and $\epsilon_{\text{norm}} = 10^{-8}$.

Additionally, for the KL penalty term in the objective function, we employ the estimator from (Ouyang et al., 2022):

$$\mathbb{D}_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}}) = \frac{\pi_{\text{ref}}(a_t \mid s_t)}{\pi_\theta(a_t \mid s_t)} - \log \frac{\pi_{\text{ref}}(a_t \mid s_t)}{\pi_\theta(a_t \mid s_t)} - 1. \tag{15}$$

We provide the pesudo codes of RRPO in Algorithm 1 of Appendix A.

4 Experiments

4.1 Experimental Settings

Datasets AmbigNQ (Min et al., 2020) is a disambiguated version of Natural Questions (NQ) (Kwiatkowski et al., 2019). AmbigNQ aims to elicit unique, definitive answers by providing more precise question formulations or contexts. We use its full training set and evaluate on validation set. HotpotQA (Yang et al., 2018) is a widely used multi-hop question answering (multi-hop QA) benchmark dataset, where questions typically require reasoning over information combined from multiple documents. We selected HotpotQA because multi-hop reasoning scenarios place higher demands on the collaborative capabilities of the various modules within RAG systems. We use full training set and evaluate on validation set.

Baselines To evaluate RRPO, we establish several key comparisons. Our primary baseline is the base reranker model gte-multilingual-reranker-base (Zhang et al., 2024) (reported as "gte reranker") without RRPO's reinforcement learning optimization, providing a direct ablation of our training paradigm's impact. We further benchmark against a diverse set of off-the-shelf alternatives. Specifically, we include **encoder-only** models like jina-reranker-v2-multilingual (Jina AI, 2024) and bge-reranker-base (Xiao et al., 2023), alongside the **decoder-only** model Qwen3-reranker-0.6B (Zhang et al., 2025c) (reported as "qwen3 reranker"). For broader context on RAG system performance, we also reference advanced RAG methods like FLARE (Jiang et al., 2023) and DRAGIN (Su et al., 2024) (with results cited from DeepRAG (Guan et al., 2025) for fair comparison). These methods typically employ strategies like confidence-based adaptive retrieval, representing orthogonal approaches to RAG optimization rather than direct reranker baselines.

Implementation Details We conduct all experiments using the preprocessed knowledge corpus provided by DPR (Karpukhin et al., 2020), where articles are segmented into 100-word passages. Our retrieval pipeline employs BM25 (Robertson and Zaragoza, 2009) to initially fetch 100 candidate documents, which are subsequently processed by the evaluated reranking models (e.g., gte, jina, and qwen3). From these reranked candidates, we select the top-k passages ($k = 3$ for HotpotQA and $k = 5$ for AmbigNQ) to serve as input for the Qwen2.5-7B (Team, 2024) LLM reader. Regarding the optimization objective, we follow previous work (Ma et al., 2023) by adopting a unified reward function $R_{lm} = \text{EM} + \lambda_f \text{F1} + \lambda_h \text{Hit}$ for both datasets, with hyperparameters set to $\lambda_f = 1.0$ and $\lambda_h = 1.0$. The Hit metric is defined to be 1 if the ground-truth answer appears in the generated response and -1 otherwise. Further training details including fair comparison discussions are provided in Appendix B.

Methods	HotpotQA		AmbigNQ	
	EM	F1	EM	F1
LLM Reader only	17.80	26.85	3.55	17.80
LLM + CoT	21.34	31.17	10.69	27.85
FLARE	23.40*	32.06*	25.62	37.78
DRAGIN	16.70*	24.60*	18.33	30.09
Naive RAG (BM25)	24.23	35.42	20.33	32.18
+ qwen3 reranker	17.03	25.71	31.17	42.87
+ bge reranker	28.06	40.46	31.37	45.01
+ jina reranker	28.09	40.65	32.27	45.53
+ gte reranker	28.24	41.23	35.51	48.55
+ qwen3 reranker (RRPO)	18.43	27.73	32.52	45.17
+ bge reranker (RRPO)	28.12	40.76	32.37	45.87
+ jina reranker (RRPO)	29.26	42.16	32.86	46.20
+ gte reranker (RRPO)	**30.03**	**43.22**	**36.56**	**49.67**

Table 1: Main experimental results on the HotpotQA and AmbigNQ datasets. * adapts from previous work DeepRAG (Guan et al., 2025).

4.2 Main Results

The results in Table 1 demonstrate that while standard pre-trained rerankers significantly improve upon Naive RAG, applying the RRPO paradigm yields decisive further gains. Specifically, the RRPO-optimized gte reranker boosts F1 scores by 1.99 on HotpotQA and 1.12 on AmbigNQ, with similar consistency observed across other architectures like jina. Crucially, our results demonstrate that RRPO is effective across diverse model architectures, yielding consistent gains for both **encoder-only** and **decoder-only** rerankers. Statistical significance tests, detailed in Appendix H, confirm that these improvements are statistically significant. These results not only verify that our RL-based optimization successfully aligns retrieval with downstream generation needs but also establish RRPO as highly competitive against advanced RAG strategies like FLARE and DRAGIN.

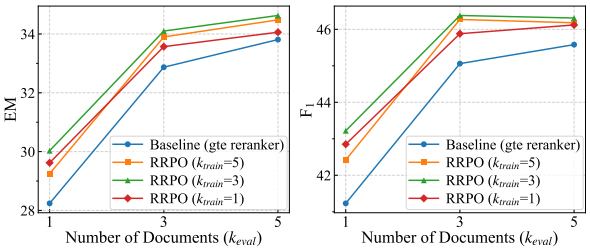

Figure 2: Ablation Experiments on HotpotQA.

4.3 Ablation Study

Figure 2 investigates the impact of training interaction depth ($k_{train} \in \{1, 3, 5\}$) on HotpotQA. All RRPO variants consistently outperform the non-RL baseline, demonstrating robustness. Notably, $k_{train} = 3$ yields optimal results, balancing sufficient context against the noise and attention diffusion of larger sets ($k_{train} = 5$) and the limited scope of single-document training ($k_{train} = 1$).

4.4 Extensive Analysis on Generalizability and Robustness

In this section, we provide a comprehensive analysis demonstrating that RRPO learns a robust, transferable, and highly efficient document utility policy. Going beyond standard benchmarks, our experiments validate five key properties of the proposed framework: (1) it generalizes effectively to structurally complex multi-hop reasoning scenarios; (2) it transfers seamlessly across diverse LLM readers (including closed-source models) without fine-tuning; (3) it outperforms state-of-the-art list-wise rerankers by explicitly optimizing for reader utility; (4) it yields orthogonal gains when integrated into advanced RAG pipelines; and (5) it maintains high label efficiency even when distilled from smaller supervisors.

RRPO demonstrates robust generalization to structurally complex multi-hop datasets. Instead of limiting our evaluation to simple pattern matching, we extend the scope to 2WikiMultiHopQA (Ho et al., 2020) and MusiQue (Trivedi et al., 2022), datasets that necessitate rigorous information integration across disconnected documents. Using the identical training configuration as the HotpotQA experiments, Table 2 reports the performance in the top-5 context setting (with comprehensive results in Appendix E). RRPO consistently outperforms the strong gte reranker baseline across these benchmarks. Crucially, this performance gain on complex multi-hop datasets confirms that our method does not merely overfit to specific query patterns.

Methods	EM	F1
Naive RAG (BM25)	26.11	30.40
+ gte reranker	26.87	31.42
+ gte reranker (RRPO)	**27.68**	**32.48**

Table 2: Performance of RRPO on 2WikiMultiHopQA.

Methods	HotpotQA		AmbigNQ	
	EM	F1	EM	F1
BM25 + gte reranker	32.87	45.06	40.26	51.36
BM25 + RankZephyr	32.40	44.49	40.06	51.58
BM25 + gte reranker (RRPO)	**34.10**	**46.38**	**41.11**	**52.18**

Table 3: Comparison of RRPO and the list-wise LLM reranker RankZephyr.

Instead, it successfully learns to identify the cohesive chain of evidence required for multi-hop reasoning, validating its semantic robustness beyond the training domain.

The learned ranking policy proves highly transferable across diverse LLM architectures and scales. We investigate whether RRPO captures a universal criterion for document utility regardless of the downstream reader. As illustrated in Figure 3, our method demonstrates strong "plug-and-play" potential across a wide spectrum of models without any reader-specific fine-tuning. First, the gains consistently transfer across model scales, benefiting both smaller (3B) and larger (14B) variants of the Qwen family. More importantly, the policy proves effective across differing architectures, extending benefits to Llama-3.1 and GLM-4. This cross-architecture transferability suggests that RRPO relies on intrinsic semantic information density rather than model-specific parametric knowledge. Finally, the method acts as a genuinely reader-agnostic filter, enhancing even proprietary, closed-source models such as Gemini-2.5-Flash, Claude-3.5-Sonnet, and GPT-4o.

RRPO outperforms state-of-the-art list-wise rerankers by explicitly optimizing for reader utility. To position our method within the current landscape, we compare it against RankZephyr (Pradeep et al., 2023), a representative 7B list-wise LLM reranker known for its GPT-4-level performance. As shown in Table 3, RRPO achieves superior performance compared to RankZephyr in the top-3 context setting across both HotpotQA and AmbigNQ. While list-wise approaches leverage the general reasoning capabilities of LLMs to sort documents, our results indicate

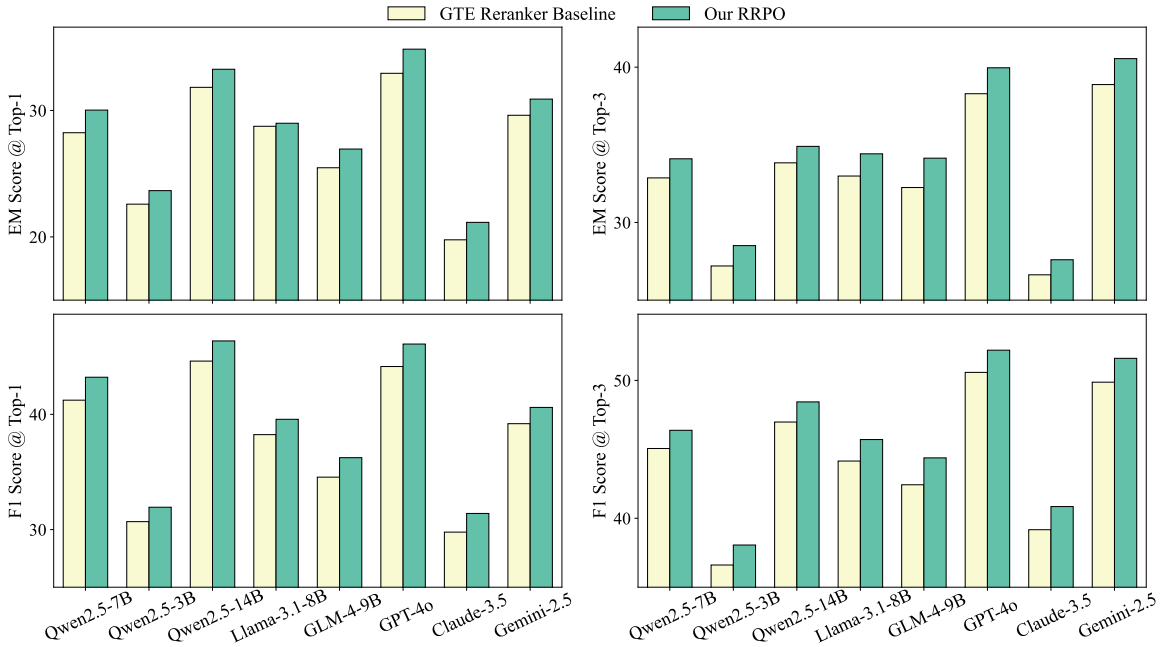

Figure 3: Generalization of RRPO-trained reranker to Various LLM Readers on HotpotQA.

that they may not be optimally aligned with the specific needs of a reader in RAG. In contrast, RRPO's policy—distilled through RRPO —is explicitly optimized to maximize reader performance, allowing it to filter for utility more effectively than general-purpose ranking prompts.

RRPO provides orthogonal improvements to query expansion techniques within advanced RAG frameworks. Retrieval pipelines often employ query expansion (e.g., Query2Doc (Wang et al., 2023)) to address low recall. We explore whether our method contributes value beyond merely compensating for weak initial retrieval. Table 4 reveals a strictly tiered improvement structure. Even atop the strong baseline established by Query2Doc, RRPO delivers significant additional gains. This confirms that the two components play orthogonal roles: while Query2Doc expands the search boundary to enhance Recall, RRPO optimizes the context utility to enhance Precision. Consequently, our method proves essential even when initial retrieval quality is already high, serving as a critical refinement stage in sophisticated RAG pipelines.

The framework maintains high label efficiency even when distilled from smaller, lightweight supervisors. Finally, we examine the cost-efficiency of our approach by training a reranker using a smaller, potentially "noisier" supervisor (Qwen2.5-3B), denoted as RRPO-3B. Table 5 reports its performance (with full details in Appendix F). Remarkably, RRPO-3B delivers consistent improvements. This resilience to supervisor noise suggests that the core signals for document utility are prominent enough to be captured even by lightweight models. This finding highlights the label efficiency of our approach, demonstrating that a robust reranking policy can be distilled without relying on computationally expensive, large-scale supervisors.

Methods	HotpotQA		AmbigNQ	
	EM	F1	EM	F1
Naive RAG (BM25)	24.23	35.42	20.33	32.18
+ Query2Doc	28.79	41.78	36.06	50.85
+ gte reranker	28.87	42.06	38.76	53.14
+ gte reranker (RRPO)	**30.51**	**44.10**	**39.81**	**54.10**

Table 4: Performance of RRPO within the Query2Doc-enhanced pipeline on HotpotQA and AmbigNQ.

LLM Reader	Methods	EM	F1
Qwen2.5-3B	+ GTE (Baseline)	22.59	30.69
	+ GTE (RRPO-3B)	**23.36**	**31.56**

Table 5: Performance of RRPO trained with a smaller LLM Reader (Qwen2.5-3B).

5 Conclusion

In this work, we propose RRPO to address the fundamental misalignment between static retrieval metrics and the dynamic needs of LLM readers

in RAG systems. By formulating reranking as a sequential decision-making process anchored by a deterministic baseline, we successfully stabilize the reinforcement learning training, eliminating the dependency on expensive human annotations or unstable critic networks. Our extensive experimental results offer a critical insight: optimizing for context utility yields significantly better RAG performance than merely optimizing for semantic relevance. Beyond performance gains, RRPO demonstrates remarkable robustness—it generalizes across diverse LLM readers (including proprietary models like GPT-4o) and distills effective policies even from smaller, noisy supervisors. This establishes RRPO not just as a performance booster, but as a scalable, cost-efficient paradigm for constructing task-aware RAG systems.

Limitations

As a reranking model, RRPO operates on a candidate set of documents provided by an initial retrieval module. If the initial retriever fails to retrieve a sufficient number of relevant documents (i.e., low recall in the top-N candidates), the ability of RRPO to improve the final RAG performance will be inherently limited, regardless of how well it reorders the provided candidates.

Ethics Statement

This work does not pose any ethical issues. All the data and models used in this paper are publicly available and are used under following licenses: Creative Commons BY 4.0 License, MIT License, Apache license 2.0.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grant No. 62476134 and No. 62376120.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, and 1 others. 2022. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR.

Shijie Chen, Bernal Jiménez Gutiérrez, and Yu Su. 2024. Attention in large language models yields efficient zero-shot re-rankers. *arXiv preprint arXiv:2410.02642*.

Yiqun Chen, Lingyong Yan, Weiwei Sun, Xinyu Ma, Yi Zhang, Shuaiqiang Wang, Dawei Yin, Yiming Yang, and Jiaxin Mao. 2025. Improving Retrieval-Augmented Generation through Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 38.

Sukmin Cho, Soyeong Jeong, Jeong yeon Seo, and Jong C Park. 2023. Discrete prompt optimization via constrained generation for zero-shot re-ranker. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 960–971.

Guanting Dong, Yutao Zhu, Chenghao Zhang, Zechen Wang, Ji-Rong Wen, and Zhicheng Dou. 2025. Understand what llm needs: Dual preference alignment for retrieval-augmented generation. In *Proceedings of the ACM on Web Conference 2025*, pages 4206–4225.

Revanth Gangi Reddy, JaeHyeok Doo, Yifei Xu, Md Arafat Sultan, Deevya Swain, Avirup Sil, and Heng Ji. 2024. FIRST: Faster improved listwise reranking with single token decoding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8642–8652, Miami, Florida, USA. Association for Computational Linguistics.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2:1.

Xinyan Guan, Jiali Zeng, Fandong Meng, Chunlei Xin, Yaojie Lu, Hongyu Lin, Xianpei Han, Le Sun, and Jie Zhou. 2025. Deeprag: Thinking to retrieval step by step for large language models. *arXiv preprint arXiv:2502.01142*.

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

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th*

Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, pages 874–880, Online. Association for Computational Linguistics.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251):1–43.

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)*, pages 7036–7050, Mexico City, Mexico. Association for Computational Linguistics.

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

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

Jina AI. 2024. jina-reranker-v2-base-multilingual. `https://jina.ai/models/jina-reranker-v2-base-multilingual`. Accessed: 2024-06-25.

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

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, and 1 others. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Dahyun Lee, Yongrae Jo, Haeju Park, and Moontae Lee. 2025. Shifting from ranking to set selection for retrieval augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 17606–17619, Vienna, Austria. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Richard James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, and 1 others. 2023. Ra-dit: Retrieval-augmented dual instruction tuning. In *The Twelfth International Conference on Learning Representations*.

Zihan Liu, Wei Ping, Rajarshi Roy, Peng Xu, Chankyu Lee, Mohammad Shoeybi, and Bryan Catanzaro. 2024. Chatqa: Surpassing gpt-4 on conversational qa and rag. In *Advances in Neural Information Processing Systems (NeurIPS)*.

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

Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2024. Fine-tuning llama for multi-stage text retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2421–2425.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. AmbigQA: Answering ambiguous open-domain questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5783–5797, Online. Association for Computational Linguistics.

Niklas Muennighoff, SU Hongjin, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. 2024. Generative representational instruction tuning. In *ICLR 2024 Workshop: How Far Are We From AGI*.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, and 1 others. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*.

Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-stage document ranking with bert. *arXiv preprint arXiv:1910.14424*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023. Rankzephyr: Effective and robust zero-shot listwise reranking is a breeze! *arXiv preprint arXiv:2312.02724*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2016. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3784–3803, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Weihang Su, Yichen Tang, Qingyao Ai, Zhijing Wu, and Yiqun Liu. 2024. DRAGIN: Dynamic retrieval augmented generation based on the real-time information needs of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12991–13013, Bangkok, Thailand. Association for Computational Linguistics.

Jiashuo Sun, Xianrui Zhong, Sizhe Zhou, and Jiawei Han. 2025. Dynamicrag: Leveraging outputs of large language model as feedback for dynamic reranking in retrieval-augmented generation. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is ChatGPT good at search? investigating large language models as re-ranking agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14918–14937, Singapore. Association for Computational Linguistics.

Qwen Team. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multi-hop questions via single-hop question composition. *Trans. Assoc. Comput. Linguistics*, 10:539–554.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, Toronto, Canada. Association for Computational Linguistics.

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

Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric P Xing, and Zhiting Hu. 2024. Promptagent: Strategic planning with language models enables expert-level prompt optimization. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-pack: Packaged resources to advance general chinese embedding. *Preprint*, arXiv:2309.07597.

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*.

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

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)*.

Xiaowei Yuan, Zhao Yang, Yequan Wang, Jun Zhao, and Kang Liu. 2024. Improving zero-shot llm re-ranker with risk minimization. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17967–17983.

Ye Yuan, Mohammad Amin Shabani, and Siqi Liu. 2025. Embedding-based context-aware reranker. *arXiv preprint arXiv:2510.13329*.

Le Zhang, Bo Wang, Xipeng Qiu, Siva Reddy, and Aishwarya Agrawal. 2025a. Rearank: Reasoning re-ranking agent via reinforcement learning. *arXiv preprint arXiv:2505.20046*.

Ruizhe Zhang, Yongxin Xu, Yuzhen Xiao, Runchuan Zhu, Xinke Jiang, Xu Chu, Junfeng Zhao, and Yasha Wang. 2025b. Knowpo: Knowledge-aware preference optimization for controllable knowledge selection in retrieval-augmented language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 25895–25903.

Xin Zhang, Yanzhao Zhang, Dingkun Long, Wen Xie, Ziqi Dai, Jialong Tang, Huan Lin, Baosong Yang, Pengjun Xie, Fei Huang, and 1 others. 2024. mgte: Generalized long-context text representation and reranking models for multilingual text retrieval. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1393–1412.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025c. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

Yutao Zhu, Peitian Zhang, Chenghao Zhang, Yifei Chen, Binyu Xie, Zheng Liu, Ji-Rong Wen, and Zhicheng Dou. 2024. INTERS: Unlocking the power of large language models in search with instruction tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2782–2809, Bangkok, Thailand. Association for Computational Linguistics.

A Pseudo Codes of RRPO

Here we present the pseudo codes for our RRPO algorithm.

Algorithm 1: RRPO: RL for Reranker Optimization

Input: Initial retriever, $\text{Reader}_{\text{LLM}}$, ground truth answers 'ans', instruction 'INST', eval metric R_{lm}.
Hyperparameters: k (docs to select), α (learning rate), PPO params (ϵ, β), GAE params (γ, λ), M_{iter} (iterations), E_{ppo} (PPO epochs), B_{size} (batch size).
Initialize policy network (reranker) f_θ; Fixed reference policy π_{ref} (from f_{ref}).
Initialize experience buffer $\mathcal{D}$.
for iteration $m = 1$ to M_{iter} **do**
    Clear buffer $\mathcal{D}$.
    for each training query q in a batch of B_{size} **do**
        $D_{cand} \leftarrow \text{Retriever}(q)$; $s_1 \leftarrow D_{cand}$; $L_{selected} \leftarrow []$.
        $\tau_{query\_data} \leftarrow []$. ▷ Temporary storage for the current query's trajectory
        1. Agent Rollout & Reward Calculation
        for $t = 1$ to k **do**
            Select document d_{c_t} from s_t using policy $\pi_\theta(\cdot|s_t; q)$ (derived from f_θ, Eq. (3)).
            $L_{selected} \leftarrow L_{selected} \cup \{d_{c_t}\}$; $s_{t+1} \leftarrow s_t \setminus \{d_{c_t}\}$.
            $response_t \leftarrow \text{Reader}_{\text{LLM}}(\text{INST}, q, L_{selected})$.
            $r_t \leftarrow R_{lm}(\text{ans}_q, \text{response}_t)$ (Eq. (7)).
            Add $(s_t, d_{c_t}, r_t, \log \pi_\theta(d_{c_t}|s_t))$ to τ_{query_data}.
        2. Learning-free Deterministic Baseline Calculation
        for $t = 1$ to k **do** ▷ For each step t in the agent's trajectory
            $L'_{sel\_for\_V_t} \leftarrow$ docs selected by t-step greedy rollout of π_{ref} on D_{cand} for q (Eq. (11)).
            $V_t \leftarrow R_{lm}(\text{ans}_q, \text{Reader}_{\text{LLM}}(\text{INST}, q, L'_{sel\_for\_V_t}))$ (This is $V(s_t)$ in Eq. (13)).
            Augment t-th step data in τ_{query_data} with its corresponding V_t.
        3. Advantage Estimation (GAE)
        Compute GAE advantages $A_1, \ldots, A_k$ for τ_{query_data} using $r_t, V_t, \gamma, \lambda$ (Eqs. (9)-(10)
        Store trajectory (now including A_t) from τ_{query_data} into $\mathcal{D}$.
        ▷ End of batch collection
    4. Advantage Normalization
    Normalize all advantages A_t in $\mathcal{D}$ to get $\hat{A}_t$ (Eq. (14), e.g., using mean/std over $\mathcal{D}$).
    5. Policy Optimization (PPO)
    Update policy parameters θ for E_{ppo} epochs by optimizing PPO objective (Eq. (8)) using mini-batches from $\mathcal{D}$ (which contains $s_t, d_{c_t}, \hat{A}_t, \log \pi_\theta(d_{c_t}|s_t)_{old}$, and uses π_{ref}).
Output: Optimized reranker f_θ.

B More Implementation Details

B.1 Training Details

On the HotpotQA dataset, we trained for 2 epochs with an initial learning rate of $2e-6$. For RL training interactions on this dataset, we used $k_{train} = 3$ (i.e., the LLM Reader received 3 documents per interaction). On the AmbigNQ dataset, we trained for 4 epochs with an initial learning rate of $1.6e-6$. For RL training interactions on this dataset, we used $k_{train} = 5$. We adjusted some PPO hyperparameters based on the characteristics of the RRPO task. Specifically, in GAE, the discount factor $\gamma = 0.99$ and $\lambda = 0.95$. For the PPO clipping objective, $\epsilon = 0.2$. In our objective function, the KL divergence penalty coefficient β was set to 0.1. We use AdamW as our Optimizer. We use 4 * RTX 3090 for training reranker, and 2 * A6000 for deploying LLM Reader using vllm due to accelerate LLM's inference performance. We use torch AMP and Flash Attention for training. We set batch size to 32 using gradient accumulation.

B.2 Discussion on Experimental Fairness and Baseline Selection

To ensure the rigor of our experimental conclusions, we strictly control the variables and carefully select baselines based on data availability.

Controlled Evaluation Protocol: As emphasized in the main text, strict variable control is maintained across all experiments. For every query, both the baseline (e.g., gte-reranker) and our optimized model (RRPO) rank the exact same pool of candidate documents (top-N) retrieved by BM25. All downstream parameters, including the number of selected documents (k) and the LLM Reader configuration, are kept identical. This ensures that any performance gain is exclusively attributable to

the reordering policy learned by RRPO.

Why SFT is Not Included as a Baseline: A direct comparison with Supervised Fine-Tuning (SFT) is methodologically challenging in our specific open-domain setting for two reasons:

Absence of Gold Labels: SFT relies on high-quality, passage-level relevance annotations (e.g., query-passage pairs labeled as positive/negative). However, the datasets used in our experiments (HotpotQA and AmbigNQ within the DPR retrieval setup) provide ground-truth answers but do not inherently provide gold-standard ranking labels for the retrieved top-100 candidates.

Domain Mismatch Risks: While one could fine-tune a reranker on external labeled corpora like MS MARCO, evaluating such a model on our target datasets would introduce severe domain distribution shifts, rendering the comparison unfair.

Therefore, comparing against the pre-trained base reranker provides the most direct assessment of RRPO's ability to align retrieval with generation using only the available reader feedback, effectively bypassing the data bottleneck of SFT.

C Prompt Template

We utilize this prompt template for training and evaluation on both HotpotQA and AmbigNQ dataset.

System:
You are a highly capable assistant who can read the provided passages and accurately answer the question below. You only need to answer the information required for the question within a few words, without any additional content. For questions requiring a 'yes' or 'no' response, answer solely with 'Yes' or 'No', without providing any additional explanation.

User:
passage 1: Steven Peter's father is Steven Pawl.

User:
passage 2: Steven Peter's father graduated from Harvard University.

User:
Based on these passages, answer the question. question: Where did Steven Pawl graduate from?

Assistant:
Harvard University

User:
passage 1: {{passage_1}}

User:
passage 2: {{passage_2}}

User:
passage 3: {{passage_3}}

...(the number of passages is determined by k.)

User:
Based on these passages, answer the question. question: {{query}}

D Generalization with Different Prompt Templates

Previous sections demonstrated the generalization of our method across different LLM Readers. In this subsection, we investigate a critical question: does the reranker, trained via reinforcement learning with a specific prompt template, become overfitted to that template? To assess robustness, we evaluate performance using two additional templates (Template A and Template B) that differ structurally and stylistically from the default template used in the main experiments.

Template A redefines the system role as an "information extraction AI" and groups context under "Text Snippet" headers within a single user turn. Template B adopts an "Objective-Constraints" format in the system message and positions the question before the "Supporting Documents." Both templates modify the phrasing for conciseness while strictly enforcing Yes/No constraints to maintain consistency with the Qwen2.5-7B model used as the LLM Reader.

The specific structures of Prompt Template A and B are shown below.

System:
You are an information extraction AI. Your task is to answer the question using the provided text within a few words. Be concise. For yes/no questions, respond with 'Yes' or 'No' and nothing more.

User:

Here is the text you should use:

Text Snippet 1: {{passage_1}}
—
Text Snippet 2: {{passage_2}}
—
...(passages continue based on k)
—
Based on the text snippets above, what is the answer to this question: {{query}}

System:
Objective: Answer the user's question within a few words. Constraints: Use the information given in the 'Supporting Documents'. For 'Yes' or 'No' questions, provide only that word as the answer. Be brief.

User:
Question: {{query}}

Supporting Documents:
Document [1]: {{passage_1}}
Document [2]: {{passage_2}}
...(passages continue based on k)
Please provide the answer found in the documents.

The experimental results are presented in Table 6, with default template results reproduced from Table 1 for comparison.

Template	Methods	HotpotQA	
		EM	F1
Default	BM25	24.23	35.42
	+ GTE	28.24	41.23
	+ GTE (RRPO)	**30.03**	**43.22**
A	BM25	23.08	30.80
	+ GTE	28.39	38.15
	+ GTE (RRPO)	**29.89**	**39.98**
B	BM25	24.17	32.17
	+ GTE	29.40	39.37
	+ GTE (RRPO)	**31.07**	**41.32**

Table 6: Generalization results on different prompt templates on HotpotQA.

As shown in Table 6, RRPO demonstrates strong robustness to variations in prompt templates, consistently outperforming both Naive RAG (BM25) and the standard gte reranker across all formats. Notably, while the baseline performance fluctuates due to the LLM's sensitivity to prompt phrasing (e.g., Naive RAG EM scores range from 23.08 to 24.23), RRPO delivers a stable and significant performance boost in every scenario.

This consistency confirms that the reranker has learned to identify intrinsically valuable documents rather than overfitting to the specific linguistic patterns of the training prompt. Such generalization capability is critical for real-world RAG systems, validating RRPO as a plug-and-play component compatible with diverse user-defined prompt structures.

Methods	top-1		top-3	
	EM	F1	EM	F1
Naive RAG (BM25)	23.01	28.47	25.64	30.20
+ gte reranker	22.87	28.99	**26.46**	31.36
+ gte reranker (RRPO)	**23.06**	**29.50**	26.36	**31.47**

Table 7: Performance of RRPO on 2WikiMultiHopQA.

Methods	top-1		top-3	
	EM	F1	EM	F1
BM25 + gte reranker	6.00	16.33	10.22	19.22
BM25 + gte reranker (RRPO)	**6.21**	**16.35**	**10.34**	**19.84**

Table 8: Performance of RRPO on Harder MuSiQue.

E Experimental Results on More Challenging Multi-hop QA Datasets

More experimental results on 2WikiMultiHopQA are reported in Table 7. The results further demonstrate the effectiveness of RRPO on this dataset.

For MuSiQue dataset, due to the task's complexity and the lower recall of the BM25 retriever, the overall performance baseline is not high.

Despite the limited gains, RRPO still yields positive improvements. This demonstrates that even in challenging scenarios with low-quality initial retrieval, RRPO's alignment with the LLM allows it to filter more valuable information from noisy candidates. This shows the advantage of our single-step optimization even within a complex multi-hop context.

F Experimental Results on Smaller LLM Readers

In our main experiments, we choose Qwen2.5-7B as the LLM Reader in RAG pipeline. We now explore whether the smaller LLM Readers can affect the results due to its more noisy reward feedback.

LLM Reader	Methods	HotpotQA	
		EM	F1
Qwen2.5-3B	+ GTE	22.59	30.69
	+ GTE (RRPO-3B)	23.36	31.56
	+ GTE (RRPO-7B)	**23.66**	**31.94**
Qwen2.5-7B	+ GTE	28.24	31.94
	+ GTE (RRPO-3B)	29.47	42.65
	+ GTE (RRPO-7B)	**30.03**	**43.22**
Qwen2.5-14B	+ GTE	31.83	44.62
	+ GTE (RRPO-3B)	33.09	45.99
	+ GTE (RRPO-7B)	**33.26**	**46.37**

Table 9: Performance of RRPO with smaller LLM Readers.

Methods	top-1		top-3	
	EM	F1	EM	F1
BM25 + GTE	35.51	48.55	40.26	51.36
BM25 + GTE (Listwise Bandit)	36.21	49.06	39.81	50.95
BM25 + GTE (Standard PPO)	35.91	48.97	39.66	51.34
BM25 + GTE (RRPO)	**36.56**	**49.67**	**41.11**	**52.18**

Table 10: Training on alternative RL architectures on AmbigNQ.

We choose Qwen2.5-3B here, and evaluate our results on Qwen2.5-3B, 7B and 14B.

As the results show, the reranker trained under the guidance of Qwen2.5-3B remains effective, showing a clear performance lift over the baseline, although it is slightly less performant than the reranker trained with the 7B model.

G Analysis of Alternative RL Architectures

To validate the necessity of our sequential formulation and reference-anchored baseline, we compare RRPO against two standard RL variants on the AmbigNQ dataset: (1) Listwise Bandit, which models the selection of top-k documents as a single action, and (2) Standard PPO, which utilizes a parameterized value network (Critic) instead of our deterministic reference baseline.

As the results in Table 10 show, both alternative methods achieve improvements over the static baseline (BM25 + GTE) in Top-1 metrics, confirming the validity of incorporating LLM feedback. However, their performance on Top-3 metrics is inconsistent, often stagnating compared to the baseline. This suggests that standard methods struggle to optimize the utility of the entire list (ranking) while successfully promoting single relevant documents (hitting), likely due to the difficulty of credit assignment in single-action settings or the high variance of parametric critics. In contrast, RRPO delivers consistent gains across both Top-1 and Top-3 metrics, demonstrating that our sequential step-wise reward mechanism effectively guides the model to construct a high-utility set of documents rather than merely identifying a single hit.

H Statistical Significance Tests

Notes	HotpotQA		AmbigNQ	
	EM	F1	EM	F1
Trained Results Avg.	29.89	43.09	36.61	49.59
Trained Results Std.	0.09	0.12	0.15	0.15

Table 11: Statistical Significance Tests.

To rigorously validate that the performance improvements achieved by our RRPO framework are not a result of random chance, we conducted statistical significance tests. We employed a paired t-test to compare the per-query performance of our best-performing model, "+ gte reranker (RRPO)", against the primary baseline, "+ gte reranker". The test was performed on the F1 scores obtained for each query in the validation sets of both HotpotQA and AmbigNQ.

The null hypothesis (H_0) for our test is that there is no statistically significant difference between the mean F1 scores of the two models. We aim to reject this hypothesis by demonstrating a sufficiently low p-value.

The results are illustrated in Table 11. Our analysis yielded that On HotpotQA and AmbigNQ dataset, the paired t-test resulted in a p-value of $p < 0.01$. In both cases, the extremely low p-values are well below the standard significance level of $\alpha = 0.05$. Therefore, we reject the null hypothesis. This provides strong statistical evidence that the gains observed from applying RRPO are consistent and meaningful, confirming the effectiveness of our proposed training paradigm.

I Training Acceleration

To enhance training efficiency and mitigate the computational overhead of the RL-based RAG pipeline, we implement a comprehensive set of optimization strategies:

- **Kernel Acceleration:** We leverage Flash Attention and Automatic Mixed Precision (AMP) to optimize the reranker, and deploy

the LLM reader using vLLM (Kwon et al., 2023) to maximize inference throughput.

- **Retrieval Caching:** To eliminate redundant retrieval operations, we cache the top-N documents obtained from the initial retriever, reusing them across subsequent training epochs.
- **LLM Response Replay:** By enforcing deterministic LLM generation (temperature=0), we construct a replay buffer to store and reuse historical responses. This mechanism is critical, as it reduces the required LLM inference calls by nearly 50% in later training stages.
- **Distributed Training Optimization:** We employ gradient accumulation to increase the effective batch size and fine-tune communication schedules for improved parallel efficiency.

J Extended Comparison with DPO-based Joint Training Paradigms

As discussed in our related work (Section 2.3), a growing trend in aligning retrieval with generation involves applying preference optimization (e.g., DPO) to fine-tune models. To further contextualize RRPO within these recent advancements, we provide an extended empirical comparison against DynamicRAG (Sun et al., 2025), a representative concurrent work in this direction.

While sharing a similar high-level motivation of utilizing LLM feedback, the two frameworks explore fundamentally different paradigms. DynamicRAG adopts an end-to-end joint training approach, where a 7B LLM serves simultaneously as the reranker and the generator. While this enables deep synergy between components, it inherently introduces substantial computational overhead during both training and inference. In contrast, RRPO focuses on a modular, plug-and-play paradigm. We distill the RL-based alignment strictly into lightweight, encoder-only pointwise models (e.g., gte, jina, bge) with approximately 0.3B parameters, while keeping the downstream LLM generator strictly frozen.

As shown in Table 12, we present a direct empirical comparison on the HotpotQA dataset. It is worth noting that this constitutes an asymmetric comparison, given DynamicRAG's joint tuning of a 7B parameter space compared to our isolated tuning of a 0.3B reranker. Remarkably, despite this vast difference in scale and the lack of generator fine-tuning, the RRPO-optimized gte model (0.3B) effectively outperforms the 7B DynamicRAG model. Furthermore, when applied to other lightweight encoders, RRPO maintains highly competitive performance. These results demonstrate that our pointwise RL formulation provides a highly efficient alternative, effectively aligning the reranker with reader needs without the steep deployment costs of joint LLM training.

Methods	Model Size	EM
DynamicRAG (LLaMA-2)	7B	29.40*
RRPO (bge reranker)	0.3B	28.12
RRPO (jina reranker)	0.3B	29.26
RRPO (gte reranker)	**0.3B**	**30.03**

Table 12: Performance comparison between lightweight RRPO variants and the LLM-based DynamicRAG on the HotpotQA dataset. *Results directly cited from the original paper.