Rethinking Retrieval-Augmented Generation as a Cooperative Decision-Making Problem

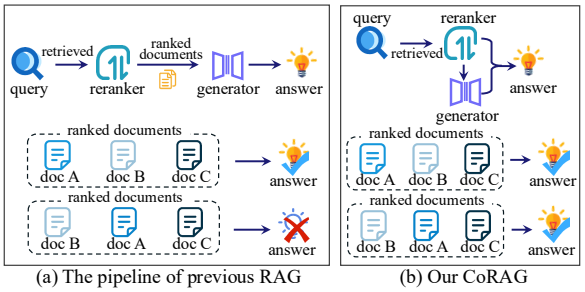
Lichang Song and Ting Long* **and Yi Chang***
Jilin University
songlc24@mails.jlu.edu.cn, longting@jlu.edu.cn, yichang@jlu.edu.cn

Abstract

Retrieval-Augmented Generation (RAG) has demonstrated strong effectiveness in knowledge-intensive tasks by grounding language generation in external evidence. Despite its success, many existing RAG systems are built based on a ranking-centric, asymmetric dependency paradigm, where the generation quality of the generator is highly dependent on reranking results of the reranker. To overcome this limitation, we propose Cooperative Retrieval-Augmented Generation (CoRAG), a framework that treats the reranker and the generator as peer decision-makers rather than being connected through an asymmetric dependency pipeline. By jointly optimizing their behaviors toward a shared task objective, the reranker and generator are encouraged to cooperate, ensuring that document reranking and generation work in concert to improve the final response. Experimental results demonstrate good generalization and improved generation stability of CoRAG, even when the model is trained on only around 10K PopQA samples. Our model released in https://github.com/CoderrrSong/CoRAG

1 Introduction

In recent years, Retrieval-Augmented Generation (RAG) (Lewis et al., 2020; Asai et al., 2024; Gao et al., 2023b; Zhao et al., 2024) has emerged as an important paradigm for enhancing the factuality and knowledge coverage of large language models (LLM) (Gao et al., 2023b). A typical RAG system consists of two core components: a retriever and a generator (Lewis et al., 2020; Oche et al.). Given a query, the retriever retrieves candidate documents from a large external corpus and further rerank them via reranker. The generator then conditions on the query and the reranked documents to produce the final response. By explicitly incorporating external knowledge during generation, RAG effectively mitigates hallucinations and achieves strong performance on open-domain question answering tasks (Lewis et al., 2020; Asai et al., 2024).



(a) The pipeline of previous RAG (b) Our CoRAG

Figure 1: Comparison with previous works. Previous works assume an asymmetric dependency between the reranker and the generator, whereas CoRAG treats them as equal participants optimized under a shared task-oriented reward.

As the reranker plays a critical role in shaping the document context provided to the generator (Lewis et al., 2020; Sharma, 2025), a growing body of recent work has focused on the design and optimization of rerankers and generators (Gao et al., 2023b; Asai et al., 2024; Sun et al., 2025; Shen et al., 2023; Jia et al., 2025). However, most existing RAG methods still adopt a ranking-centric, asymmetric-dependency paradigm (Figure 1(a)), where the reranker produces a fixed document ordering and the generator performs generation conditioned on these top-ranked documents. This design tightly couples generation with reranking decisions, making the generator highly sensitive to the reranking results. As shown in Figure 1(a), if a suboptimal reranker misranks a less relevant document at the top, the generator may produce an incorrect response, even though the optimal document is still present within the top-N set.

From an optimization perspective, this phenomenon reveals a deeper mismatch between reranking and generation. On the one hand, the

* Corresponding author

Findings of the Association for Computational Linguistics: ACL 2026, pages 43445–43456

generator is highly sensitive to fine-grained ranking results; on the other hand, learning an exact total order over multiple highly relevant documents is inherently harder than learning a relaxed ordering for the reranker. For instance, precisely ranking the top three relevant documents as positions 1, 2, and 3 is substantially more challenging than merely ensuring that these documents appear within the top few positions in any order.

A natural way to address this issue is to treat the reranker and generator as cooperative decision-makers optimizing a shared reward (Figure 1(b)). The cooperative multi-agent reinforcement learning (MARL) framework (Chen et al., 2025) provides a natural formulation for this problem, where the reranker and generator can coordinate and adapt to each other's needs. By jointly optimizing under a shared reward, the reranker is encouraged to learn a more accurate total order targeting the final results while relaxing the generator's asymmetric dependency on reranking quality. However, realizing this idea with standard MARL optimizers is challenging. A fundamental issue is that MARL's underlying MDP assumption structurally mismatches the core properties of ranking tasks. For example, MARL agents may independently select the same document for different positions, leading to invalid or degenerate rankings, which further compromises downstream generation (Chen et al., 2025).

In this paper, we propose a method called Cooperative Retrieval-Augmented Generation (CoRAG), which models the retrieval-augmented generation as a cooperative decision-making problem and devises a novel GRPO-style optimization for the reranker within the MARL framework. Unlike standard MARL optimizers that suffer from the MDP-ranking mismatch, our GRPO-style design makes the otherwise challenging MARL-based reranker optimization tractable. As a result, CoRAG mitigates the generator's asymmetric dependency to fine-grained reranking results and improves generation stability. Experimental results show that, despite being trained on only around 10K PopQA samples, CoRAG significantly outperforms baselines and generalizes well across multiple datasets and tasks.

In summary, our contributions are:

- We identify and formalize the asymmetric dependency between the reranker and generator in RAG, and propose a joint MARL optimization scheme to mitigate it.
- We propose a GRPO-style optimization for the reranker within a MARL framework to overcome the MDP-ranking mismatch.
- Extensive experiments demonstrate improved robustness and generalization of our CoRAG.

2 Problem Definition

A typical RAG system consists of two components: a retriever and a generator (Lewis et al., 2020; Oche et al.). Given an input query, the retriever retrieves a candidate document set from a large external corpus, which is further refined by a reranker before being used by the generator to generate the final response. In this work, we focus on the reranker in the retrieval stage together with the generator, as reranking plays a crucial role in improving the relevance and faithfulness of generation (Sharma, 2025). We model RAG as a cooperative multi-agent decision-making problem, treating the reranker and the generator as two cooperative agents that jointly determine the final generation outcome, and train them with the goal of maximizing a shared task-oriented objective defined on the generated response.

Specifically, given a query q and a candidate document set $\mathcal{D}$, the reranker $\mathcal{S}_\theta$ reranks and selects a set of documents $D \subseteq \mathcal{D}$, and the generator $\mathcal{G}_\phi$ generate a response $\hat{a}$ conditioned on q and D. The learning objective is defined as

$$\max_{\theta,\phi} \mathbb{E}_{D\sim\mathcal{S}_\theta(\cdot|q,\mathcal{D}),\, \hat{a}\sim\mathcal{G}_\phi(\cdot|q,D)} \left[R(a^*, \hat{a})\right], \quad (1)$$

where θ and ϕ are the parameters of the reranker and generator respectively. $R(a^*, \hat{a})$ denotes a task-oriented reward defined on the generated response. This formulation emphasizes that the reranker is optimized not for ranking accuracy, but for its contribution to downstream generation quality. Similarly, the generator is trained to produce outputs that effectively utilize the provided document configuration to maximize the same task-oriented reward. By aligning both components to the same outcome-oriented objective, the reranker and generator are encouraged to cooperate, ensuring that document reranking and generation in concert to improve the final response.

3 Method

An overview of our proposed Cooperative Retrieval-Augmented Generation (CoRAG) is shown in Figure 2. In the following sections, we

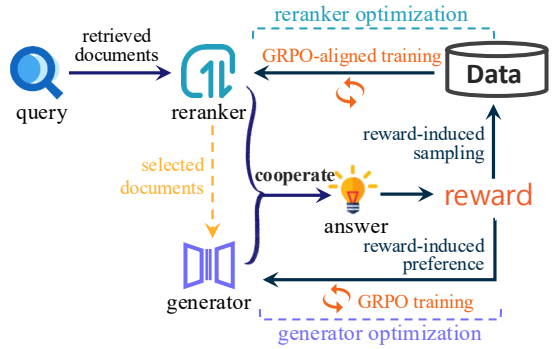


Figure 2: CoRAG overview. The reranker and generator cooperate to generate responses. The task-oriented reward derived from response guides GRPO-aligned training of the reranker and GRPO optimization of the generator.

first describe the reranker and generator in our CoRAG, and then discuss their joint optimization.

3.1 The Reranker

The reranker aims to refine the retrieved candidate document set $\mathcal{D}$ by re-ranking the documents according to their relevance, and selects a subset D to provide to the downstream generator for response generation. Specifically, we obtain D in the following steps:

Given a query q and a document d_i in the candidate document set $\mathcal{D} = \{d_1, d_2, ..., d_N\}$, we compute the relevance score of document d_i to q:

$$s_i = \mathcal{S}_\theta(q, d_i), \tag{2}$$

$\mathcal{S}_\theta$ denotes the reranker and we implemented with BGE-Reranker (Multi-Granularity, 2024).

We select a subset of top-K documents according to the relevance scores and feed them into the generator.

$$D = \{d_i \mid i \in \text{Top-K}(\{s_1, \ldots, s_N\})\} \tag{3}$$

3.2 The Generator

The Generator is responsible for generating the response $\hat{a}$ based on the selected document D and the query q. Specifically, we obtain the predicted response $\hat{a}$ with the following steps:

Given a query q and the top-K documents D, we input them into the generator model $\mathcal{G}_\phi$ to generate the final response:

$$\hat{a} = \mathcal{G}_\phi(q, D), \tag{4}$$

where $\mathcal{G}_\phi$ is the generator, typically implemented as an autoregressive language model (Radford et al., 2018; Brown et al., 2020) that generates the response token by token conditioned on both the query q and the document D.

3.3 The Optimization

The reranker and the generator are optimized under a shared task-oriented reward. From a multi-agent perspective, we treat the reranker and the generator as two cooperative agents with distinct decision roles. The reranker determines which documents to attend to, while the generator decides how to synthesize the final response based on the selected documents. The shared reward, defined on the final response, couples their behaviors and enables coordinated optimization toward task-oriented success.

$$r = \mathrm{R}(a^*, \hat{a}), \tag{5}$$

where $\mathrm{R}(a^*, \hat{a})$ denotes a task-oriented evaluation function, which we implement to return 1 if the generated response $\hat{a}$ contains the ground-truth response a^*, and 0 otherwise. In the following, we detail the optimization of the reranker and the generator under this shared task-oriented reward.

Reranker Optimization. Unlike standard learning-to-rank settings (Casalegno, 2022), document-level supervision is not directly available. To address that, we transform task-oriented rewards into document-level stochastic preference signals for reranker optimization. Since these signals indicate how documents collectively contribute to task success, we optimize the reranker in a group-relative preference aligned with GRPO (Shao et al., 2024).

First, to transform delayed task-level rewards into document-level stochastic preference signal, for each document d_i in the top-K document set D at training iteration t, we assign a binary success signal $l^{(t)}(q, d_i) \in \{0, 1\}$, indicating whether the generated response achieved task success when d_i was included. Although this signal provides only a coarse and noisy approximation of individual document contribution under multi-document conditioning, it enables scalable credit assignment without requiring explicit supervision.

Based on these signals,we estimate the expected task success associated with each document d_i:

$$\bar{l}(q, d_i) = \frac{1}{T} \sum_{t=1}^{T} l^{(t)}(q, d_i), \tag{6}$$

where T denotes the number of historical iterations. To account for uncertainty, we map $\bar{l}(q, d_i)$ to a smoothed Bernoulli parameter

$$p(q, d_i) = \alpha + (1 - 2\alpha) \cdot \bar{l}(q, d_i), \quad (7)$$

where $\alpha \in (0, 0.5)$, and we sample a stochastic preference label by:

$$p_i \sim \text{Bernoulli}\left(p(q, d_i)\right), \quad p_i \in \{0, 1\}. \quad (8)$$

This stochastic preference label serves as a surrogate feedback signal for reranker optimization, enabling us to optimize the reranker in a group-relative preference framework aligned with GRPO. To conduct the optimization, we model the reranker as a deterministic scoring function $\mathcal{S}_\theta(q, d_i)$ that induces a distribution over candidate documents: $\pi_\theta(d_i \mid q, \mathcal{D}) \propto \exp(\mathcal{S}_\theta(q, d_i))$. The group-relative advantage $\hat{A}(q, d)$ is derived from these preference labels: positive labels ($p_i = 1$) indicate relative advantage for document d_i within the group, while a negative label indicates a disadvantage. This leads to the GRPO objective:

$$\mathcal{L}^r_{\text{GRPO}} = -\mathbb{E}_{d_i \sim \pi_\theta}\left[\hat{A}(q, d_i) \log \pi_\theta(d_i \mid q, \mathcal{D})\right], \quad (9)$$

However, directly optimizing the stochastic objective suffers from high variance under noisy credit estimates. We therefore reinterpret the group-relative advantages as inducing pairwise preferences among candidate documents: documents with higher empirical success rates (more likely to receive positive labels) should be ranked higher. This allows us to reduce GRPO to a deterministic learning-to-rank problem. Constructing positive and negative sets $\mathcal{D}^+$ and $\mathcal{D}^-$ from the sampled labels, we adopt a margin-based pairwise ranking surrogate:

$$\mathcal{L}_{\text{rank}} = \sum_{d_i^+ \in \mathcal{D}^+} \sum_{d_j^- \in \mathcal{D}^-} \max\left(0,\ s_\theta(q, d_j^-) - s_\theta(q, d_i^+) + \gamma\right), \quad (10)$$

where γ is the margin hyperparameter. Although this reduction is not a strict equivalence to GRPO, the ranking loss preserves the group-relative preferences induced by the stochastic labeling scheme and the underlying GRPO objective in expectation. It thus provides a stable and efficient surrogate for optimizing the reranker while maintaining alignment with task-oriented rewards.

Generator Optimization. The generator defines a conditional generation policy $\pi_\phi(\hat{a} \mid q, D)$, where D is the set of top-K documents selected by the reranker. The generator is optimized using standard GRPO for conditional text generation:

$$\mathcal{L}_{\text{gen}} = -\mathbb{E}_{\hat{a} \sim \pi_\phi}\left[\hat{A}(\hat{a}) \log \pi_\phi(\hat{a} \mid q, D)\right], \quad (11)$$

where the group-relative advantage $\hat{A}(\hat{a})$ is computed from the task-oriented reward r using a baseline that compares the current response to other responses in the same training batch.

Overall, both the reranker and the generator are optimized with respect to the same task-oriented reward, enabling them to cooperatively improve retrieval relevance and generation quality.

4 Experiments

We conduct extensive experiments to evaluate the performance of CoRAG, and we particularly focus on the research questions: (i) Is CoRAG more effective than existing methods **(RQ1)**? (ii) How important are the reranker and the generator to overall performance, and is it necessary to jointly optimize them **(RQ2)**? (iii) How does performance vary with the number of documents used **(RQ3)**? (iv) Does CoRAG generalize to other tasks **(RQ4)**? (v) Does the CoRAG generator produce high-quality outputs as judged by other LLMs **(RQ5)**?

4.1 Experimental Setting

Datasets We evaluate our method on multiple knowledge-intensive benchmarks, following the dataset setup of recent works (Wei et al., 2024). Specifically, we evaluate on PopQA (Mallen et al., 2023), TriviaQA (Joshi et al., 2017), Natural Questions (NQ) (Kwiatkowski et al., 2019), ASQA (Stelmakh et al., 2022). Additionally, we include 2WikiMultiHopQA (Ho et al., 2020), a Wikipedia-based cross-document multi-hop QA dataset that requires models to reason across multiple documents to derive responses.

Table 1: Dataset statistics.

Dataset	Train	Test
PopQA	12,868	1,399
TriviaQA	78,785	11,313
NaturalQuestions	79,168	3,610
ASQA	4,353	948
2WikiMultiHopQA	167,454	12,576

Evaluation Metrics. Following InstructRAG (Wei et al., 2024), we report correctness, citation precision, and citation recall (Gao et al., 2023a) for

ASQA. For the other datasets, we use accuracy to measure whether the ground-truth responses are included in the generated outputs.

Baselines. Following InstructRAG (Wei et al., 2024), we compare our CoRAG with three groups of method: Baselines without Retrieval, relying solely on parametric knowledge, including ChatGPT, Llama-3-Instruct$_{8B}$, and Llama-3-Instruct$_{70B}$; RAG without Training, leveraging retrieved documents via in-context learning or prompting, such as In-Context RALM (Ram et al., 2023), Few-shot Demonstration with Instruction, and InstructRAG-ICL (Wei et al., 2024); and RAG with Training, involving explicit training in the retrieval-augmented framework, including Self-RAG (Asai et al., 2024) for iterative improvement via self-retrieval, RetRobust (Yoran et al., 2023) for noise robustness training, and InstructRAG-FT (Wei et al., 2024) for instruction-following fine-tuning with retrieval.

Implementation Details We adopt BGE-reranker-v2-m3 (Multi-Granularity, 2024) as the reranker and Llama-3-Instruct$_{8B}$ (Dubey et al., 2024) as the generator. During training, we use Llama-3-Instruct$_{8B}$ to provide coarse annotations for positive and negative documents in the training dataset, which helps the reranker learn more effectively by alleviating the sparsity of stochastic preference labels at the early stages of training. We set the learning rate of reranker to 5e-5, and the learning rate of generator to 1e-5. Both reranker and generator adopt the LoRA fine-tuning strategy for efficient parameter updating. We set $\gamma = 1$ in Eq.(10), and the top-K selected documents $K = 1$ to precisely attribute the impact of individual documents on task success. During inference, we set $K = 3$ for PopQA, TriviaQA, NQ and ASQA, and $K = 7$ for 2WikiMultiHopQA. The generator uses a temperature coefficient of 0.7 to balance generation diversity and stability. It is worth noting that our CoRAG is trained only on PopQA (Mallen et al., 2023), while all other datasets are used solely for evaluation, both due to our limited computational resources and to enable assessment of the generalization ability of our CoRAG.

4.2 Main Results (RQ1)

The results are reported in Table 2. As it is shown, our method (CoRAG) achieves outstanding performance. Notably, unlike the InstructRAG series of methods that perform separate training on each dataset, our model is trained only on PopQA. Even so, our method yields state-of-the-art results on four core tasks (PopQA, TriviaQA, NQ and 2WikiMultiHopQA), with the corresponding accuracy rates reaching 71.2%, 81.0%, 72.4% and 58.2% respectively, which significantly outperform existing methods such as RetRobust and InstructRAG-FT. This can be attributed to the joint optimization framework, in which the reranker progressively selects more relevant documents and the generator continuously improves its ability to extract and integrate information from them.

However, our CoRAG underperforms on the ASQA dataset. We attribute this primarily to the task discrepancy: ASQA requires synthesizing answers from multiple sources and handling ambiguous questions, which differs substantially from the factoid-style, single-answer questions prevalent in our training data (PopQA). Consequently, the retrieval-generator synergy optimized on factoid QA does not generalize effectively to this more complex, multi-answer setting.

Beyond overall performance gains, our method shows strong robustness to reranking order. Baselines degrade significantly with shuffled documents, while CoRAG remains stable (see Appendix C, Table 8 for detailed experiments).

4.3 Ablation Study (RQ2)

To investigate the importance of the reranker and the generator to overall performance, as well as whether it is necessary to jointly optimize them, we conduct ablation studies. Specifically, we have five variants:

- **RTrain**: only finetune the reranker with the labels annotated by Llama-3-Instruct$_{8B}$.
- **GTrain**: only finetune the generator with GRPO.
- **RGReplace**: Replace the generator of CoRAG with Llama-3-Instruct$_{8B}$, and replace the reranker with BGE-Reranker in inference.
- **GReplace**: Replace the generator of CoRAG with Llama-3-Instruct$_{8B}$ in inference.
- **RReplace**: Replace the reranker of CoRAG with BGE-Reranker in inference.

The result presented in Table 3 , we can observe that: (1) The individually trained RTrain and GTrain underperform CoRAG across all four datasets, which implies that a multi-agent cooperative optimization formulation may better capture

Table 2: Overall results of our method and baselines on five benchmarks. Baseline results are reported in InstructRAG (Wei et al., 2024). - indicates the results are not reported or not applicable. The best and second-best performances are highlighted in bold and with an underline, respectively. Notably, our model is trained only on PopQA (Mallen et al., 2023), while all other datasets are used exclusively for evaluation.

Method	PopQA (acc)	TriviaQA (acc)	NQ (acc)	2WikiMultiHopQA (acc)	ASQA (em)	ASQA (pre)	ASQA (rec)
Baselines w/o Retrieval							
Vanilla Zero-shot Prompting							
ChatGPT	29.3	74.3	–	–	35.3	–	–
Llama-3-Instruct$_{8B}$	22.8	69.4	46.6	45.6	30.6	–	–
Llama-3-Instruct$_{70B}$	28.9	80.6	57.9	57.5	39.1	–	–
RAG w/o Training							
In-Context RALM (Ram et al., 2023)							
ChatGPT	50.8	65.7	–	–	40.7	65.1	76.6
Llama-3-Instruct$_{8B}$	62.3	71.4	56.8	43.4	40.0	62.1	66.4
Llama-3-Instruct$_{70B}$	63.8	76.3	60.2	51.2	43.1	62.9	67.6
Few-Shot Demo. w/ Instruction							
Llama-3-Instruct$_{8B}$	63.1	74.2	60.1	45.3	42.6	55.0	64.4
Llama-3-Instruct$_{70B}$	63.9	79.1	62.9	53.9	45.4	49.3	57.1
InstructRAG-ICL (Wei et al., 2024)							
Llama-3-Instruct$_{8B}$	64.2	76.8	62.1	50.4	44.7	70.9	74.1
Llama-3-Instruct$_{70B}$	65.5	81.2	66.5	57.3	47.8	69.1	71.2
RAG w/ Training							
Vanilla Supervised Fine-tuning							
Llama-3-Instruct$_{8B}$	61.0	73.9	56.6	56.1	43.8	–	–
Self-RAG (Asai et al., 2024)							
Llama-2$_{7B}$	55.8	68.9	42.4	35.9	30.0	66.9	67.8
Llama-2$_{13B}$	56.3	70.4	46.4	36.0	31.4	**70.3**	**71.3**
Llama-3-Instruct$_{8B}$	55.8	71.4	42.8	32.9	36.9	69.7	69.7
RetRobust (Yoran et al., 2023)							
Llama-2$_{13B}$	–	–	39.6	51.5	–	–	–
Llama-3-Instruct$_{8B}$	56.5	71.5	54.2	54.7	40.5	–	–
InstructRAG-FT (Wei et al., 2024)							
Llama-3-Instruct$_{8B}$	66.2	78.5	65.7	57.2	**47.6**	65.7	70.5
CoRAG							
Llama-3-Instruct$_{8B}$	**71.2**	**81.0**	**72.4**	**58.2**	45.8	54.9	48.9

the interaction between the reranker and the generator, leading to improved performance compared to independent optimization. (2) Comparing RGReplace, GReplace and CoRAG reveals that the generator contributes more significantly to performance improvement, whereas the reranker plays a relatively limited role. We attribute this phenomenon to the joint optimization. Because the reranker and generator are jointly optimized, the generator may achieve strong performance even under suboptimal reranking signals, potentially weakening the learning pressure on the reranker and reflecting a trade-off in the current design.

To examine this hypothesis, we conduct cross-component experiments by swapping rerankers and generators between CoRAG and other RAG frameworks (Self-RAG and InstructRAG). As shown in Figure 3, replacing the reranker with our reranker while keeping other generators (Self-RAG and InstructRAG) fixed yields only marginal improvements and occasionally leads to performance degradation. This suggests that the standalone effectiveness of our reranker is relatively limited. However, when paired with our generator, CoRAG consistently outperforms the variant that combines our generator with BGE-Reranker. This contrast reveals an inherent tension between reranking effectiveness and generator sensitivity: jointly optimizing the reranker and generator encourages the generator to rely less on fine-grained ranking signals, which in turn limits the observable gains from further improving the reranker in isolation. Nevertheless, the strong performance of CoRAG indicates that its reranker and generator are well aligned and mutually reinforcing when optimized together.

Table 3: Ablation study.

	PopQA	NQ	TriviaQA	2Wiki
RTrain	66.19	62.71	75.57	49.36
GTrain	66.54	63.37	76.25	51.53
RGReplace	65.83	63.21	74.97	49.72
GReplace	66.26	62.46	75.00	50.00
RReplace	70.12	69.77	80.76	56.05
CoRAG (Ours)	**71.26**	**72.49**	**81.00**	**58.26**

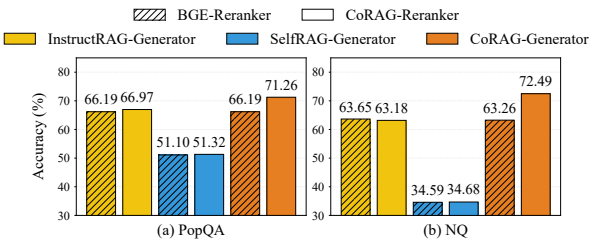


Figure 3: Cross-validation results with different reranker and generator combinations (Top-3 setting).

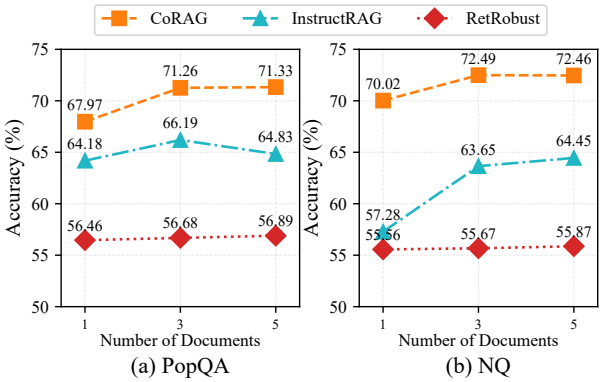


Figure 4: Impact of the document number.

4.4 Top-N Analysis (RQ3)

To analyze how the number of documents provided to the generator affects performance, we evaluate top-1(T1), top-3(T3), and top-5(T5) settings across different datasets. The results are presented in Figure 4, which reveal three insights: (1) CoRAG outperforms InstructRAG and RetRobust under all document count settings on both datasets, this indicates that CoRAG can effectively utilize the effective information of documents under different document number; (2) on PopQA, InstructRAG's performance declines as document number increases (e.g., dropping from 66.19% at T3 to 64.83% at T5), implying more documents may trigger the noise of less relevant documents, and leading to hallucinations. In contrast, our CoRAG demonstrates a consistent upward trend in performance on both PopQA and NQ, showing strong robustness.

4.5 Cross-Task Evaluation (RQ4)

To further explore the cross-domain generalization capability of CoRAG, we conduct additional experiments on other tasks, following (Wei et al., 2024). Specifically, we evaluate the generator of our CoRAG and other generator (Llama-3-Instruct$_{8B}$ and InstructRAG) on code generation datasets HumanEval, HumanEval+ (Chen, 2021), and table question answering dataset WikiTable Questions (Pasupat and Liang, 2015). Results are summarized in Table 4.

Table 4: The cross-task evaluation. The InstructRAG in the table is trained on the Pop dataset. WTQ represents WikiTable Questions. The best and second-best performances are highlighted in bold and with an underline.

Metric	Llama-3-Instruct$_{8B}$	InstructRAG	CoRAG
human-eval			
pass@1	**64.45**	55.85	63.96
pass@10	83.53	76.82	**85.97**
human-eval+			
pass@1	56.58	49.26	**57.43**
pass@10	77.43	70.12	**79.26**
WTQ			
accuracy	49.10	**68.57**	68.11

As shown in Table 4, compared with the Llama-3-Instruct$_{8B}$ and InstructRAG, our generator achieves the best or second-best performance across all tasks: on HumanEval and HumanEval+, CoRAG achieves the strongest results on pass@10 and achieves the best pass@1 performance on the more challenging HumanEval+, indicating improved generation quality and robustness on code generation. On WTQ, a table-based question answering benchmark, the generator of CoRAG achieves competitive accuracy, closely matching the strongest baseline while substantially outperforming Llama-3-Instruct$_{8B}$. Overall, these results demonstrate that the generator of CoRAG remains effective across both code generation and table-based reasoning tasks.

4.6 Evaluation with LLM-as-a-judge (RQ5)

Following InstructRAG (Wei et al., 2024), We use LLMs as a judgment to further evaluate the generation of our CoRAG. Specifically, for a given question and documents ranked by reranker, responses are generated by generator and its quality is assessed using different LLMs. The specific prompt template is shown in Appendix B. We choose LlaMa, GPT, DeepSeek, Qwen as the judge. For LlaMa, we use version Llama-3-Instruct$_{8B}$. For GPT, we use version gpt-4o. For DeepSeek, we use version deepseek-v3.2. For Qwen, we use version qwen3-vl-235b-a22b-instruction. The results are presented in Table 5 . As shown in Table 5, CoRAG consistently outperforms both InstructRAG and RetRobust across all evaluation settings, regardless of the LLM used as the judge. CoRAG achieves the highest scores under every evaluator, including pattern-based metrics as well as LlaMa,

Table 5: Evaluation with LLM-as-a-judge (PopQA).

Method	InstructRAG	RetRobust	CoRAG
Pattern-based	66.19	56.68	**71.26**
LlaMa	75.63	25.52	**89.21**
GPT	60.83	54.90	**65.90**
DeepSeek	59.69	34.10	**64.26**
Qwen	60.69	57.26	**66.12**
Average	64.60	45.69	**71.35**

GPT, DeepSeek, and Qwen, indicating robust and consistently preferred generation quality. Notably, the performance gap is most pronounced under the LlaMa judge, where CoRAG receives higher scores than other LLMs. We attribute this effect to the fact that CoRAG is fine-tuned from LlaMa, which may lead to a higher alignment between CoRAG's outputs and the LlaMa-based judge.

5 Related Work

RAG enhances LLMs' performance by fusing external documents and is the mainstream solution for knowledge-intensive tasks. Existing research could be categorized into three types based on core efficiency-enhancing mechanisms (Gao et al., 2023b; Zhao et al., 2023). The first category is data-driven methods, focusing on the mining and reconstruction of information at the query or document level: Decomposition Prompting (Khot et al., 2022), which splits complex tasks through prompt engineering. EviNoteRAG (Dai et al., 2025) annotates uncertain information in documents with a note-taking-first approach. HtmlRAG (Tan et al., 2025) uses HTML instead of plain text to preserve semantic structural information. The second category is model-driven methods, which improve a model's ability to interpret, filter and utilize retrieved documents through fine-tuning. Representative works include: RetRobust (Yoran et al., 2023) enhances robustness through contrastive training with positive and negative samples. InstructRAG (Wei et al., 2024) explicitly learns the denoising process through self-synthesized reasoning. DynamicRAG (Sun et al., 2025) optimizes the order and quantity of documents used to train the reranker through generator feedback. The third category is strategy-driven methods, also called agentic RAG, which introduces agentic behaviors to dynamically adjust retrieval and generation strategies. FLARE (Jiang et al., 2023) triggers lookahead retrieval when encountering uncertain tokens during generation. SelfRAG (Asai et al., 2024) introduces a reflection module to synchronously and dynamically adjust both retrieval and generation. MA-RAG (Nguyen et al., 2025) decomposes workflows into sub-agents via CoT. ComposeRAG (Wu et al., 2025) enables modular agent composition. MMOA-RAG (Chen et al., 2025) jointly optimizes multiple RAG modules via MARL, but forces heterogeneous modules with distinct functionalities to share a single LLM, risking inter-module knowledge conflicts and incurring substantial inference overhead.

6 Conclusion

In this paper, we reformulate reranking and generation as two equal participants in a cooperative decision-making problem and propose Cooperative Retrieval-Augmented Generation (CoRAG). Unlike conventional RAG frameworks, where the generator exhibits an asymmetric dependency on the reranker and generation quality is highly sensitive to the reranking results, CoRAG treats the reranker and generator as two peer decision-makers. Specifically, the reranker decides which documents, and in what organization, to present to the generator, while the generator determines how to effectively utilize the provided information to accomplish the generation task. Through this cooperative formulation, CoRAG alleviates the generator's reliance on overly strict document ordering and enables more robust coordination between retrieval and generation, leading to improved overall performance.

7 Limitations

Although CoRAG improves robustness by reducing the generator's sensitivity to reranking results, this design may also attenuate the impact of further reranker improvements on generation quality. This reflects an inherent tension between reranking effectiveness and generation sensitivity under joint optimization, which we leave for future exploration.

8 Acknowledgments

This work has been supported by the National Key R&D Program of China (2023YFF0905400), the National Natural Science Foundation of China (Grants No.62307020), and the New Cornerstone Science Foundation through the XPLORER PRIZE.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Francesco Casalegno. 2022. Learning to rank: A complete guide to ranking using machine learning. *Towards Data Science*.

Mark Chen. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

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. *arXiv preprint arXiv:2501.15228*.

Yuqin Dai, Guoqing Wang, Yuan Wang, Kairan Dou, Kaichen Zhou, Zhanwei Zhang, Shuo Yang, Fei Tang, Jun Yin, Pengyu Zeng, and 1 others. 2025. Evinote-rag: Enhancing rag models via answer-supportive evidence notes. *arXiv preprint arXiv:2509.00877*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. 2023a. Enabling large language models to generate text with citations. *arXiv preprint arXiv:2305.14627*.

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

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*.

Pengyue Jia, Derong Xu, Xiaopeng Li, Zhaocheng Du, Xiangyang Li, Yichao Wang, Yuhao Wang, Qidong Liu, Maolin Wang, Huifeng Guo, and 1 others. 2025. Bridging relevance and reasoning: Rationale distillation in retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 4242–4256.

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.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2022. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*.

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.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

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

Multi-Linguality Multi-Functionality Multi-Granularity. 2024. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation.

Thang Nguyen, Peter Chin, and Yu-Wing Tai. 2025. Ma-rag: Multi-agent retrieval-augmented generation via collaborative chain-of-thought reasoning. *arXiv preprint arXiv:2505.20096*.

AJ Oche, AG Folashade, T Ghosal, and A Biswas. A systematic review of key retrieval-augmented generation (rag) systems: Progress, gaps, and future directions. arxiv 2024. *arXiv preprint arXiv:2409.15730*.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. *arXiv preprint arXiv:1508.00305*.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, and 1 others. 2018. Improving language understanding by generative pre-training.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 11:1316–1331.

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*.

Chaitanya Sharma. 2025. Retrieval-augmented generation: A comprehensive survey of architectures, enhancements, and robustness frontiers. *arXiv preprint arXiv:2506.00054*.

Weizhou Shen, Yeyun Gong, Yelong Shen, Song Wang, Xiaojun Quan, Nan Duan, and Weizhu Chen. 2023. Joint generator-ranker learning for natural language generation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7681–7699.

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

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. *arXiv preprint arXiv:2505.07233*.

Jiejun Tan, Zhicheng Dou, Wen Wang, Mang Wang, Weipeng Chen, and Ji-Rong Wen. 2025. Htmlrag: Html is better than plain text for modeling retrieved knowledge in rag systems. In *Proceedings of the ACM on Web Conference 2025*, pages 1733–1746.

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

Ruofan Wu, Youngwon Lee, Fan Shu, Danmei Xu, Seung-won Hwang, Zhewei Yao, Yuxiong He, and Feng Yan. 2025. Composerag: A modular and composable rag for corpus-grounded multi-hop question answering. *arXiv preprint arXiv:2506.00232*.

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

Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, Jie Jiang, and Bin Cui. 2024. Retrieval-augmented generation for ai-generated content: A survey. *arXiv preprint arXiv:2402.19473*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, and 1 others. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2).

A The training of CoRAG

The training of CoRAG has been summarized as Algorithm 1.

Algorithm 1 Training of CoRAG

for $t = 1$ to T **do**
  for query $q \in Q$ **do**
    Compute score $s_i = \mathcal{S}_\theta(q, d_i)$;
    Select top-K documents D according to scores;
    Generate answer: $\hat{a} = \mathcal{G}_\phi(q, D)$
    Compute reward: $\mathrm{R}(a^*, \hat{a})$;
    Estimate the expected task success for $d_i \in D$;
    Sample a stochastic preference label;
    Update reranker according to $\mathcal{L}_{\text{rank}}$;
    Update generator according to $\mathcal{L}_{\text{gen}}$;
  end for
end for

B Prompt Template

We provide the prompt used for LLM-as-a-Judge in Table 6, and the prompt used by CoRAG in Table 7.

Table 6: Prompt of LLM-as-a-Judge

Prompt: You are an expert evaluator for large model responses. Your core task is to determine whether the large model's response points to any of the correct answers.
Please conduct the evaluation based on the following information:
1. Question: {question}
2. List of correct answers: {answers}
3. Large model's response: {response}
Evaluation Rules:
1. If the core information and key content of the large model's response point to any answer in the correct answer list (semantic consistency is acceptable, no need for word-for-word matching), please output "yes".
2. If the large model's response deviates from all correct answers, contains obvious errors, or fails to effectively respond to the question, please output "no".
3. You only need to output a single word: either "yes" or "no", without any additional redundant content.

Output: yes

Table 7: Prompt of Inference

Prompt: You are tasked with answering the given question by analyzing a set of documents. Please follow this STRICT TWO-STEP PROCESS:

—

STEP 1: Document Analysis

For each document:

- First, extract potentially relevant information from the original document. This includes facts, names, dates, or statements that may relate to the question, even if the connection is not immediately obvious.

- Then, explain the reason for the information extraction in your previous step based on the question. (e.g., how the document addresses the question's focus).

STEP 2: Final Answer

- Summarize your answer to the question based on the analysis above. - If none of the documents are helpful or relevant, answer based on your own general knowledge. In that case, clearly state that you are doing so.

—

Use the following format for your response:

Step 1: Document Analysis

Document 1: - Extraction: ... - Explanation: ...

Document 2: - Extraction: ... - Explanation: ...

Step 2: Final Answer

Well-supported answer, based on the relevant documents. If no relevant documents, answer based on general knowledge and say so explicitly.

—

EXAMPLE:

Question: Who is the author of The Mahdi?

Step 1: Document Analysis

Document 1: - Extraction: 'Mahdi' is a thriller novel by Philip Nicholson written in 1981 under the identity of a. J. Quinnell ... - Explanation: This document directly states that the author of 'Mahdi' is Philip Nicholson. Upon re-examining the question "Who is the author of The Mahdi?", the document mentions the corresponding book title and provides the author information requested.

Document 2: - Extraction: 'The Mahdi' was published by Philip Nicholson in 1981 under the pen name A.J. Quinnell, establishing his presence in the thriller genre with a novel known for its gripping plot and enduring popularity... - Explanation: This document directly states that 'The Mahdi' was published by Philip Nicholson. Upon re-examining the question "Who is the author of The Mahdi?", the document mentions the corresponding book title and provides the author information requested.

Step 2: Final Answer

Based on Documents 1 and 2, The answer to question 'Who is the author of The Mahdi?' is A.J. Quinnell, a pseudonym of Philip Nicholson.

—

Now it is your turn to analyze the following documents and answer the given question by following the two-step process.

{context} {question}

C Additional Experimental Results

C.1 Robustness to Document Order

To validate whether our CoRAG reduces the ranking sensitivity, we conduct shuffle experiments on PopQA and NQ datasets. Specifically, we evaluate Llama-3-Instruct$_{8B}$, the generator of InstructRAG (InstructRAG-Gen), and the generator of CoRAG (CoRAG-Gen), under two conditions: (1) Ranked: documents are provided in the order produced by the reranker; (2) Shuffled: the document order is randomly shuffled before being fed into the generator. All other settings remain identical to the main experiments.

As shown in Table 8, while Llama-3-Instruct$_{8B}$ shows a slight performance drop on both PopQA and NQ after shuffling, and InstructRAG-Gen suffers a more notable drop of 2.96% on NQ after shuffling, CoRAG-Gen remains stable, confirming its robustness to document order.

Table 8: Performance comparison under shuffled vs. ranked document order on PopQA and NQ datasets

	PopQA		NQ	
	Ranked	Shuffle	Ranked	Shuffle
Llama-3-Instruct$_{8B}$	65.83	65.68	64.65	63.68
InstructRAG-Gen	66.69	66.33	64.45	61.49
CoRAG-Gen	**71.26**	**71.05**	**72.46**	**72.10**