A Simple yet Effective Retrieval-Augmented Generation Framework for the Meta KDD Cup 2024

Liyang He*
University of Science and Technology of China
Hefei, China
heliyang@mail.ustc.edu.cn

Rui Li*
University of Science and Technology of China
Hefei, China
ruili2000@mail.ustc.edu.cn

Shuanghong Shen
University of Science and Technology of China
Hefei, China
closer@mail.ustc.edu.cn

Junyu Lu
University of Science and Technology of China & Institute of Artificial Intelligence Comprehensive National Science Center
Hefei, China
lujunyu@mail.ustc.edu.cn

Linbo Zhu
University of Science and Technology of China & Institute of Artificial Intelligence Comprehensive National Science Center
Hefei, China
lbzhu@iai.ustc.edu.cn

Yu Su
Hefei Normal University & Institute of Artificial Intelligence Comprehensive National Science Center
Hefei, China
yusu@hfnu.edu.cn

Zhenya Huang
University of Science and Technology of China
Hefei, China
huangzhy@ustc.edu.cn

Abstract

This paper describes our team's solution for the Meta KDD CUP 2024: CRAG Comprehensive RAG Benchmark Challenge Task 1 (Retrieval Summarization). The task involved building a retrieval-augmented generation framework and testing it on the CRAG benchmark. Our solution is a pipeline encompassing data processing, retrieval, and chain-of-thought-based generation. In this process, we also experimented with popular existing RAG techniques. Our framework ultimately won the Simple_w_condition, Set, and Aggregation questions in Task 1.

CCS Concepts

• **Information systems** → **Information retrieval**.

Keywords

retrieval-augmented generation, retrieval, ranking, large language model

1 Introduction

Retrieval-augmented generation (RAG) has proven to be an effective solution for addressing LLM hallucinations and knowledge update challenges in both academic research and industry [4, 10]. A basic RAG framework involves: (1) retrieving relevant information from a database based on a query, and (2) combining this information with some prompts, inputting them into an LLM to generate the final result. Currently, researchers are proposing various RAG-related techniques for this basic scenario, such as query fusion [6], transforming queries into documents for retrieval [3], and incorporating reflection mechanisms in the retrieval process [1, 9]. However, which method is more effective in practical applications remains an open question.

To address this problem, Meta has organized the KDD Cup 2024 challenge and introduced the Comprehensive RAG Benchmark (CRAG) [8]. This initiative aims to provide a robust benchmark with clear metrics and evaluation protocols, facilitating rigorous assessment of RAG systems, driving innovations, and advancing solutions.

The challenge comprises three tasks. Our team focuses on Task 1 and has won the Simple_w_condition, Set, and Aggregation questions in Task 1. We will introduce our specific implementation methods and some of the experiments we conducted in the following sections.

2 Dataset & Task

2.1 Dataset

CRAG [8] contains 4,409 pairs of question-answer samples, which encompass five domains: Finance, Sports, Music, Movies, and Open-domain Encyclopedia. These questions exhibit various types, including simple factual questions, conditional questions, comparative questions, aggregation questions, multi-hop questions, set queries, post-processing questions, and false premise questions. Additionally, the questions vary in dynamics, ranging from real-time

*Both authors contributed equally to this research

KDD'24, August 25–29, 2024, Barcelona, Spain

questions and rapidly changing questions to slowly changing questions and static questions. Table 1 and Table 2 represent a question sample in the CRAG dataset.

query_time	03/19/2024, 23:23:54 PT
query	what's the date of birth of ben wolfinsohn...
search_results	see Table 2
domain	movie
question_type	simple
static_or_dynamic	static
answer	1973-04-01

Table 1: A question sample from the CRAG dataset.

page_name	Ben Wolfinsohn (visual voices guide)...
page_url	https://www.behindthevoiceactors...
page_snippet	Known for voicing Ben. View...
page_result	<!DOCTYPE html>\n<html lang=\ ...
page_last_modified	None

Table 2: Case of search results.

2.2 Task

Meta KDD CUP 2024 [1] includes three tasks, where our team focuses on Task 1: Retrieval Summarization. In this task, each question is given five web pages. These web pages may be relevant or irrelevant. Our goal is to construct a Retrieval-Augmented Generation (RAG) framework to extract potentially useful information from these web pages and assist in answering the question.

3 Methodlogy

Our final RAG framework is a simple yet efficient model, as depicted in Figure 1. However, during the construction of this framework, we experimented with many methods. We will detail some of our attempts according to different stages in the following sections.

3.1 Data Processing

3.1.1 Web Pages. Since the reference documents originate from web pages, they contain various HTML tags. Therefore, it is crucial to first convert this structured format into natural language text for improved retrieval and comprehension by the large model. We experimented with multiple HTML parsing methods and ultimately selected BeautifulSoup from the bs4 library.

3.1.2 Query. Query augmentation is a crucial step in modern information retrieval. Currently, various query enhancement methods have been proposed in the RAG field. We have experimented with these methods, including:

- **Hyde** [3]: Enhancing the original query by generating hypothetical documents. It has been proven to significantly improve the zero-shot performance of the retrieval step.
- **Step-Back prompting** [11]: Abstracting the original question to obtain queries targeting high-level concepts and fundamental principles. This method has been validated to be relatively effective in time-sensitive QA tasks and multi-hop reasoning tasks. The final prompt is shown in Figure 2.
- **Query rewriting** [5]: Due to the lack of detail in the original question provided by the user, the retriever finds it difficult to understand. Query rewriting aims to enrich the query information. We utilized a zero short llama3-8B to complete the query rewrite.
- **Query fusion** [6]: This method generates multiple queries from the original query through a large language model. It then executes these search queries in parallel and merges the retrieved results. This approach is very useful when a question may depend on multiple sub-questions. We utilized a zero short llama3-8B to complete the query fusion.

In this competition, some of the query enhancement strategies were beneficial to the results, but due to the increased time overhead of these enhancement strategies and the diversity of questions in the CRAG, we did not adopt the above query enhancement strategies in our final solution. Detailed results can be found in section 4.3.

3.2 Retreival

The retrieval process is a critical component of the RAG framework. A poor retriever will result in incorrect information input, affecting the final output of the LLM. We experimented with various methods in the retrieval process, including:

- **An individual recall model**: After splitting the web pages into chunks of 512 in length, we used the bge-m3 model [2] to map these chunks into vector representations. We then performed a brute-force search to calculate the relevance between the query and the web page fragments, selecting the top 10 results as evidence for the LLM input.
- **An individual ranking model**: Considering the characteristics of Task 1, where a small range of web page candidates is already provided, we can directly use a ranking model without worrying about timeout issues. We employed the bge-m3-reranker [2] to calculate the similarity between the query and the web page fragments. We selected the top 10 results with a similarity greater than 0 as evidence for the LLM input.
- **Recall-then-ranking**: Combining the aforementioned approaches, we first use the beg-m3 model to recall 10 candidate sets. Then, we employ the beg-m3-reranker to rank these 10 candidates, selecting the top 5 results.
- **Hierarchical Index**: The previous methods utilized fixed-length chunks, which may limit the length of retrieval results and hinder the correct retrieval of evidence due to semantic segmentation. Therefore, we also experimented with the Hierarchical Index approach, setting chunk sizes to 256, 128, and 64 to construct a hierarchical indexing structure.

We experimented with various approaches and their combinations, discovering some insightful results. First, an individual ranking model outperformed both an individual recall model and the recall-then-ranking method. However, this was only feasible with the llama3-8B model. When we scaled up to the 70B model, it often

[1] https://www.aicrowd.com/challenges/meta-comprehensive-rag-benchmark-kdd-cup-2024

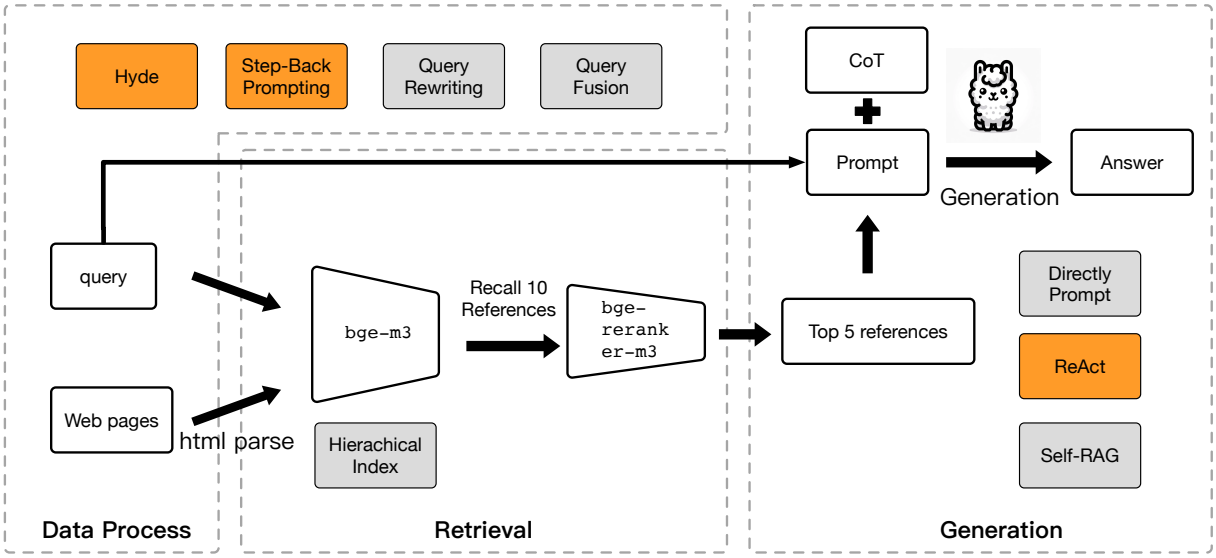

Figure 1: The pipeline of our solution consists of three stages: data processing, retrieval, and generation. We designed a simple yet effective framework while experimenting with various approaches. The orange blocks indicate methods that improve performance for certain question types but often incur additional time overhead. The gray blocks represent methods with negligible improvement or those that may lead to negative outcomes.

resulted in memory overflow or inference timeout issues. Consequently, we ultimately chose the recall-then-ranking method. Additionally, employing a Hierarchical Index did not yield significant performance improvements and sometimes even led to considerable performance degradation, indicating that the Hierarchical Index is not a universally effective method for enhancing performance.

3.3 Generation

We experimented with the following methods to prompt or guide the language model to answer questions, including:

- **Directly prompt**: By adding the retrieved content to the context and prompting the language model to output as little content as possible (with a competition limit of a maximum of 75 words), and responding with "I don't know" to questions for which no relevant information was retrieved, our chosen prompt is shown in Figure 3.
- **Chain-of-thought prompting** [7]: Chain-of-thought prompts have been proven to greatly enhance language models' reasoning ability. In competitions, due to the need to handle complex questions, it is crucial to enhance the reasoning ability of LLMs. Here, we prompt the language model to output answers step by step and extract the final answer. Our chosen prompt is shown in Figure 4.
- **ReAct** [9]: ReAct uses LLM to generate reasoning trajectories and actions for specific tasks in an interleaved manner. For this task, we created a web retrieval tool that allows LLM to make tool calls through prompts. The final prompt is shown in Figure 5.
- **Self-RAG** [1]: Self-RAG is an adaptive rag technology that allows LLM to combine its own knowledge and references to respond to questions, and it will evaluate the relevance of the retrieval results to the question and whether they can help with the response. Here, we directly use the checkpoint [2] provided in the original text to conduct experiments in the competition.

Step-back prompt

```
You are a helpful assistant. Your task is to
step back and paraphrase a question to a more
generic step-back question, which is easier to
answer. Here are a few examples:

Query: Who was the spouse of Anna Karina from
1968 to 1974?
Step Back Query: Who were the spouses of Anna
Karina?

Query: Estella Leopold went to whichschool
between Aug 1954and Nov 1954?
Step Back Query: What was Estella
Leopold'seducation history?
```

Figure 2: Step-Back prompt.

After weighing performance and time, we ultimately used chain-of-thought prompting as the final submission version for the competition. Detailed results can be found in section 4.4.

4 Experiments

In this section, we analyze the effectiveness of our model, especially for some methods that could not be used in the challenge due to timeout issues. In the following experiments, unless otherwise

[2] https://huggingface.co/selfrag

Directly prompt

You are a helpful assistant. For the given questions, please reply with as few words as possible.
Note:
- The user's question may contain factual errors, in which case you must reply `invalid question.`
- If you don't know the answer, simply respond `I don't know.`

Figure 3: Direct prompt.

COT prompt

You are a helpful assistant. For the given question and multiple references from web pages, think step by step, then provide the final answer.
Current Date: {query_time}
Note:
- The user's question may contain factual errors (a false premise that can be inferred from the references), in which case you MUST reply `invalid question.`
- If you don't know the answer, you MUST respond with `I don't know.`
- For your final answer, please use as few words as possible.
- Your output format needs to meet the requirements: First, start with `## Thought\n` and then output the thought process regarding the user's question. After you finish thinking, you MUST reply with the final answer on the last line, starting with `## Final Answer\n` and using as few words as possible.

Figure 4: Chain-of-thought prompt.

specified, we used a 4-bit GPTQ quantized Llama3-70B model. Using data provided by the Meta KDD Cup 2024, we performed stratified sampling for each type of problem, obtaining 371 samples as the validation set, with the remaining samples serving as the training set.

4.1 Metrics

Meta KDD CUP 2024 employs both automated (auto-eval) and human (human-eval) evaluations. In this section, we adopt the same automatic evaluation method as the competition to analyze our approach. Specifically, automatic evaluation employs rule-based matching and GPT-4 assessment to check answer correctness. It assigns three scores: correct (1 point), missing (0 points), and incorrect (-1 point). Missing denotes that the answer does not provide the requested information and should use the standard response "I don't know." as the answer. All false premise questions should

ReAct prompt

You are a helpful and honest assistant. You are given a quesition and references which may or may not help answer the question. Your goal is to answer the question in as few words as possible.

Tools
You can utilize web_search tools to gather the necessary information for answering questions. Moreover, you may need to break down complex question into sub-questions and invoke tools for each sub-question to gather relevant information.
The detailed information of the web search tool is as follows:
{tool_desc}

...

Figure 5: React prompt.

be answered with the standard response "invalid question." The ground truth is the answer that was correct at the time the question was posed and data were collected.

4.2 Main Result

As shown in Table 3, we provide the results of our final pipeline evaluated locally. From the experimental results, it can be observed that we performed relatively well in solving Simple_with_condition, Set, and Aggregation questions. The multi-hop questions are the most challenging to answer correctly.

Question Type	Accuracy	Missing	Hallucination
simple	0.31	0.51	0.18
simple_w_condition	0.34	0.53	0.13
post-processing	0.27	0.55	0.18
aggregation	0.35	0.34	0.31
false_premise	0.35	0.33	0.33
set	0.38	0.35	0.26
multi-hop	0.13	0.58	0.29
comparison	0.21	0.6	0.19

Table 3: Results of all types of questions in our pipeline.

4.3 Analysis on Query Augmentation

As previously mentioned, we employed various methods for query augmentation. Utilizing the Hyde method, we observed a notable improvement for set and false_premise type questions, with the accuracy for set questions at 0.41 (+0.03) and for false_premise questions at 0.38 (+0.03). The Step-Back prompting method demonstrated some enhancement for simple_w_condition type questions (Accuracy: 0.43 (+0.09), Missing: 0.4 (-0.13), Hallucination: 0.17 (+0.04)). However, these methods invariably increased the time overhead

for this challenge. Additionally, we found that query rewriting and query fusion did not result in significant improvements, and in some cases, even led to a decline in overall performance.

4.4 Analysis on Generation Pipeline

In this competition, we employed various prompts and generation pipelines. The overall score using the original prompt was 0.11 lower compared to the CoT prompt, and the final score using ReAct was 0.1 lower than the CoT prompt, despite achieving a higher multi-hop accuracy of 0.21. However, there were significant timeout issues with ReAct. We just got the Self-RAG with llama-2-7b and llama-2-13b versions during the competition, thus its performance was relatively poor.

5 Conclusion

The Meta KDD Cup 2024 competition was a unique challenge due to the diversity of problems and the efficiency requirements. We have presented how we addressed Task 1. We explored numerous relevant techniques for this task; however, due to gaps between research and practical application, we found that some techniques could not be applied effectively or efficiently. Ultimately, we proposed an elegant yet effective Retrieval-Augmented Generation Framework to tackle this challenge, achieving victory in the Simple_w_condition, Set, and Aggregation questions in Task 1.

References

[1] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. arXiv:2310.11511 [cs.CL] https://arxiv.org/abs/2310.11511

[2] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216* (2024).

[3] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2022. Precise Zero-Shot Dense Retrieval without Relevance Labels. arXiv:2212.10496 [cs.IR] https://arxiv.org/abs/2212.10496

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

[5] Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query Rewriting for Retrieval-Augmented Large Language Models. arXiv:2305.14283 [cs.CL] https://arxiv.org/abs/2305.14283

[6] Zackary Rackauckas. 2024. Rag-Fusion: A New Take on Retrieval Augmented Generation. *International Journal on Natural Language Computing* 13, 1 (Feb. 2024), 37–47. https://doi.org/10.5121/ijnlc.2024.13103

[7] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv:2201.11903 [cs.CL] https://arxiv.org/abs/2201.11903

[8] Xiao Yang, Kai Sun, Hao Xin, Yushi Sun, Nikita Bhalla, Xiangsen Chen, Sajal Choudhary, Rongze Daniel Gui, Ziran Will Jiang, Ziyu Jiang, Lingkun Kong, Brian Moran, Jiaqi Wang, Yifan Ethan Xu, An Yan, Chenyu Yang, Eting Yuan, Hanwen Zha, Nan Tang, Lei Chen, Nicolas Scheffer, Yue Liu, Nirav Shah, Rakesh Wanga, Anuj Kumar, Wen tau Yih, and Xin Luna Dong. 2024. CRAG – Comprehensive RAG Benchmark. arXiv:2406.04744 [cs.CL] https://arxiv.org/abs/2406.04744

[9] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. arXiv:2210.03629 [cs.CL] https://arxiv.org/abs/2210.03629

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

[11] Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H. Chi, Quoc V Le, and Denny Zhou. 2024. Take a Step Back: Evoking Reasoning via Abstraction in Large Language Models. arXiv:2310.06117 [cs.LG] https://arxiv.org/abs/2310.06117