RankCoT: Refining Knowledge for Retrieval-Augmented Generation through Ranking Chain-of-Thoughts

Mingyan Wu[1], **Zhenghao Liu**[1]*, **Yukun Yan**[2]*,
Xinze Li[1], **Shi Yu**[2], **Zheni Zeng**[2], **Yu Gu**[1], **Ge Yu**[1]
[1]School of Computer Science and Engineering, Northeastern University, China
[2]Department of Computer Science and Technology, Institute for AI, Tsinghua University, China
Beijing National Research Center for Information Science and Technology, China

Abstract

Retrieval-Augmented Generation (RAG) enhances the performance of Large Language Models (LLMs) by incorporating external knowledge. However, LLMs still encounter challenges in effectively utilizing the knowledge from retrieved documents, often being misled by irrelevant or noisy information. To address this issue, we introduce RankCoT, a knowledge refinement method that incorporates reranking signals in generating CoT-based summarization for knowledge refinement based on given query and all retrieval documents. During training, RankCoT prompts the LLM to generate Chain-of-Thought (CoT) candidates based on the query and individual documents. It then fine-tunes the LLM to directly reproduce the best CoT from these candidate outputs based on all retrieved documents, which requires LLM to filter out irrelevant documents during generating CoT-style summarization. Additionally, RankCoT incorporates a self-reflection mechanism that further refines the CoT outputs, resulting in higher-quality training data. Our experiments demonstrate the effectiveness of RankCoT, showing its superior performance over other knowledge refinement models. Further analysis reveals that RankCoT can provide shorter but effective refinement results, enabling the generator to produce more accurate answers. All code and data are available at https://github.com/NEUIR/RankCoT.

1 Introduction

Retrieval-Augmented Generation (RAG) (Lewis et al., 2020b; Guu et al., 2020; Ram et al., 2023; Shi et al., 2024) empowers Large Language Models (LLMs) to access external knowledge, providing up-to-date information during the generation process. RAG models have demonstrated their effectiveness in mitigating the hallucination problem commonly encountered by LLMs (Shuster et al., 2021), enhancing the performance of LLMs, such as GPT-4 (Achiam et al., 2023) and LLaMA (Touvron et al., 2023), in different NLP tasks.

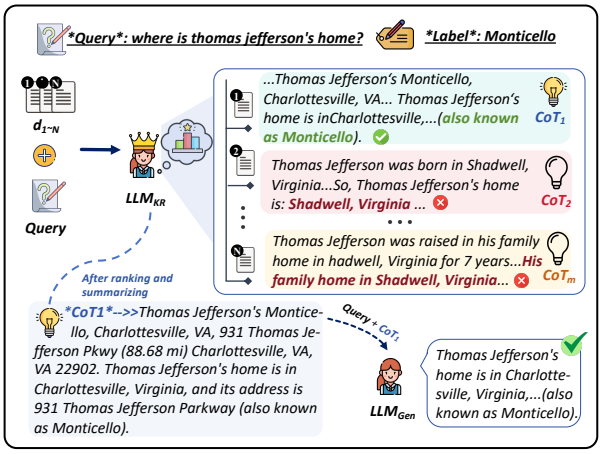


Figure 1: Illustration of implicit ranking and summarization in the RankCoT model. We present how knowledge refinement can be achieved by incorporating reranking into CoT-based summarization.

RAG models typically use dense retrieval methods (Karpukhin et al., 2020; Xiong et al., 2021) to retrieve query-relevant documents from external knowledge bases. These documents, along with the query, are then fed as the input context into LLMs (Ram et al., 2023). Thriving on their in-context learning capabilities (Brown et al., 2020; Dong et al., 2023), LLMs are able to extract relevant semantics from retrieved documents and generate appropriate responses to address the given query. However, the potential knowledge conflict between external knowledge and parameterized memory still poses a challenge for LLMs in generating precise responses (Chen et al., 2024b; Asai et al., 2024b).

Many RAG models focus on building modular RAG pipelines to enhance retrieval performance (Gao et al., 2024; Asai et al., 2024a). These models primarily aim to refine the retrieved knowl-

* indicates corresponding author.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 12857–12874

edge by assessing the relevance of each document to the query and subsequently filtering out irrelevant ones (Yan et al., 2024; Asai et al., 2024a). However, the reranking model still requires feeding the remaining documents into LLMs, which means that query-unrelated content within a relevant document may still mislead the generators (Xu et al., 2024a). Some models address this problem by prompting LLMs to summarize query-relevant knowledge from the retrieved documents, thereby reducing the influence of irrelevant information (Vig et al., 2022; Yu et al., 2024a; Xu et al., 2024a). This summarization approach often incorporates information from unrelated documents as part of the summaries, resulting in the introduction of noise. Both the reranking and summarization modules have advantages for knowledge refinement. However, in existing RAG systems, these modules are typically modeled separately by prompting the same LLMs.

This paper presents RankCoT, a knowledge refinement method that combines the strengths of both ranking and summarization to effectively enhance the process for retrieval result refinement. As shown in Figure 1, we feed both the query and all retrieved documents into the RankCoT model, which incorporates reranking signals in generating CoT-based summarization as knowledge refinements, thereby aiding LLMs in generating more accurate responses for answering the given query. During training the RankCoT model, we independently feed the query and retrieved document to the LLM, asking it to generate several Chain-of-Thought (CoT) responses to answer the question, which can be considered as summarization results. We then design the self-refinement model to prompt LLMs to answer the question according to these sampled CoTs, helping to refine the CoT results for more effective training. If the refined CoT contains the ground truth answer, it is considered a positive refinement result, while those that do not contain the ground truth answer are considered negative refinements.

Our experiments demonstrate that RankCoT outperforms all baseline models, achieving over a 2% improvement. Notably, RankCoT proves effective across LLMs of various scales. It generates shorter knowledge refinement results compared to both reranking and summarization methods, while enhancing the response accuracy of generator. Further analysis reveals that RankCoT successfully incorporates ground truth answers into the knowledge refinement results, while also including more query-relevant content. Additionally, RankCoT effectively extracts crucial semantics from the retrieved documents and alleviates the conflict between retrieved contents and internal knowledge.

2 Related Work

Retrieval-Augmented Generation (RAG) aims to enhance Large Language Models (LLMs) by enabling them to access external knowledge bases, providing up-to-date information during the generation process (Shi et al., 2024; Ram et al., 2023). This approach has demonstrated promising results across various NLP tasks, including open-domain question answering (Izacard et al., 2023), code generation (Zhou et al., 2023), and dialogue (Shuster et al., 2022). In these RAG models, retrieved documents are typically used as context to assist LLMs in generating more accurate responses (Ram et al., 2023). However, the conflict between the external knowledge and the parametric memory of LLMs often undermines the effectiveness of current RAG systems (Asai et al., 2024b; Xie et al., 2024; Chen et al., 2024b).

To mitigate the potentially negative impact of retrieved knowledge, existing models focus on refining the external knowledge through various modules designed to help LLMs generate more precise responses. Earlier works concentrate on reranking the retrieved documents (Yu et al., 2023; Shi et al., 2024; Yu et al., 2024b), while others employ query-focused summarization techniques (Vig et al., 2022; Xu et al., 2023) to reduce noise. However, reranking models often overlook noise within individual passages, and summarization models may fail to account for query-document relevance, sometimes incorporating misleading content in the summarization results. Chain-of-Note (Yu et al., 2024a) attempts to instruct LLMs to generate query-related notes when answering a given query. This model incorporates the knowledge refinement process into the reasoning stage (Wei et al., 2022) and heavily relies on the capabilities of LLMs, which may limit its applicability in RAG systems (Gao et al., 2024).

Modular RAG systems (Gao et al., 2024; Xu et al., 2024c) focus on refining external knowledge through different modules implemented by LLMs, which have become a key trend in the RAG area. For instance, Self-RAG (Asai et al., 2024a) uses different tags for adaptive retrieval (Jiang et al., 2023) and self-reflection to refine knowledge. Some ap-

proaches also focus on reformulating queries to identify more useful documents for answering questions (Yan et al., 2024; Trivedi et al., 2023). Yan et al. (2024) introduce a retrieval evaluator that acts as a judge to trigger query reformulation, search, and knowledge refinement actions to supply more accurate evidence for generation.

To further improve the performance of modular RAG systems, these models focus on fine-tuning various components of the RAG framework. Some efforts aim to align the information needs between the retriever and the generator by optimizing the retrievers based on feedback from the generation models (Yu et al., 2023; Shi et al., 2024; Izacard and Grave, 2021). Lin et al. (2024) adapt LLMs within the RAG setting by constructing instruction-tuning data for Supervised Fine-Tuning (SFT), enabling the models to better leverage the retrieved documents. Additionally, Li et al. (2024) use Direct Preference Optimization (DPO) (Rafailov et al., 2024) to jointly optimize the modules in a RAG system, aligning their data preferences.

3 Methodology

As illustrated in Figure 2, this section introduces the RankCoT method. First, we introduce the preliminary of knowledge refinement in Retrieval-Augmented Generation (RAG) systems (Sec. 3.1). And then we describe how to optimize LLMs to produce more effective chain-of-thoughts for knowledge refinement (Sec. 3.2).

3.1 Preliminary of Knowledge Refinement in Retrieval-Augmented Generation Systems

Given a query q and a set of retrieved documents $D = \{d_1, d_2, \ldots, d_n\}$, vanilla RAG system (Ram et al., 2023) uses retrieved documents as context and leverages the in-context learning method to help the generation model $\mathcal{M}_{\text{Gen}}$ produce the answer y_{Gen}:

$$(q, D) \rightsquigarrow \mathcal{M}_{\text{Gen}} \rightsquigarrow y_{\text{Gen}}. \tag{1}$$

Instead of directly feeding retrieved documents to the generation model ($\mathcal{M}_{\text{Gen}}$), some RAG models (Gao et al., 2024; Asai et al., 2024a) design various modules to refine retrieved documents D:

$$(q, D) \rightsquigarrow \mathcal{M}_{\text{KR}} \rightsquigarrow y_{\text{KR}}. \tag{2}$$

The refined knowledge y_{KR} is then passed to the generation model to mitigate the negative impact of retrieval noise:

$$(q, y_{\text{KR}}) \rightsquigarrow \mathcal{M}_{\text{Gen}} \rightsquigarrow y_{\text{Gen}}. \tag{3}$$

In the rest of this subsection, we will introduce different methods for implementing the knowledge refinement model $\mathcal{M}_{\text{KR}}$ in Eq. 2, including reranking, summarization, and RankCoT.

Reranking. Following previous work (Asai et al., 2024a), we prompt LLMs to evaluate the relevance of the i-th retrieved document d_i with respect to the query q, and output a binary label y^i_{Rerank} for filtering out noisy documents:

$$y^i_{\text{Rerank}} = \text{LLM}(\text{Instruct}_{\text{Rerank}}, q, d_i), \tag{4}$$

where $\text{Instruct}_{\text{Rerank}}$ prompts the LLM to assess the relevance between q and d_i. The prediction label y^i_{Rerank} can be "YES" or "NO", indicating whether the i-th document d_i is relevant or irrelevant to the query q. We then retain the documents predicted as "YES" and construct the filtered document set $\{d_1, \ldots, d_k\}$. The knowledge refinement result y_{KR} is then represented as:

$$y_{\text{KR}} = d_1 \oplus \cdots \oplus d_k, \tag{5}$$

where $\oplus$ is the concatenation operation.

Summarization. Another approach for knowledge refinement is summarization, which aims to extract query-related content from the retrieved documents (Vig et al., 2022). The knowledge refinement result can be obtained as:

$$y_{\text{KR}} = \text{LLM}(\text{Instruct}_{\text{Sum}}, q, D), \tag{6}$$

where $\text{Instruct}_{\text{Sum}}$ is the instruction prompting the LLM to generate a summary. Unlike reranking, summarization directly generates the refined knowledge, avoiding the need to feed raw documents to the generation model $\mathcal{M}_{\text{Gen}}$.

RankCoT. RankCoT further incorporates a Chain-of-Thought (CoT) (Wei et al., 2022) into the knowledge refinement process:

$$y_{\text{KR}} = \text{LLM}(\text{Instruct}_{\text{CoT}}, q, D), \tag{7}$$

where $\text{Instruct}_{\text{CoT}}$ is the instruction that prompts the LLM to generate CoT. RankCoT incorporates the chain-of-thought reasoning as the knowledge refinement result y_{KR} to extract relevant knowledge from retrieved documents D, thereby assisting RAG models in answering the query. Unlike summarization, RankCoT integrates the reranking mechanism during the chain-of-thought generation (Sec. 3.2) to mitigate the influence of noisy documents (Liu et al., 2024; Xie et al., 2024).

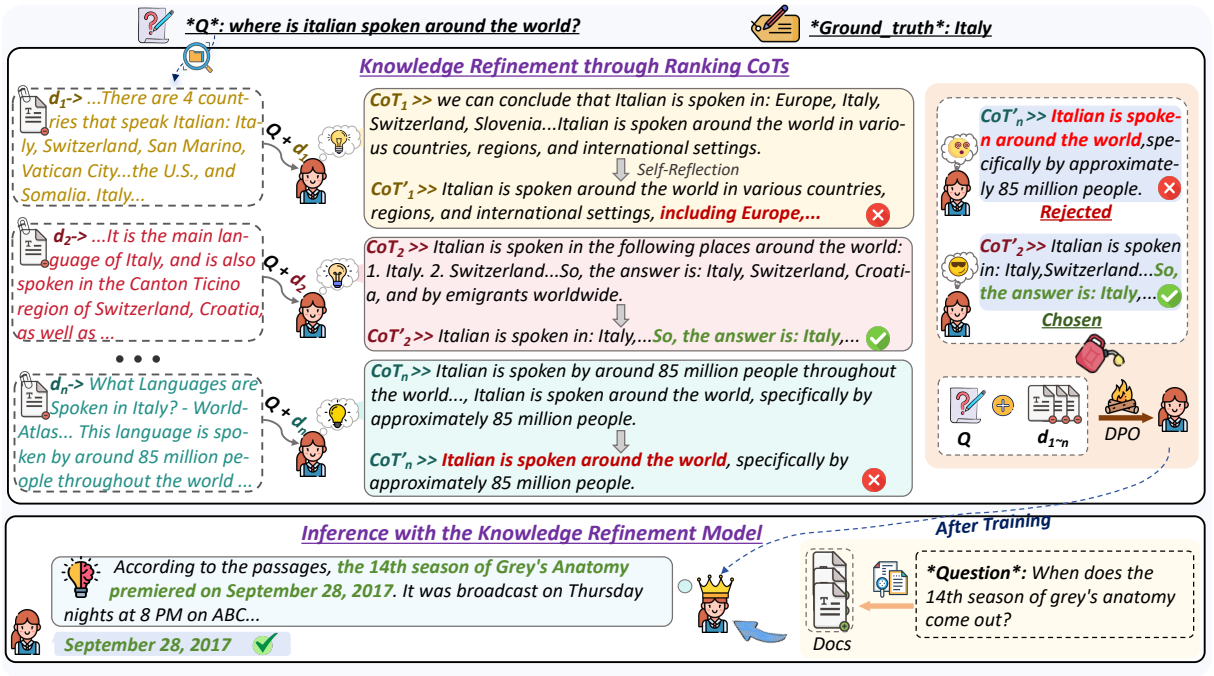


Figure 2: Illustration of RankCoT.

3.2 Knowledge Refinement through Ranking Chain-of-Thoughts

To generate tailored knowledge refinement results y_{KR} for RAG modeling, RankCoT optimizes the LLM ($\mathcal{M}$) by incorporating reranking into the CoT generation process. Furthermore, we introduce a self-reflection method to further refine the CoT results, mitigating the risk of overfitting to undesired CoT patterns during training RankCoT.

3.2.1 Reranking Modeling in CoT Generation

To learn the query-document relevance, we feed each document d_i into the LLM ($\mathcal{M}$) and sample one CoT output $y_{\text{CoT}}(d_i)$:

$$y_{\text{CoT}}(d_i) \sim \mathcal{M}(q, d_i). \tag{8}$$

Next, we gather all generated CoT results from each document in the retrieved document set D to form the candidate CoT set Y_{CoT}:

$$Y_{\text{CoT}} = \{y_{\text{CoT}}(d_1), \ldots, y_{\text{CoT}}(d_n)\}. \tag{9}$$

We treat the CoT result $y_{\text{CoT}} \in Y_{\text{CoT}}$ that contains the ground truth answer as the positive y^{+}_{CoT}, while the result that does not contain the ground truth answer is regarded as the negative y^{-}_{CoT}.

Finally, we can optimize the LLM ($\mathcal{M}$) to assign higher generation probabilities to positive knowledge refinement results y^{+}_{CoT} than the negative ones y^{-}_{CoT}. The training process is implemented with the Direct Preference Optimization (DPO) method (Rafailov et al., 2024):

$$\mathcal{L} = -\mathbb{E}_{(q, y^{+}_{\text{CoT}}, y^{-}_{\text{CoT}}) \sim \mathcal{T}} \Big[\log \sigma \Big(\beta \log \frac{\mathcal{M}(y^{+}_{\text{CoT}} \mid q, D)}{\mathcal{M}^{\text{Ref}}(y^{+}_{\text{CoT}} \mid q, D)} - \beta \log \frac{\mathcal{M}(y^{-}_{\text{CoT}} \mid q, D)}{\mathcal{M}^{\text{Ref}}(y^{-}_{\text{CoT}} \mid q, D)} \Big) \Big], \tag{10}$$

where β is a hyperparameter and σ is the Sigmoid function. $\mathcal{M}^{\text{Ref}}$ is the reference model, which remains frozen during training.

In DPO training, we input all documents D into the model $\mathcal{M}$ and aim to assign a higher probability to the positive knowledge refinement result y^{+}_{CoT}, which is individually generated from one of the retrieved documents. This guides the model $\mathcal{M}$ to rerank the retrieved documents D when generating chain-of-thought as the refinement.

3.2.2 CoT Refinement through Self-Reflection

While these generated CoTs help the RAG model generate more accurate answers, the generated CoT results of LLMs may contain undesired patterns, such as "According to the document" and "the reasoning process is". These training patterns can mislead the LLM ($\mathcal{M}$) to overfit these CoT results during training (Gudibande et al., 2023). To address this problem, RankCoT proposes a self-reflection method to refine the CoT results Y_{CoT}.

Specifically, we first sample the CoT outputs $\tilde{y}_{\text{CoT}}(d_i)$ by feeding the given query q and each

document d_i to the LLM:

$$\tilde{y}_{\text{CoT}}(d_i) \sim \mathcal{M}(\text{Instruct}_{\text{CoT}}, q, d_i), \quad (11)$$

where $\text{Instruct}_{\text{CoT}}$ is used to prompt the LLM to generate a chain-of-thought. Then the CoT result $\tilde{y}_{\text{CoT}}(d_i)$ is refined as $y_{\text{CoT}}(d_i)$ by using the same LLM ($\mathcal{M}$):

$$y_{\text{CoT}}(d_i) = \mathcal{M}(\text{Instruct}_{\text{Ref}}, q, \tilde{y}_{\text{CoT}}(d_i)), \quad (12)$$

where the instruction $\text{Instruct}_{\text{Ref}}$ prompts the LLM ($\mathcal{M}$) to answer the given query q based on the initial CoT result $\tilde{y}_{\text{CoT}}(d_i)$. Such a self-reflection mechanism helps to extract more query-related contents from the initial CoT $\tilde{y}_{\text{CoT}}(d_i)$, producing higher-quality data to optimize LLMs. Finally, we collect the refined CoT results to form $Y_{\text{CoT}}(d_i) = \{y_{\text{CoT}}(d_1), \ldots, y_{\text{CoT}}(d_n)\}$, which is used to train the LLM through DPO (Eq. 10).

4 Experimental Methodology

In this section, we describe the datasets, baselines, evaluation metrics, and implementation details in our experiments. More experimental details are shown in Appendix A.4.

Datasets. In our experiments, we follow previous work (Lin et al., 2024) and utilize the instruction tuning datasets to train and evaluate RAG models. For all datasets and baselines, we use BGE-large (Chen et al., 2024a) to retrieve documents from the MS MARCO V2.1 document collection (Bajaj et al., 2016). We select six datasets for evaluation, including NQ (Kwiatkowski et al., 2019), HotpotQA (Yang et al., 2018), Trivia QA (Joshi et al., 2017), PopQA (Mallen et al., 2023), ASQA (Stelmakh et al., 2022), and MARCO QA (Bajaj et al., 2016), which require models to retrieve factual knowledge or conduct more complex reasoning to help answer the given query. All data statistics are shown in Table 1.

Split	#Query	Tasks
Train	15,444	Open-Domain QA
	8,527	Reasoning
Dev	1,752	Open-Domain QA
	912	Reasoning
Test	20,294	Open-Domain QA

Table 1: Data Statistics.

Baselines. In our experiments, we compare RankCoT with the Vanilla RAG (No Refinement) model and three knowledge refinement models, including Rerank, Summary, and CoT, which are described in Sec. 3.1. For the vanilla RAG model, we follow previous work (Ram et al., 2023) and feed 5 retrieved documents as context to answer the question. For the Rerank model, we prompt the LLM to evaluate the relevance between the query and retrieved documents (Asai et al., 2024a; Li et al., 2024). If the document is relevant to the question, it outputs "YES" and retains the document, otherwise, it outputs "NO" and discards the document. The Summary model and CoT model prompt the LLM to extract query-related knowledge from retrieved documents using summarization and Chain-of-Thought (Wei et al., 2022) formats to conclude query-related knowledge from retrieved documents.

Evaluation Metrics. Following Xu et al. (2024b), we utilize Rouge-L as evaluation metric for MARCO QA task. Following Gao et al. (2023), we utilize String-EM as evaluation metric for ASQA. For other tasks, we use the Accuracy metric for evaluation.

Implementation Details. We implement our RankCoT model using Llama3-8B-Instruct (Touvron et al., 2023) as the backbone model. To construct a training dataset for DPO training, we ask Llama3-8B-Instruct to generate CoTs using 10 retrieved documents independently and use the same model to refine CoTs. During training, we feed 5 relevant documents as external knowledge and ask the LLM to reproduce the refined CoT results. We use LoRA (Hu et al., 2022) method to finetune Llama3-8B-Instruct, with β set to 0.1 and the learning rate set to 2e-5.

For the RAG model, we concatenate the generated CoT with the query to let Llama3-8B-Instruct generate the final answer. In addition, we also use LLMs of different scales, such as MiniCPM3-4B (Hu et al., 2024) and Qwen2.5-14B-Instruct (Yang et al., 2024), to build the RAG model and evaluate the generalization ability of RankCoT.

5 Evaluation Result

In this section, we first evaluate the performance of various RAG methods, followed by ablation studies to examine the impact of self reflection module and different training strategies. We then investigate the characteristics of RankCoT by analyzing the knowledge utilization capabilities of RAG models

Method	NQ (acc)	HotpotQA (acc)	TriviaQA (acc)	PopQA (acc)	ASQA (str-em)	MARCO (rouge)	Avg.
Llama3-8B-Instruct							
No Refinement	45.68	29.43	82.85	35.60	38.79	20.73	42.18
Rerank	46.18	30.30	83.51	36.20	39.92	20.72	42.81
Summary	44.27	28.05	82.09	33.67	37.81	**22.67**	41.32
CoT	45.33	26.36	81.45	34.13	40.25	19.52	41.17
RankCoT	**47.41**	**32.21**	**85.18**	**41.17**	**41.02**	20.84	**44.64**
MiniCPM3-4B							
No Refinement	42.51	24.93	80.91	32.53	24.31	13.55	36.46
RankCoT	**48.78**	**33.13**	**85.20**	**36.87**	**35.85**	**24.59**	**44.07**
Qwen2.5-14B-Instruct							
No Refinement	47.66	29.70	79.49	36.97	**44.73**	18.50	42.84
RankCoT	**49.98**	**33.91**	**86.68**	**44.45**	41.94	**24.62**	**46.93**

Table 2: Overall Performance of RAG System with Different Knowledge Refinement Models. We use Llama3-8B-Instruct as the backbone model for different knowledge refinement models and apply RankCoT to the RAG system, which is implemented with Llama3-8B-Instruct, MiniCPM3-4B, and Qwen2.5-14B-Instruct.

using different knowledge refinement models. We also examine the effectiveness of the refined knowledge generated by RankCoT through answering consistency in Appendix A.3. The case study is conducted in Appendix A.7.

5.1 Overall Performance

This section presents the knowledge refinement performance of different models, as shown in Table 2. Additional baseline comparison results are shown in Appendix A.2.

The evaluation results reveal that these three knowledge refinement models, Rerank, Summary, and CoT, show distinct performance. Specifically, Rerank exhibits a slight improvement, while both Summary and CoT lead to a decrease in RAG performance. This highlights the challenge of effectively refining knowledge for RAG modeling. In contrast, RankCoT demonstrates a 2.5% improvement over vanilla RAG model, indicating its effectiveness in providing more meaningful refinements that help LLMs better answer questions. Furthermore, RankCoT outperforms the Rerank model with a 1.8% improvement and avoids the need to feed raw passages into the LLM twice for knowledge refinement and question answering.

Then we present the performance of RankCoT by applying it to different RAG systems implemented with LLMs of various scales. The results indicate that RankCoT maintains its effectiveness across different RAG configurations, yielding a 7.6% improvement over the MiniCPM3-4B-based RAG model and a 4.1% improvement over the RAG model implemented with Qwen2.5-14B-Instruct. These findings demonstrate that RankCoT has strong generalization ability in knowledge refinement, enabling LLMs of different scales to effectively leverage external knowledge. For more experiments, please see Appendix A.5.

Method	NQ (acc)	HotpotQA (acc)	TriviaQA (acc)	Avg.
Vanilla RAG	45.68	29.43	82.85	52.65
Inference w/ Finetuned QA Model				
Rerank	45.96	29.88	83.04	52.96
Summary	38.46	25.29	76.13	45.63
CoT	41.63	**33.23**	83.24	52.70
SFT Training				
Rerank	47.02	31.21	84.16	54.13
Summary	43.36	28.39	82.26	51.34
CoT	43.43	27.23	82.80	51.15
DPO Training				
Rerank	46.80	30.70	84.70	54.07
Summary	46.63	30.61	83.95	53.73
RankCoT	**47.41**	32.21	**85.18**	**54.97**
w/o Reflect	46.70	30.59	83.82	53.70

Table 3: Ablation Study. Both SFT and DPO methods optimize knowledge refinement models using self-reflection labels. RankCoT w/o Reflect refers to that the RankCoT model is optimized using unrefined CoT.

5.2 Ablation Study

We conduct ablation studies to evaluate the effectiveness of various training strategies.

As shown in Table 3, we first conduct the approach from prior work (Lin et al., 2024), where a fine-tuned QA model is used for knowledge re-

Method	Has-Answer			Miss-Answer			Internal Knowledge		
	NQ	HotpotQA	TriviaQA	NQ	HotpotQA	TriviaQA	NQ	HotpotQA	TriviaQA
LLM w/o RAG	46.41	42.99	80.48	6.51	14.73	46.04	100.0	100.0	100.0
Vanilla RAG	63.45	57.69	88.76	2.41	8.89	19.70	77.30	67.89	90.80
Rerank	64.09	57.09	88.74	3.14	10.87	27.41	78.20	68.28	91.24
Summary	61.40	53.20	87.20	3.02	10.19	25.27	76.89	63.23	89.93
CoT	62.90	51.76	87.53	2.53	7.69	22.91	77.19	59.77	89.38
RankCoT	**65.44**	**58.92**	**90.49**	**4.10**	**13.03**	**30.84**	**80.42**	**70.81**	**92.61**

Table 4: RAG Performance by Using Different Knowledge Refinement Models. We conduct three testing scenarios to evaluate the knowledge usage of RAG systems, including Has-Answer, Miss-Answer and Internal Knowledge.

finement. We then train several knowledge refinement models–Rerank, Summary, and CoT–using the self-reflection mechanism introduced by RankCoT. Specifically, the self-reflection mechanism involves feeding the knowledge refinement results into LLMs to generate self-reflection results based on these inputs. In this setup, SFT methods select self-reflection results containing ground truth answers to train knowledge refinement models, while DPO methods select both positive and negative responses–those that contain ground truth answers and those that do not–for training.

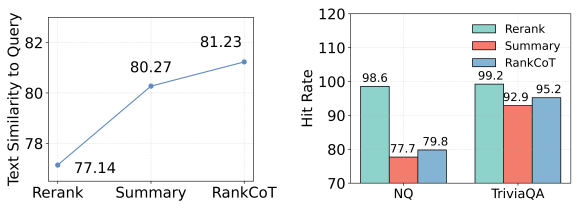
After fine-tuning with QA supervision, the LLM is able to generate knowledge refinement results using different prompts. However, the evaluation results illustrate that these QA models present limited effectiveness compared to vanilla RAG model. We then train the knowledge refinement models using training signals refined by the LLM itself. Among the SFT-based models, Rerank achieves the best performance, illustrating that the reranking signals can be easily learned by LLMs through SFT. Using the DPO training method, both Summary and RankCoT show significant improvements over these knowledge refinement models using the SFT strategy. Furthermore, RankCoT outperforms all knowledge refinement models, demonstrating its effectiveness in producing effective knowledge refinement results to help LLMs generate more accurate answers. By replacing self-refined CoTs with the raw CoT outcomes during DPO training, RankCoT achieves a 1.3% decline, showing the effectiveness of our self-reflection mechanism.

5.3 Knowledge Usage Performance of Different Refinement Models

In this experiment, we evaluate the ability of RankCoT to assist RAG models in leveraging external knowledge to generate final answers. We compare RankCoT with three knowledge refinement models: Rerank, Summary, and CoT.

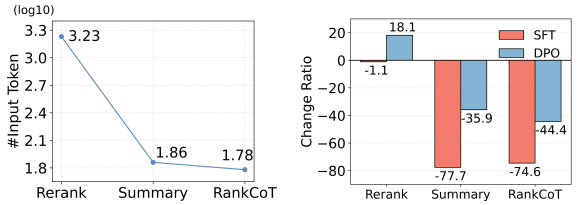
As shown in Table 4, we conduct three testing scenarios to assess the effectiveness of the different knowledge refinement models: Has-Answer, Miss-Answer, and Internal Knowledge. The Has-Answer scenario involves cases where the retrieved documents contain the correct (golden) answer. This scenario evaluates whether the knowledge refinement model can effectively extract key information from these documents to aid the LLM in answering the question. The Miss-Answer scenario, on the other hand, deals with cases where the retrieved documents do not include the golden answer. This scenario further tests the ability of the knowledge refinement models to minimize the impact of retrieved noise. Finally, the Internal Knowledge scenario examines the ability of different knowledge refinement models to handle conflicts between internal and external knowledge.

As shown in the evaluation results, RankCoT outperforms all knowledge refinement models across all datasets in the Has-Answer scenario. This demonstrates the effectiveness of RankCoT in incorporating more query-relevant information from retrieved documents, thereby enhancing the accuracy of the RAG system in this scenario. In the Miss-Answer scenario, the performance of vanilla RAG models significantly drops compared to LLMs w/o RAG, indicating that query-irrelevant documents mislead LLMs into producing incorrect answers. However, the use of different knowledge refinement models mitigates this performance decline. Among these models, RankCoT exhibits the most substantial improvements, demonstrating its effectiveness in filtering out noisy information from retrieval and reducing the misleading information of irrelevant documents. In the internal knowledge scenario, knowledge refinement models–Rerank, Summary, and CoT–perform comparably or even worse than vanilla RAG model, illustrating that existing methods are less effective in addressing the knowledge conflict issue. In contrast, RankCoT outperforms these knowledge refinement models, demonstrating its ability to pro-



(a) Similarity between Query and Refined Knowledge. (b) Hit Rate of Ground Truth Answers.

Figure 3: Quality of Refined Knowledge Generated by Different Models. In Figure 3(a), we first estimate the text similarity between the query and the knowledge refinement results using the BGE model (Chen et al., 2024a). Then, we calculate the hit rate of these knowledge refinement results in Figure 3(b), which evaluates whether the ground truth answers are included in the knowledge refinement results.



(a) Average Length. (b) Length Change Ratio.

Figure 4: The Length of Knowledge Refinement Results Produced by Different Models. We first present the average length of the refinement results in Figure 4(a). Then, the length change ratio relative to vanilla LLMs is illustrated in Figure 4(b).

vide more tailored knowledge refinement results. The RankCoT-produced knowledge refinement results effectively alleviate knowledge conflicts, aiding LLMs in better utilizing both internal and external knowledge.

5.4 Characteristics of the Refined Knowledge Produced by RankCoT

This experiment further explores the characteristics of RankCoT-produced knowledge refinement results by estimating both the quality and length of the refined knowledge. We conduct the experiment on the NQ and TriviaQA datasets, specifically in the Has-answer test scenario, where the retrieved documents contain the ground truth answers.

Refinement Quality. As shown in Figure 3, we evaluate the quality of the knowledge refinement results based on query relevance and the hit rate of the golden answer. Specifically, the similarity scores between the query and the refined knowledge generated by three different knowledge refinement models are presented in Figure 3(a). RankCoT achieves the highest similarity score with the query, demonstrating its effectiveness in retaining more query-related information from the retrieved documents. Furthermore, we show the hit rate of the ground truth answer in Figure 3(b). As indicated by the evaluation results, Rerank achieves the highest hit rates, while Summary performs the worst. This outcome likely stems from the fact that the Rerank model only selects the most relevant document, whereas the Summary model must extract key information, inevitably discarding some of the relevant contents that contain the ground truth answers. Although RankCoT is also a summarization-style knowledge refinement model, it achieves higher hit rates, showing that RankCoT can capture more ground truth answers in its refinement results.

Length of Knowledge Refinement. Subsequently, we present the results of knowledge refinement lengths for different models in Figure 4. As shown in Figure 4(a), the summarization-style knowledge refinement methods, Summary and RankCoT, significantly reduce the length of the refined knowledge compared to the Rerank model. Notably, RankCoT achieves the shortest refinement length, demonstrating its effectiveness in minimizing the consumption of prompt inputs for LLMs (Mu et al., 2023) while helping to reduce inference time (please see Appendix A.6 for related experiments). Additionally, we investigate the length change ratio across different training methods in Figure 4(b). As shown in the results, these SFT methods generally result in shorter knowledge refinement outputs, illustrating that SFT encourages the summarization-style knowledge refinement model to overfit training signals (Li et al., 2024). In contrast, the DPO training method helps these knowledge refinement models produce longer results, facilitating more flexible responses that incorporate more crucial knowledge.

6 Conclusion

This paper proposes RankCoT, a knowledge refinement method that leverages the strengths of both ranking and summarization to effectively refine the knowledge from retrieval results, thereby aiding LLMs in generating more accurate responses. Our experimental studies show that RankCoT can effectively refine external knowledge and balance the utilization of internal and external knowledge. In-depth analysis reveals that the CoT generated by

our method has a high similarity to the query and a high ground truth answer hit rate. In addition, RankCoT shows its effectiveness in minimizing the consumption of prompt inputs for LLMs while helping to reduce inference time.

Limitations

Although RankCoT demonstrates its effectiveness in refining retrieved knowledge for RAG systems, the quality of the refinement is still constrained by the capabilities of LLMs. Specifically, RankCoT is optimized using the DPO method, which relies on LLMs to generate meaningful chosen and rejected pairs during optimization. Therefore, the generation of meaningful preference pairs for optimization still heavily depends on the performance of the LLMs.

Acknowledgments

This work is partly supported by the Natural Science Foundation of China under Grant (No. 62206042 and No. 62137001), CCF-zhipu Large Model Innovation Fund (No. 202403), the Joint Funds of Natural Science Foundation of Liaoning Province (No. 2023-MSBA-081), and the Fundamental Research Funds for the Central Universities under Grant (No. N2416012). This work is also supported by the AI9Stars community.

The authors would like to thank Weiqing Yang for his help in drawing figures during the writing process of the paper, and Sijia Yao for her careful and patient review of the final draft of the paper and her suggestions for revision.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *ArXiv preprint*.

Shourya Aggarwal, Divyanshu Mandowara, Vishwajeet Agrawal, Dinesh Khandelwal, Parag Singla, and Dinesh Garg. 2021. Explanations for CommonsenseQA: New Dataset and Models. In *Proceedings of ACL*, pages 3050–3065.

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of NAACL-HLT*, pages 2357–2367.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024a. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *Proceedings of ICLR*.

Akari Asai, Zexuan Zhong, Danqi Chen, Pang Wei Koh, Luke Zettlemoyer, Hannaneh Hajishirzi, and Wentau Yih. 2024b. Reliable, adaptable, and attributable language models with retrieval. *ArXiv preprint*.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. 2016. Ms marco: A human generated machine reading comprehension dataset. *ArXiv preprint*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of EMNLP*, pages 1533–1544.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Proceedings of NeurIPS*.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024a. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *ArXiv preprint*.

Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. 2024b. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of AAAI*, 16, pages 17754–17762.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *ArXiv preprint*.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, et al. 2023. A survey on in-context learning. *ArXiv preprint*.

Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. 2023. Enabling large language models to generate text with citations. In *Proceedings of EMNLP*, pages 6465–6488.

Yunfan Gao, Yun Xiong, Meng Wang, and Haofen Wang. 2024. Modular rag: Transforming rag systems into lego-like reconfigurable frameworks. *ArXiv preprint*.

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

Arnav Gudibande, Eric Wallace, Charlie Snell, Xinyang Geng, Hao Liu, Pieter Abbeel, Sergey Levine, and Dawn Song. 2023. The false promise of imitating proprietary llms. *ArXiv preprint*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Retrieval augmented language model pre-training. In *Proceedings of ICML*, pages 3929–3938.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. In *Proceedings of ICLR*.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. 2024. Minicpm: Unveiling the potential of small language models with scalable training strategies. *ArXiv preprint*.

Gautier Izacard and Edouard Grave. 2021. Distilling knowledge from reader to retriever for question answering. In *Proceedings of ICLR*.

Gautier Izacard, Patrick S. H. 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*, pages 251:1–251:43.

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 EMNLP*, pages 7969–7992.

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

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of EMNLP*, pages 6769–6781.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, pages 452–466.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of ACL*, pages 7871–7880.

Patrick S. H. 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. 2020b. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proceedings of NeurIPS*.

Xinze Li, Sen Mei, Zhenghao Liu, Yukun Yan, Shuo Wang, Shi Yu, Zheni Zeng, Hao Chen, Ge Yu, Zhiyuan Liu, et al. 2024. Rag-ddr: Optimizing retrieval-augmented generation using differentiable data rewards. *ArXiv preprint*.

Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Richard James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, Luke Zettlemoyer, and Wen tau Yih. 2024. RA-DIT: Retrieval-augmented dual instruction tuning. In *Proceedings of ICLR*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of ACL*, pages 158–167.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, pages 157–173.

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 ACL*, pages 9802–9822.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. 2023. Learning to compress prompts with gist tokens. In *Proceedings of NeurIPS*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. In *Proceedings of NeurIPS*.

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*, pages 1316–1331.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. REPLUG: Retrieval-augmented black-box language models. In *Proceedings of NAACL-HLT*, pages 8371–8384.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation. In *Proceedings of EMNLP Findings*, pages 3784–3803.

Kurt Shuster, Jing Xu, Mojtaba Komeili, Da Ju, Eric Michael Smith, Stephen Roller, Megan Ung, Moya Chen, Kushal Arora, Joshua Lane, et al. 2022. Blenderbot 3: a deployed conversational agent that continually learns to responsibly engage. *ArXiv preprint*.

Ivan Stelmakh, Yi Luan, Bhuwan Dhingra, and Ming-Wei Chang. 2022. ASQA: Factoid questions meet long-form answers. In *Proceedings of EMNLP*, pages 8273–8288.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of NAACL-HLT*, pages 4149–4158.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *ArXiv preprint*.

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 ACL*, pages 10014–10037.

Jesse Vig, Alexander R. Fabbri, Wojciech Kryscinski, Chien-Sheng Wu, and Wenhao Liu. 2022. Exploring neural models for query-focused summarization. In *Findings of the Association for Computational Linguistics: NAACL*, pages 1455–1468.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. 2024. Adaptive chameleon or stubborn sloth: Revealing the behavior of large language models in knowledge conflicts. In *Proceedings of ICLR*.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *Proceedings of ICLR*.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2024a. RECOMP: Improving retrieval-augmented LMs with context compression and selective augmentation. In *Proceedings of ICLR*.

Ruochen Xu, Song Wang, Yang Liu, Shuohang Wang, Yichong Xu, Dan Iter, Pengcheng He, Chenguang Zhu, and Michael Zeng. 2023. Lmgqs: A large-scale dataset for query-focused summarization. In *Proceedings of EMNLP Findings*, pages 14764–14776.

Shicheng Xu, Liang Pang, Mo Yu, Fandong Meng, Huawei Shen, Xueqi Cheng, and Jie Zhou. 2024b. Unsupervised information refinement training of large language models for retrieval-augmented generation. *ArXiv preprint*.

Zhipeng Xu, Zhenghao Liu, Yibin Liu, Chenyan Xiong, Yukun Yan, Shuo Wang, Shi Yu, Zhiyuan Liu, and Ge Yu. 2024c. Activerag: Revealing the treasures of knowledge via active learning. *ArXiv preprint*.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. 2024. Corrective retrieval augmented generation. *ArXiv preprint*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *ArXiv preprint*.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of EMNLP*, pages 2013–2018.

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 EMNLP*, pages 2369–2380.

Wenhao Yu, Hongming Zhang, Xiaoman Pan, Peixin Cao, Kaixin Ma, Jian Li, Hongwei Wang, and Dong Yu. 2024a. Chain-of-note: Enhancing robustness in retrieval-augmented language models. In *Proceedings of EMNLP*, pages 14672–14685.

Yue Yu, Wei Ping, Zihan Liu, Boxin Wang, Jiaxuan You, Chao Zhang, Mohammad Shoeybi, and Bryan Catanzaro. 2024b. Rankrag: Unifying context ranking with retrieval-augmented generation in llms. *ArXiv preprint*.

Zichun Yu, Chenyan Xiong, Shi Yu, and Zhiyuan Liu. 2023. Augmentation-adapted retriever improves generalization of language models as generic plug-in. In *Proceedings of ACL*, pages 2421–2436.

Shuyan Zhou, Uri Alon, Frank F. Xu, Zhengbao Jiang, and Graham Neubig. 2023. Docprompting: Generating code by retrieving the docs. In *Proceedings of ICLR*.

Method	NQ (acc)	HotpotQA (acc)	TriviaQA (acc)	Avg.
Vanilla RAG	45.68	29.43	82.85	52.65
Self-RAG (2024a)	39.90	24.59	78.35	47.61
Recomp (2024a)	40.47	25.36	79.04	48.29
SEGENC (2022)	40.47	24.89	76.73	47.36
RankCoT	**47.41**	**32.21**	**85.18**	**54.93**

Table 5: Overall Performance of More Baselines.

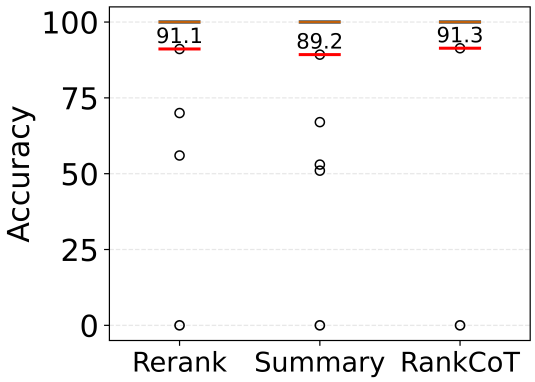


Figure 5: QA Consistency of the RAG Model Using Different Knowledge Refinement Models.

A Appendix

A.1 License

This section summarizes the licenses of the datasets used in our experiments.

All of these datasets under their respective licenses and agreements allow for academic use: Natural Questions (CC-BY-SA-3.0 License); PopQA, Commonsense QA, Wiki QA, MARCO QA, StrategyQA, and Grade School Math 8K (MIT License); Web Questions and HotpotQA (CC-BY-4.0 License); TriviaQA, ASQA, Algebra QA with Rationales, and Math QA (Apache 2.0 License); Explanations for CommonsenseQ (CDLA-Sharing-1.0 License); Yahoo! Answers QA shows its terms of use at website[1].

A.2 Additional Baseline Comparison Results

This section presents the comparison results between RankCoT and several baseline models.

In this experiment, we compare RankCoT with four baselines: vanilla LLM, Self-RAG, Recomp, and SEGENC. Self-RAG (Asai et al., 2024a) optimizes Llama3-8B-Instruct to retrieve documents on demand and ranks them by reflecting the retrieved documents using reflection tokens. SEGENC (Vig et al., 2022) is a Query-Focused Summarization model, initialized from the BART model (Lewis et al., 2020a), which summarizes documents based on a given query. Recomp (Xu et al., 2024a) proposes a method to compress retrieved documents, reducing the computational overhead of language models during inference.

As shown in Table 5, Self-RAG, Recomp, and SEGENC all show performance degradation compared to vanilla RAG. This indicates that ranking, compressing, or summarizing documents inevitably lead to information loss, thereby reducing response accuracy. In contrast, RankCoT not only incorporates advantages of ranking and summarization, but also generates a CoT that preserves as much useful information as possible. This approach reduces the input length for the QA model while improving response accuracy.

A.3 QA Consistency Using Different Knowledge Refinements

This experiment evaluates the QA consistency based on different knowledge refinement results. Since the experiment is conducted in the internal knowledge scenario, all queries can be accurately answered by the RAG model without any external knowledge.

As illustrated in Figure 5, we use the TriviaQA dataset to conduct the experiment. For each query, we input both the query and the refinement results from different models, then sample responses from the RAG model 300 times. The ratio of correct answers is calculated and denoted as Accuracy, which serves to evaluate the QA consistency of the RAG model. A higher accuracy reflects that the knowledge refinement results help the RAG model consistently produce correct answers.

As shown in the evaluation results, RankCoT demonstrates its effectiveness by achieving an average accuracy of approximately 91.3%, outperforming both refinement baselines, Rerank and Summary. The accuracy of the Rerank and Summary methods shows significant variation, indicating that the knowledge refinements produced by both models still contain knowledge that causes the RAG model to lack consistency in its answers. In contrast, after applying RankCoT, the accuracy becomes concentrated at either 0 or 1, demonstrating that it better supports the RAG model in maintaining answer consistency.

[1] https://tensorflow.google.cn/datasets/community_catalog/huggingface/yahoo_answers_qa

Split	Task	Dataset	Metric	Raw	Filtered
Training	Open-Domain QA	Commonsense QA (2019)	Accuracy	4,000	3,037
		Math QA (2019)	Accuracy	4,000	3,509
		Web Questions (2013)	Accuracy	3,578	1,810
		Wiki QA (2015)	Rouge-L	840	840
		Yahoo! Answers QA	Rouge-L	4,000	4,000
		MARCO QA (2016)	Rouge-L	4,000	4,000
	Reasoning	Algebra QA with Rationales (2017)	Accuracy	2,527	2,222
		Explanations for CommonsenseQ (2021)	Accuracy	4,000	3,202
		Grade School Math 8K (2021)	Accuracy	4,000	3,090
		StrategyQA (2021)	Accuracy	1,860	925
Evaluation	Open-Domain QA	Natural Questions (2019)	Accuracy	2,837	-
		HotpotQA (2018)	Accuracy	5,600	-
		TriviaQA (2017)	Accuracy	5,359	-
		PopQA (2023)	Accuracy	3,000	-
		ASQA (2022)	STR-EM	948	-
		MARCO QA (2016)	Rouge-L	3,000	-

Table 6: Data Statistics.

Method	NQ (acc)	HotpotQA (acc)	TriviaQA (acc)	Avg.
MiniCPM3-4B				
Vanilla RAG	42.51	24.93	80.91	49.45
RankCoT (Not Aligned)	48.78	33.13	85.20	55.70
RankCoT (Aligned)	45.68	27.52	79.21	50.80
Qwen2.5-14B-Instruct				
Vanilla RAG	47.66	29.70	79.49	52.28
RankCoT (Not Aligned)	49.98	33.91	86.68	56.86
RankCoT (Aligned)	52.17	37.34	86.49	58.67

Table 7: Overall Performance of the RankCoT Model with and without Backbone Model Alignment in the RAG System. "Not aligned" refers to using Llama3-8B-Instruct as the backbone model while applying RankCoT to RAG systems built with MiniCPM3-4B or Qwen2.5-14B-Instruct. "Aligned" refers to using consistent backbone models, either MiniCPM3-4B or Qwen2.5-14B-Instruct, throughout the RankCoT and RAG components.

A.4 Additional Experimental Details

In this subsection, we first describe the process of constructing the training data and then show the prompt templates used in our experiments.

Data Preprocessing for RankCoT. The quantities of our training and evaluation data, along with the corresponding evaluation metrics, are presented in Table 6. The "Filtered" column indicates the number of training samples used for DPO training.

During RankCoT training, we collect ten datasets, obtain 32,805 samples, and process them as described in Section 3.2. Since the generated CoT can either be fully correct or incorrect, we cannot form preference data pairs from these generated CoT candidates. Thus, we filter out such samples and divide the remaining ones into training and validation datasets with a 9:1 ratio.

Prompt Templates. The prompts used for CoT generation ($\text{Instruct}_{\text{CoT}}$) are shown in Figure 6(a). The prompt used for question answering is illustrated in Figure 7. Additionally, the prompts used for CoT refinement ($\text{Instruct}_{\text{Ref}}$) are shown in Figure 6(b). Finally, Figure 8 presents the prompts used for implementing baselines.

A.5 Evaluating RankCoT under Aligned Parameter Scales in RAG Systems

Although RankCoT (Llama3-8B-Instruct) can be applied to various RAG systems implemented with LLMs of different scales and demonstrates its effectiveness, the performance gains may diminish when larger-scale LLMs are used as the generation models in RAG systems. This is because larger LLMs inherently possess stronger knowledge refinement capabilities. To assess the impact of model scale on RankCoT, we conduct experiments by aligning the parameter scales of both the RAG models and the RankCoT model.

As shown in Table 7, we compare two variants of RankCoT: one without alignment and one with alignment. Both variants implement RankCoT using either Llama3-8B-Instruct or the same LLMs employed by the corresponding RAG models. The results demonstrate that when Qwen2.5-14B-Instruct is used consistently across components, the knowledge refinement process is more effective, leading to improved answer accuracy. In contrast, aligning all components with MiniCPM3-4B results in lower overall performance compared to the unaligned setting. This outcome is expected, as the unaligned setting leverages Llama3-8B-Instruct for knowledge refinement, which benefits from a larger model size and stronger refinement capabilities.

Method	NQ		HotpotQA		TriviaQA		Number of
	Refine	Gen.	Refine	Gen.	Refine	Gen.	LLM Calls
Vanilla RAG	-	54s	-	57s	-	58s	1
Rerank	212s	56s	638s	58s	280s	56s	2
Summary	101s	11s	101s	11s	89s	9s	2
CoT	106s	12s	111s	13s	96s	10s	2
RankCoT	82s	10s	84s	10s	74s	8s	2

Table 8: Inference Time of RAG with Different Knowledge Refinement Models. A total of 500 samples are drawn from each of the NQ, HotpotQA, and TriviaQA datasets. All experiments are conducted using the Llama3-8B-Instruct model with the vLLM inference engine and a batch size of 60. "Refine" denotes the time spent on knowledge extraction, while "Gen." refers to the time taken for answer generation.

A.6 Inference Time of RAG with Different Knowledge Refinement Models

This section reports the inference time of various knowledge refinement models across different datasets, as summarized in Table 8.

During inference, both RankCoT and the baseline refinement models make the same number of calls to the LLM: once for refining the retrieved knowledge and once for answering the question based on the refined knowledge. Compared to vanilla RAG, most of the additional time introduced by these models is spent on the knowledge refinement step. Among all evaluated methods, RankCoT achieves the lowest inference latency. As discussed in Section 5.4, RankCoT generates the shortest knowledge refinement results, which largely accounts for its superior inference efficiency.

A.7 Case Study

In Table 9, we present a case study to illustrate the effectiveness of the RankCoT model. For the given query, it asks about "Australia's location in the world and region". And the retrieved documents contain both related and unrelated information about the geographical location of Australia.

The summarization method captures some geographical information about Australia from the retrieved documents, such as "being in the Southern Hemisphere and located between the Indian and Pacific Oceans". However, these descriptions offer only a broad geographical scope rather than directly answering the query about the regional location of Australia. Consequently, the LLM is misled by the ambiguous information in the summarized documents and generates inaccurate answers. Moreover, the summarization contains some irrelevant information, such as "7,682,300 square kilometers" and "smallest continent and the sixth-largest country".

In contrast, RankCoT refines the retrieved documents in a question-answering based summarization manner by generating a Chain-of-Thought (CoT). These CoT results are constructed by sequentially integrating information from different retrieved documents, while ranking and prioritizing the most query-relevant knowledge. Rather than directly summarizing keypoint information from retrieved documents, RankCoT identifies crucial geographical attributes in each document and organizes them in a structured reasoning result. At the start of the CoT, broad geographical attributes, such as "Southern Hemisphere" and "between the Indian and Pacific Oceans" are strengthened, as they appear consistently across documents. More specific regional information, such as "Oceania", is ranked higher in the reasoning process, ensuring that the final CoT provides the most accurate regional classification. This demonstrates that RankCoT is not merely a direct extraction or summary of retrieved documents, but rather a refined reasoning chain that aligns closely with the query.

Query: What is australia's location in the world and region? **Ground Truth: Oceania**

Documents:

(1) ... Australia is both a continent and a country. It is located to the south of Indonesia and Papua New Guinea, in the southeast part of the Indian Ocean. New Zealand lies to its east, and most of the Indian Ocean lies to its west. ... Australia is the world's smallest continent, though it is also the sixth-largest country in the world. The total land area of Australia is 7,682,300 sq. km.

(2) ... Australia is the smallest continent and the largest country in Oceania located between the Indian Ocean and Pacific Ocean in the Southern hemisphere. ... Australia is separated from Papua New Guinea by the Coral Sea and Torres Strait to the northeast; from Indonesia by the Timor and Arafura seas to the northwest; from Coral Sea Islands Territory by the Great Barrier Reef; from New Zealand by Tasman Sea to the southeast;

(3) ... Australia is located south of the equator and is bordered by the Indian Ocean and the Pacific Ocean.

...

Summary:

Summary: Here is a concise summary: Australia is located in the Southern Hemisphere, between the Indian Ocean and the Pacific Ocean. It is the world's smallest continent and the sixth-largest country, covering an area of 7,682,300 square kilometers. Australia is surrounded by the Indian Ocean to the west, the Timor, Arafura, and Coral Seas to the north, the Pacific Ocean to the east, and the Tasman Sea and Southern Ocean to the south. The country is made up of six states (Western Australia, South Australia, Victoria, Tasmania, New South Wales, and Queensland) and two territories (the Northern Territory and the Australian Capital Territory)

Answer: Australia is located in the Southern Hemisphere, between the Indian Ocean and the Pacific Ocean. (✗)

RankCoT:

Chain-of-Thought: According to the passages, Australia is:

1. An island continent located in the Southern Hemisphere.
2. Situated between the Indian Ocean and the Pacific Ocean.
3. Located south of the equator.
4. Surrounded by the Indian Ocean to the west, the Timor, Arafura, and Coral Seas to the north, the Pacific Ocean to the east, and the Tasman Sea and Southern Ocean to the south.
5. An island continent in Oceania, which is a region that includes Australia, New Zealand, and many Pacific island nations.

So, Australia's location in the world is in the Southern Hemisphere, in the region of Oceania.

Answer: Australia is an island continent located in the Southern Hemisphere, in the region of Oceania. (✓)

Table 9: Case Study. We randomly sample one case from the NQ dataset to show the knowledge refinement result. Different colors are used to annotate the key information from the retrieved knowledge retained by different knowledge refinement models: Pink for Summary, Blue for RankCoT. And we also highlight the key points that can help answer the query in Purple.

Prompts for CoT Generation

```
[TASK]: Web Questions, Wiki QA, Yahoo! Answers QA, MARCO QA ,
Grade School Math 8K, StrategyQA, Natural Questions,
HotpotQA, TriviaQA, PopQA, ASQA
[Instruction]: Passage:{passage}
Based on these passages, answer the question below.
Question:{question}
Let's think step by step.

[TASK]: Commonsense QA, Math QA, Algebra QA with Rationales,
Explanations for CommonsenseQ
[Instruction]: Passage:{passage}
Based on these passages, please answer the multiple choice
question below.
Question:{question}
Let's think step by step.
```

(a) CoT Generation.

Prompts for CoT Refinement

```
[TASK]: Web Questions, Wiki QA, Yahoo! Answers QA, MARCO QA ,
Grade School Math 8K, StrategyQA
[Instruction]: Task Description:
1.  Read the given question and related chain of thought to
    gather relevant information.
2. The content of the chain of thought is the thinking
process that may be used to answer the question.
3. If the chain of thought don't work, please answer the
question based on your own knowledge.
4. Give a short answer to a given question.
Question:{question}
Chain of Thought:{CoT}

[TASK]: Commonsense QA, Math QA, Algebra QA with Rationales,
Explanations for CommonsenseQ
[Instruction]: Task Description:
1.  Read the given question and related chain of thought to
    gather relevant information.
2. The content of the chain of thought is the thinking
process that may be used to answer the question.
3. If the chain of thought don't work, please answer the
question based on your own knowledge.
4.Output only the choice between ***** and ***** at
first,then give a short answer to the mutiple choice
question.
Question:{question}
Chain of Thought:{CoT}
```

(b) CoT Refinement.

Figure 6: Prompt Templates Used in RankCoT.

Prompts for Question Answering

[TASK]: Natural Questions, HotpotQA, TriviaQA, PopQA
[Instruction]: Task Description:
1. Read the given question and related chain of thought to gather relevant information.
2. The content of the chain of thought is the thinking process that may be used to answer the question.
3. If the chain of thought don't work, please answer the question based on your own knowledge.
4. Please answer the question and only output the answer.
Question:{question}
Chain of Thought:{COT}

[TASK]: ASQA
[Instruction]: Task Description:
1. Read the given question and related chain of thought to gather relevant information.
2. The content of the chain of thought is the thinking process that may be used to answer the question.
3. If the chain of thought don't work, please answer the question based on your own knowledge.
4. Answer the following question. The question may be ambiguous and have multiple correct answers, and in that case, you have to provide a long-form answer including all correct answers.
Question:{question}
Chain of Thought:{COT}

[TASK]: MARCO QA
[Instruction]: Task Description:
1. Read the given question and related chain of thought to gather relevant information.
2. The content of the chain of thought is the thinking process that may be used to answer the question.
3. If the chain of thought don't work, please answer the question based on your own knowledge.
4. Give a short answer to a given question.
Question:{question}
Chain of Thought:{COT}

Figure 7: Prompt Templates Used for Question Answering.

Prompts for Baselines

[MODEL]: Vanilla RAG
[TASK]: All except ASQA
[Instruction]: Background:{Passages}
Question:{Question}Answer:

[MODEL]: Vanilla RAG
[TASK]: ASQA
[Instruction]: Background:{Passages}
Answer the following question. The question may be ambiguous and have multiple correct answers, and in that case, you have to provide a long-form answer including all correct answers.
Question:{Question}Answer:

[MODEL]: Rerank
[TASK]: All
[Instruction]: Given the following question and context, return YES if the context is relevant to the question and NO if it isn't.
Question: {question}
Context:{context}
Relevant (YES / NO):

[MODEL]: Summary Generation
[TASK]: All
[Instruction]: You are tasked with summarizing the context in a concise manner, focusing on the key information relevant to answering the given question. Question: {question}
Context:{context}
Provide a concise summary that captures the main points and relevant details from the context to address the question:

[MODEL]: Summary Question Answering
[TASK]: Natural Questions, HotpotQA, TriviaQA, PopQA
[Instruction]: Task Description:
1. Read the given question and related summary to gather relevant information.
2. If the summary don't work, please answer the question based on your own knowledge.
3. Please answer the question and only output the answer.
Question:{question}
Summary:{summary}

[MODEL]: Summary Question Answering
[TASK]: ASQA
[Instruction]: Task Description:
1. Read the given question and related summary to gather relevant information.
2. If the summary don't work, please answer the question based on your own knowledge.
3. Answer the following question. The question may be ambiguous and have multiple correct answers, and in that case, you have to provide a long-form answer including all correct answers.
Question:{question}
Summary:{summary}

[MODEL]: Summary Question Answering
[TASK]: MARCO QA
[Instruction]: Task Description:
1. Read the given question and related summary to gather relevant information.
2. If the summary don't work, please answer the question based on your own knowledge.
3. Give a short answer to a given question Question:{question}
Summary:{summary}

Figure 8: Prompt Templates Used for Implementing Baselines.