Revisiting the Solution of Meta KDD Cup 2024: CRAG

Jie Ouyang
State Key Laboratory of Cognitive
Intelligence, University of Science and
Technology of China
Hefei, Anhui, China
ouyang_jie@mail.ustc.edu.cn

Yucong Luo
State Key Laboratory of Cognitive
Intelligence, University of Science and
Technology of China
Hefei, Anhui, China
prime666@mail.ustc.edu.cn

Mingyue Cheng*
State Key Laboratory of Cognitive
Intelligence, University of Science and
Technology of China
Hefei, Anhui, China
mycheng@ustc.edu.cn

Daoyu Wang
State Key Laboratory of Cognitive
Intelligence, University of Science and
Technology of China
Hefei, Anhui, China
wdy030428@mail.ustc.edu.cn

Shuo Yu
State Key Laboratory of Cognitive
Intelligence, University of Science and
Technology of China
Hefei, Anhui, China
yu12345@mail.ustc.edu.cn

Qi Liu
State Key Laboratory of Cognitive
Intelligence, University of Science and
Technology of China
Hefei, Anhui, China
qiliuql@ustc.edu.cn

Enhong Chen
State Key Laboratory of Cognitive
Intelligence, University of Science and
Technology of China
Hefei, Anhui, China
cheneh@ustc.edu.cn

Abstract

This paper presents the solution of our team APEX in the Meta KDD CUP 2024: CRAG Comprehensive RAG Benchmark Challenge. The CRAG benchmark addresses the limitations of existing QA benchmarks in evaluating the diverse and dynamic challenges faced by Retrieval-Augmented Generation (RAG) systems. It provides a more comprehensive assessment of RAG performance and contributes to advancing research in this field. We propose a routing-based domain and dynamic adaptive RAG pipeline, which performs specific processing for the diverse and dynamic nature of the question in all three stages: retrieval, augmentation, and generation. Our method achieved superior performance on CRAG and ranked 2nd for Task 2&3 on the final competition leaderboard. Our implementation is available at this link: https://github.com/USTCAGI/CRAG-in-KDD-Cup2024.

CCS Concepts

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

Keywords

Retrieval-Augmented Generation, Large Language Model

1 Introduction

Large Language Models (LLMs) have revolutionized the landscape of Natural Language Processing (NLP) tasks [5, 8, 10], particularly in question answering (QA). Despite advances in LLMs, hallucination remains a significant challenge, particularly for dynamic facts and information about less prominent entities.

Retrieval-Augmented Generation (RAG) [9] has recently emerged as a promising solution to mitigate LLMs' knowledge deficiencies.

---

*Mingyue Cheng is the corresponding author.

Given a question, a RAG system queries external sources to retrieve relevant information and subsequently provides grounded answers. Despite its potential, RAG continues to face numerous challenges, including the selection of the most relevant information, the reduction of question answering latency, and the synthesis of information to address complex questions.

To bridge this gap, Meta introduced the Comprehensive RAG Benchmark (CRAG) [13], a factual question answering benchmark of 4,409 question-answer pairs and Mock APIs to simulate web and Knowledge Graph (KG) search, and hosted the KDD CUP 2024 Challenge.

1.1 Dataset Description

The CRAG contains two parts of data: the QA pairs and the content for retrieval.

QA pairs. The CRAG dataset contains a rich set of 4,409 QA pairs covering five domains: finance, sports, music, movie, and open domain, and eight types of questions. For the KDD CUP 2024 Challenge, the benchmark data were splited into three sets with similar distributions: validation, public test, and private test at 30%, 30%, and 40%, respectively. In total, 2,706 examples from validation and public test sets were shared.

The dataset also reflects varied entity popularity from popular to long-tail entities, and temporal spans ranging from seconds to years. Given the temporal nature of many questions, each question-answer pair is accompanied by an additional field denoted as "query time." This temporal marker ensures the reliability and uniqueness of the answers within their specific temporal context.

Content for retrieval. The CRAG dataset incorporates two types of content for retrieval to simulate a practical scenario for RAG: web search and knowledge graph (KG) search. This encompasses up to 50 full HTML pages for each question, retrieved from

a real-world search engine, as well as Mock KGs containing 2.6 million entities. Additionally, CRAG provides Mock APIs to simulate retrieval from a wide range of available information sources.

- **Web search results** Each retrieved web search result comprises five fields: *page name, page url, page snippet, page last modified* and *page result*. See Table 3 in Appendix A.1 for an example.
- **Mock KGs** A Knowledge Graph containing 2.6 million entities, which is accessed through Mock API.
- **Mock APIs** APIs for retrieving structured data from Mock KGs with predefined parameters, which are categorized by domain and output in JSON format. See Example in Appendix A.2 .

1.2 Task Desription

This challenge comprises three tasks designed to improve question-answering (QA) systems.

TASK 1: The organizers provide 5 web pages per question, potentially containing relevant information. The objective is to measure the systems' capability to identify and condense this information into accurate answers.

TASK 2: This task introduces mock APIs to access information from underlying mock Knowledge Graphs (KGs), with structured data possibly related to the questions. Participants use mock APIs, inputting parameters derived from the questions, to retrieve relevant data for answer formulation. The evaluation focuses on the systems' ability to query structured data and integrate information from various sources into comprehensive answers.

TASK 3: The third task increases complexity by providing 50 web pages and mock API access for each question, encountering both relevant information and noise. It assesses the systems' skill in selecting the most important data from a larger set, reflecting the challenges of real-world information retrieval and integration.

Each task builds upon the previous, steering participants toward developing sophisticated end-to-end RAG systems. This challenge showcases the potential of RAG technology in navigating and making sense of extensive information repositories, setting the stage for future AI research and development breakthroughs.

2 Methodlogy

Our approach, akin to most Retrieval-Augmented Generation (RAG) systems, comprises three primary phases: retrieval, augmentation and generation. In all phases, we implement routing mechanisms to address diverse query types. Figure 1 illustrates the pipeline of our solution, while the remainder of this section details the three main phases and the routers employed in this challenge.

2.1 Router

Routing is a crucial component of RAG systems, especially in real-world QA scenarios. In practical applications, RAG systems frequently incorporate multiple data sources. In the CRAG Challenge, we have three distinct data sources: Web Pages, Mock KGs, and Mock APIs. The diversity of questions requires routing queries to different data sources, individually or in combination. Even within a single data source, such as Mock APIs, the question-specific selection of appropriate APIs is crucial. Furthermore, we can tailor

prompt templates based on the nature of the question or route questions to different post-processing components.

In response to the specific characteristics of the questions in the CRAG Challenge, we designed two specialized routers: the Domain Router and the Dynamism Router. These routers are designed to efficiently navigate the complex landscape of multisource information retrieval and question-specific processing in our RAG system.

Domain Router. Domain router is fundamentally a classifier, more specifically, a sentence classifier. Given a query, the domain router assigns a specific domain from a predefined set: finance, sports, music, movie, and open. Based on the assigned domain, the workflow is then routed to the corresponding path.

We utilize *Llama3-8B-Instruct* [2] as our base model and enhance it with a classification head (Multilayer Perceptron, MLP) for domain classification. The 8B model inherently demonstrates a robust capability to comprehend the domain of queries. We randomly split the CRAG dataset into training, validation, and test sets with a ratio of 8:1:1. Based on this split, we performed a simple LORA (Low-Rank Adaptation) [7] fine-tuning to adapt to the distribution of the CRAG dataset. This approach facilitates the development of a high-quality classifier with minimal additional training.

The trained Domain router is employed at multiple stages within the system. During the retrieval phase, the Domain router is primarily used to select appropriate APIs. Following the retrieval of Web Pages and APIs, it is further applied for selective fusion of the retrieved knowledge. In the generation phase, we first customize the prompt templates based on the domain. Subsequently, after the model completes its generation, the Domain is also utilized for corresponding post-processing.

Dynamic Router. Analogous to the Domain router, the Dynamism router is also a sentence classifier. Given a question, the Dynamism router assigns a specific Dynamism, specifically one of: static, slow-changing, fast-changing, or real-time. The specific training methodology for the Dynamism router is congruent with that of the Domain Router, and thus will not be recapitulated here.

Due to the inherent limitation of Large Language Models in updating their internal knowledge, they are prone to providing outdated answers to dynamic questions. Even when employing external knowledge through RAG, LLMs can readily generate hallucinations as most data sources are static, unless real-time APIs are utilized. In the absence of real-time APIs, the more rapidly a question changes over time, the more susceptible LLMs are to hallucinations.

To attenuate hallucinations arising from dynamic questions, we implemented the Dynamism router for post-processing. In scenarios where real-time APIs are inaccessible, we excluded certain real-time questions and those that change rapidly over time.

2.2 Retrieval

As mentioned above, the CRAG dataset encompasses three types of content for retrieval: Web Pages, Mock KGs, and Mock APIs. For our final solution, we utilize two of these content types: Web Pages and Mock APIs.

2.2.1 ***Web Pages***. Web Pages are available for all 3 tasks of the challenge. For Task 1&2, 5 Web Pages per question are provided, each potentially containing relevant information. Task 3 increases
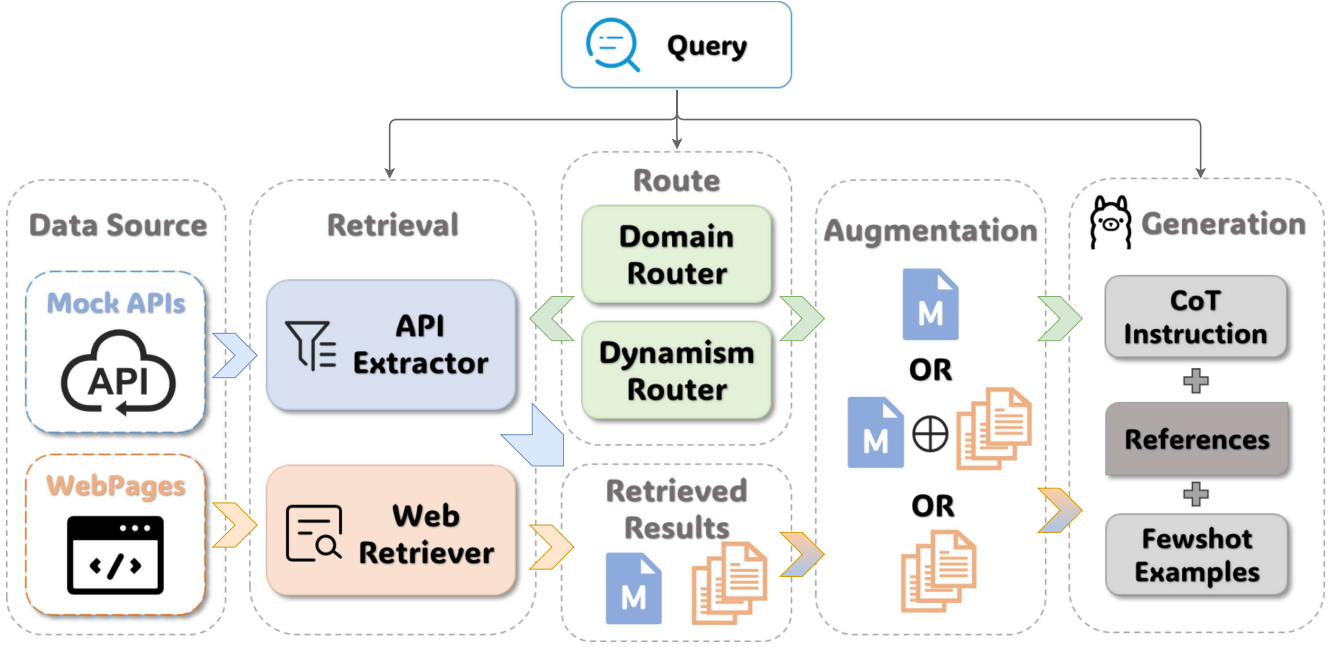
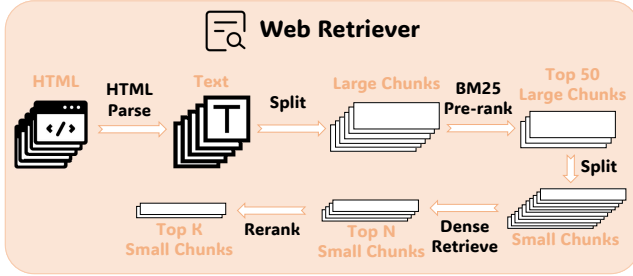**Figure 1: The overall pipeline of our solution.**

Figure 2: The pipeline of Web Retriever.

complexity by offering 50 Web Pages for each question, presenting both pertinent information and noise.

To enhance our QA system, we need to extract useful and relevant information from web search results. The primary process for retrieving web content is illustrated in Figure 2.

(1) **HTML Parsing**: Structured HTML is often unnecessarily verbose and contains substantial extraneous information that can impede subsequent segmentation operations. Therefore, it is crucial to first convert this structured format into natural language text that is more amenable to processing by Large Language Models. We conducted experiments with various HTML parsing methods, including BeautifulSoup, Newspaper, Markdownify, and several others. After evaluating both parsing efficiency and quality, we ultimately selected Newspaper. See A.3 for more details about experiment results.

(2) **Pre-ranking** (Task 3 only): For Task 3, fine-grained processing 50 Web Pages would be excessively time-consuming.

Therefore, we initially filter out an appropriate amount of relevant text before ranking. Specifically, we segment all the text from the Web Pages into chunks of 1024 tokens (calculated based on tokens rather than characters). For these segmented text chunks, we use BM25 [11] to select the top 50 most relevant text blocks.

(3) **Ranking**: In the ranking phase, we further refine the pre-ranked text blocks. For task 1&2, the text blocks are the 5 raw plain text extracted from HTML. The text blocks are segmented into smaller chunks, each comprising 256 tokens. We then transform these 256-token chunks into embeddings utilizing the *bge-m3* [4] model. Finally, we calculate the cosine similarity to select top 10 relevant chunks.

(4) **Re-ranking**: We utilized *bge-m3-v2-reranker* [6] to re-rank the aforementioned 10 relevant chunks, ultimately selecting the top 5 segments.

2.2.2 Mock APIs. A total of 38 Mock APIs were provided for tasks 2&3. As mentioned above, these Mock APIs can be categorized into five distinct domains, with no overlap between the APIs of different domains. Naturally, we designed separate workflows for each domain using a Domain Router. However, the overarching process flow of the workflows across all domains remains consistent, as shown in Figure 3:

(1) **Named Entity Recognition (NER)**: We directly utilize *Llama3-70B-Instruct* [2] to identify and classify named entities in the question into predefined categories, such as movie names and artist names. Specific prompts are presented in the Appendix A.4.
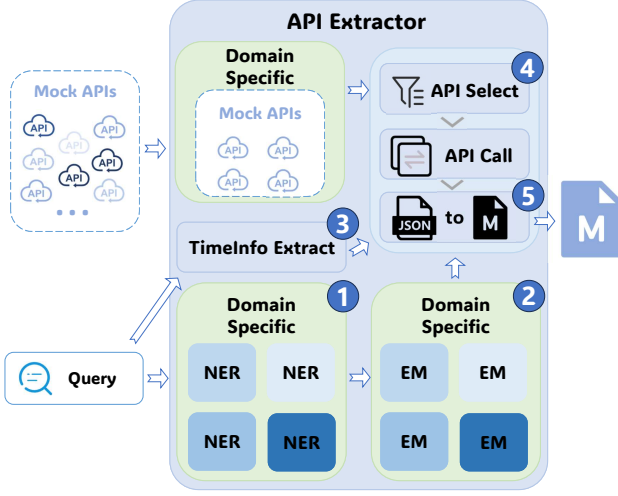
Figure 3: The pipeline of API Extractor.

(2) **Entity Match**: Matching extracted entities with API input parameters. Taking finance as an example, the input parameter for finance API is typically the ticker symbol, while user questions often contain full company names. We need to convert the company names to their corresponding ticker symbols. We first perform exact matching, requiring the extracted entity to be exactly the same as the input parameter. If no match is found, we then use BM25 to select the most similar one.

(3) **Time Information Extraction**: In addition to entities, numerous API inputs incorporate temporal information, requiring the extraction of relevant time points or intervals from user inputs. Notably, temporal information is often dependent with query time, as illustrated by terms such as "yesterday." In such cases, we must determine the specific time point based on the query time through relative time computation. We first use regular expressions to match certain time and date-related terms. Once matched, we use two Python packages (pytz and datetime) to calculate the Absolute Time. If no matches are found, the current time is used by default.

(4) **API Select**: Each domain comprises numerous APIs, not all of which are inherently relevant to a user's query. We have manually designed a set of rules to select APIs that correspond to the given question. To minimize the risk of overfitting the rules to the training set, we implemented constraints on the rule design process, prioritizing simplicity and robustness.

(5) **Json to Markdown**: The JSON output from APIs, while structured and machine-readable, may not be optimal for large language models (LLMs) to process efficiently. Converting this JSON data into a more LLM-friendly Markdown format can enhance the model's ability to understand and utilize the information.

2.3 Augmentation

We employ input-layer integration for generation augmentation, which combines retrieved information/documents with the original input/query and jointly passes them to the generator. In contrast to common input-layer integration, we do not utilize all retrieved documents. For different domains, we select specific data sources and integrate them to construct the final reference.

For the open domain, since we did not employ a Mock API, we exclusively utilized web search results. For the movie and music domains, where most queries are relatively static or evolve slowly, results retrieved from both web pages and mock APIs remain relevant. Therefore, we chose to integrate these two sources. For the sports and finance domains, which involve numerous real-time and fast-changing queries, we exclusively used Mock APIs to ensure the timeliness and relevance of the retrieved information.

2.4 Generation

In the generation phase, we employed two widely-used methodologies: Chain-of-Thought (CoT) reasoning and In-context Learning. After generation, we performed a simple post-processing procedure on the generated results based on the Domain and Dynamism Routers.

2.4.1 ***Chain of Thought***. Chain-of-Thought (CoT) [12] enhances the reasoning process of language models by prompting them to articulate intermediate steps in problem-solving. This approach not only enhances the model's ability to handle complex tasks but also significantly reduces hallucinations.

2.4.2 ***In-context Learning***. We improve the model's ability to recognize invalid questions, particularly those based on false premises, through In-context Learning. We develop adaptive few-shot examples [3], selecting two of the most representative invalid question samples for each domain and elucidating the reasons for their invalidity. Using the sports domain as an example, our few-shot samples are as follows:

Few-shot Example I

What's the latest score for OKC's game today?

- -

There is no game for OKC today.

Few-shot Example II

How many times has Curry won the NBA dunk contest?

- -

Steph Curry has never participated in the NBA dunk contest.

2.4.3 ***Post-processing***. Before finalizing the results, we implement basic post-processing strategies. Based on the question domain and volatility, we assign "I don't know" responses to queries susceptible to hallucination. For domains lacking real-time API access, specifically open, movie, and music categories, we designated "I don't know" answers to fast-changing and real-time questions. Furthermore, due to the model's limited mathematical computation

Table 1: Overall Preformance of our solutions on all 3 Tasks.

	Score(%)	Accuracy(%)	Hallucination(%)	Missing(%)
LLM Only	-7.29	28.01	35.30	36.69
Direct RAG	-6.78	34.79	41.58	**23.63**
Task 1	11.82	29.98	18.16	51.86
Task 2	31.22	46.75	**15.54**	37.71
Task 3	**31.66**	**48.21**	16.56	35.23

Table 2: Ablation Study for Major Strategies Employed in the System.

		Score(%)	Accuracy(%)	Hallucination(%)	Missing(%)	Time Cost(s)
Task 2	**w/o Rerank**	29.17	43.54	14.37	42.09	-
	w/o EntityMatch	21.44	32.31	10.87	56.82	-
	w/o TimeInfoExtract	18.45	28.45	**9.99**	61.56	-
	w/o Fewshot&CoT	25.53	52.08	26.55	**21.37**	-
	w/o Fewshot	27.13	51.35	24.22	24.43	-
	w/o CoT	28.52	**53.32**	24.80	21.88	-
	Ours	**31.22**	46.75	15.54	37.71	-
Task 3	**w/o Prerank**	29.53	44.34	**14.81**	40.85	68.17
	Ours	**31.66**	**48.21**	16.56	**35.23**	**5.96**

capabilities, we implement same processing for questions requiring numerical calculations, such as those involving the term "average." Although these approach may be deemed simplistic, it demonstrated efficacy in significantly reducing hallucinations.

3 Experiments

In this section, we present our main results and ablation studies for some crucial components.

We did not employ a strategy of fine-tuning the LLM; instead, we used the LLM in a zero-shot setting. According to the rules set by the organizers, we used *Llama3-70B-Instruct* [2] for all our LLMs. For the embedding model, we used *BAAI/bge-m3*, and for the rerank model, we used *BAAI/bge-m3-v2-reranker*. In the 1371 public test cases officially released for this round, we compared the following baselines: LLM only (using the LLM without retrieving references) and straightforward RAG (the baseline provided by the organizers, using straightforward RAG solutions).

3.1 Metrics and Evaluation

In line with CRAG Benchmark, we conduct a model-based automatic evaluation for our experiment. Automatic evaluation employs rule-based matching and GPT-4 [1] assessment to check answer correctness. It will assign three scores: correct (1 point), missing (0 points), and incorrect (-1 point). The final score for a given RAG system is computed as the average score across all examples in the evaluation set.

3.2 Overall Performance

Comparing our solutions to the RAG Baseline, we observe significant advantages in performance across all three tasks. In Table 1, our approach showcases notable improvements in accuracy and

information retention. Specifically, when contrasted with the RAG Baseline, our solutions demonstrate superior results with reduced hallucination rates and enhanced information completeness. Task 2 and Task 3, in particular, exhibit substantial enhancements in accuracy and reduced hallucination percentages, highlighting the effectiveness of our proposed methodologies in addressing these key metrics.

3.3 Ablation Study

Table 2 presents the ablation study for major strategies employed in our solution.

During the retrieval phase, we implemented several strategies, including pre-ranking, re-ranking, Entity Match, and Time Information Extraction. Ablation studies revealed that pre-ranking and re-ranking marginally reduce performance, while Entity Match and Time Information Extraction significantly decrease performance. Both pre-ranking and re-ranking significantly contribute to the improvement of retrieval quality. Pre-ranking enhances retrieval performance by proactively filtering out a significant amount of noise, while re-ranking ensures the accuracy of retrieval results through more refined and granular sorting. The enhancement in retrieval quality ultimately translates into an increase in answer accuracy. The absence of pre-ranking and re-ranking demonstrably leads to a substantial decrease in the accuracy of the final answers. Furthermore, pre-ranking significantly enhances retrieval effiency and reduces the retrieval time. Entity Matching and Time Information Extraction form the basis of using MOCK APIs. They ensure the accuracy of API call parameters, which is crucially linked to the overall performance. The absence of either component can result in a significant performance decline.

During the generation phase, we employed 2 main components: domain specific fewshot examples and Chain of Thought prompt. Both aforementioned components led to a substantial reduction in hallucinations. The combined implementation of these components yielded a reduction in hallucinations of up to 71%, consequently resulting in a 22% increase in the final score. The experimental results demonstrate the effectiveness and necessity of the two components.

4 Perspectives

Our method presents a robust and versatile framework for addressing a wide range of dynamic and complex real-world problems. This approach, however, also opens up several avenues for further investigation.

- **Model Cognitive Ability Assessment** Most conventional QA evaluation methods primarily focus on accuracy, neglecting the impact of hallucinations. Models should be aware of their knowledge boundaries, discerning what they should and should not answer. CRAG incorporates hallucinations into evaluation metrics, but its settings lack sufficient justification. Responding "I don't know" to all questions can yield a satisfactory score. Exploring the assessment of models' cognitive abilities using methodologies for evaluating human cognition is a promising research direction.
- **API Integration and Scalability**. In real-world scenarios, where extensive API usage is common, our manually designed matching rules are likely to prove inadequate. The development of a more universal method for selecting and calling APIs, as well as processing the returned results, represents a promising avenue for future research.
- **Handling Dynamic Information**. For questions that involve information that changes dynamically over time, simply refusing to answer is merely a basic solution. Future research should focus on exploring methods to acquire the most up-to-date knowledge and determine the timeliness of information. This is crucial to avoid hallucinations caused by outdated knowledge and to ensure the system provides accurate, current information. Developing techniques for real-time information retrieval and verification, as well as implementing mechanisms to assess the reliability and currency of data sources, are key areas for investigation.

5 Conclusion

In this paper, we introduce our solution for the Meta KDD CUP 2024: CRAG Comprehensive RAG Benchmark. We adopt a classic RAG framework with two specific routers. In the retrieval phase, we demonstrated the process of obtaining high-quality information from various data sources and utilizing the Domain Router for information filtering. In the augmentation phase, we employed the Domain Router in a similar manner for information aggregation based on domain characteristics. Finally, in the generation phase, we implemented two methods to significantly improve the model's accuracy and reduce hallucinations, while further mitigating hallucinations through post-processing based on the question's domain and dynamic nature.

Our approach offers a viable pathway for addressing the diverse and dynamic challenges encountered in real-world scenarios. Nevertheless, our method has certain limitations. We have identified several inherent issues in the current methodology and provided our insights and reflections on the specific problems related to our approach. Ultimately, we anticipate that this study will make a modest contribution to the broader RAG and LLM communities.

References

[1] 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 arXiv:2303.08774* (2023).
[2] Meta AI. 2024. Meta LLaMA 3. *Meta AI Blog* (2024). https://ai.meta.com/blog/meta-llama-3/
[3] 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 the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS '20)*. Curran Associates Inc., Red Hook, NY, USA, Article 159, 25 pages.
[4] 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).
[5] Mingyue Cheng, Hao Zhang, Jiqian Yang, Qi Liu, Li Li, Xin Huang, Liwei Song, Zhi Li, Zhenya Huang, and Enhong Chen. 2024. Towards Personalized Evaluation of Large Language Models with An Anonymous Crowd-Sourcing Platform. *Companion Proceedings of the ACM on Web Conference 2024* (2024). https://api.semanticscholar.org/CorpusID:268379217
[6] FlagOpen. 2024. FlagEmbedding Reranker. https://github.com/FlagOpen/FlagEmbedding/tree/master/FlagEmbedding/reranker.
[7] J. Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. *ArXiv* abs/2106.09685 (2021). https://api.semanticscholar.org/CorpusID:235458009
[8] Junzhe Jiang, Shang Qu, Mingyue Cheng, and Qi Liu. 2023. Reformulating Sequential Recommendation: Learning Dynamic User Interest with Content-enriched Language Modeling. *ArXiv* abs/2309.10435 (2023). https://api.semanticscholar.org/CorpusID:262054865
[9] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems* 33 (2020), 9459–9474.
[10] Yucong Luo, Mingyue Cheng, Hao Zhang, Junyu Lu, Qi Liu, and Enhong Chen. 2023. Unlocking the Potential of Large Language Models for Explainable Recommendations. *ArXiv* abs/2312.15661 (2023). https://api.semanticscholar.org/CorpusID:266551720
[11] Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval* 3, 4 (2009), 333–389.
[12] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems* 35 (2022), 24824–24837.
[13] Xiao Yang, Kai Sun, Hao Xin, Yushi Sun, Nikita Bhalla, Xiangsen Chen, Sajal Choudhary, Rongze Daniel Gui, Ziran Will Jiang, Ziyu Jiang, et al. 2024. CRAG–Comprehensive RAG Benchmark. *arXiv preprint arXiv:2406.04744* (2024).

A Appendix

A.1 An Example of Web Search Results.

Table 3: An Example of Web Search Results.

Key	Value
"page name"	"Microsoft Office 2019 - Wikipedia"
"page url"	"https://en.wikipedia.org/wiki/Microsoft_Office_2019"
"page snippet"	"For Office 2013 and 2016, various editions containing the client apps were available in both Click-To-Run..."
"page last modified"	"Tue, 27 Feb 2024 22:55:55 GMT"
"html page"	"<!DOCTYPE html> <html class=...> <head> ... <title>Microsoft Office 2019 - Wikipedia</title>..."

A.2 An Example of Mock APIs.

Most mock APIs typically take entities as input and output entity-related information in JSON format. The following is an example of an API in the economic domain:

```
get_detailed_price_history
```

Description: The function returns the past 5 days' history of 1-minute Stock price, starting from 09:30:00 EST to 15:59:00 EST.

Input:
- `ticker_name`: str, upper case

Output:
- Past 5 days' 1-minute price history: `json`

A.3 Experiment Results of HTML Parsing Methods

Table 4: Experiment Results of HTML Parsing Methods

Method	Time Cost(s)	Success Rate(%)	Score(%)
beautifulsoup	0.31	**100.0**	5.11
boilerpy3	**0.25**	97.2	9.42
trafilatura	0.64	96.4	10.14
newspaper	1.96	98.8	**11.82**
markdownify	3.65	**100.0**	10.94

A.4 Prompts Used for Name Entity Recognition

Music NER Prompt

Please identify and list all the named entities present in the following question about music instead answering it, categorizing them appropriately (e.g., persons, song, band) Your answer should be short and concise in 50 words.

Format your response as follows: For each entity, provide the name followed by its category in parentheses. Categories include persons, songs and bands. Ensure that your response is clearly structured and easy to read.

Question: "{query}"

Output only the named entities present in the question. Do not include any other information. If there are no named entities in the question, please provide an empty response.

Expected Output Format:
a name of a person in the sentence (person)
a name of a song in the sentence (song)
a name of a band in the sentence (band);

Every entity should be in a new line and be in the format of "entity_name (entity_category)"

Sports NER Prompt

Please identify and list all the named entities present in the following question about sports instead answering it, categorizing them appropriately (e.g., nba team, soccer team, nba player, soccer player) Your answer should be short and concise in 50 words.

Format your response as follows: For each entity, provide the name followed by its category in parentheses. Categories include nba teams, soccer teams, nba players, soccer players. Ensure that your response is clearly structured and easy to read.

Question: "{query}"

Output only the named entities present in the question. Do not include any other information. If there are no named entities in the question, please provide an empty response.

Expected Output Format:
a name of a nba team in the sentence (nba team)
a name of a soccer team in the sentence (soccer team)
a name of a nba player in the sentence (nba player)
a name of a soccer player in the sentence (soccer player);

Every entity should be in a new line and be in the format of "entity_name (entity_category)"

Movie NER Prompt

Please identify and list all the named entities present in the following question about movie instead answering it, categorizing them appropriately (e.g., person, movie) Your answer should be short and concise in 50 words.

Format your response as follows: For each entity, provide the name followed by its category in parentheses. Categories include persons, and movies. Ensure that your response is clearly structured and easy to read.

Question: "{query}"

Output only the named entities present in the question. Do not include any other information. If there are no named entities in the question, please provide an empty response.

Expected Output Format:
a name of a person in the sentence (person)
a name of a movie in the sentence (movie)

Every entity should be in a new line and be in the format of "entity_name (entity_category)"

Finance NER Prompt

Please identify and list all the named entities present in the following question about finance instead answering it, categorizing them appropriately (e.g., company, ticker symbol) Your answer should be short and concise in 50 words.

Format your response as follows: For each entity, provide the name followed by its category in parentheses. Categories include company, and symbol(which means the ticker symbol of a company). Ensure that your response is clearly structured and easy to read.

Question: "{query}"

Output only the named entities present in the question. Do not include any other information. If there are no named entities in the question, please provide an empty response.

Expected Output Format:
a name of a company in the sentence (company)
a name of a ticker symbol in the sentence (symbol);

Every entity should be in a new line and be in the format of "entity_name (entity_category)"