KDD Cup CRAG competition: Systems, Findings and Learnings

Xiao Yang[††1], Yifan Ethan Xu[*,1], Kai Sun[*,1], Jiaqi Wang[*,1], Lingkun Kong[1],
Wen-tau Yih[2], Xin Luna Dong[1]
[1]Meta Reality Labs, [2]FAIR, Meta
{xiaoyangfb, ethanxu, sunkaicn, jqwang, klk,
scottyih, lunadong}@meta.com

Abstract

The KDD Cup 2024 CRAG Challenge aims to provide a foundation for evaluating Retrieval Augmented Generation (RAG) systems. RAG systems take natural language questions as input, decide whether and how to use the facilitating information such as web pages and mock Knowledge graphs, and return natural language answers. A good RAG system provides correct and useful answers to questions without bringing in hallucinated information. In this report, we describe the motivation for organizing this challenge, review the design for the benchmark and the competition, discuss observations from the submissions, and reflect the learnings from hosting the challenge.

A Introduction to RAG and the CRAG benchmark

The rise of Large Language Models (LLMs) has revolutionized the field of natural language processing [22, 24, 53, 58], but these models still struggle with providing accurate and reliable answers to complex questions. One major challenge is the tendency for LLMs to "hallucinate" or generate responses that are not grounded in factual information [18, 34, 40, 42]. To address this issue, researchers have turned to Retrieval-Augmented Generation (RAG) [6, 10, 13, 20], a technique that involves searching external sources to gather relevant information and then using that information to generate more accurate answers (Figure 50).

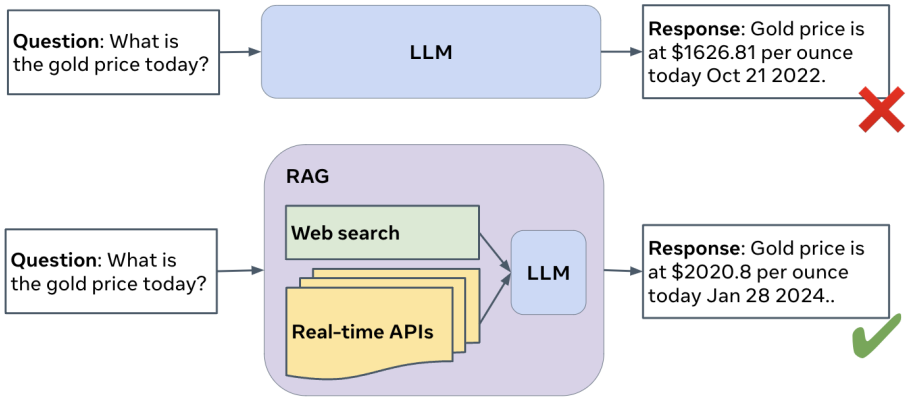

Figure 50: Given a question, a RAG system searches external sources to retrieve relevant information and then provides grounded answers.

[††]The first four authors contributed equally.

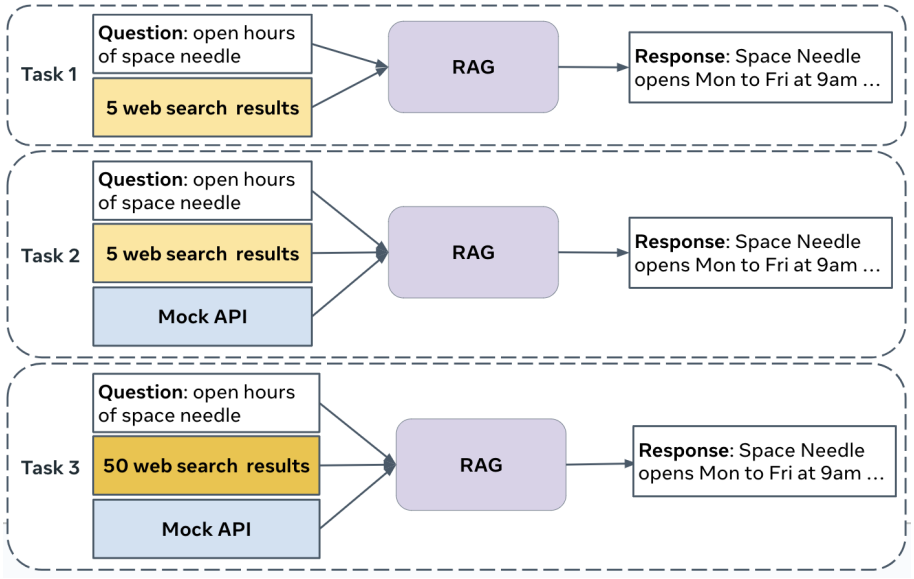

Figure 51: KDD Cup 2024 Meta CRAG challenge tasks.

However, existing datasets for evaluating RAG systems are limited in their scope and diversity, failing to capture the complexity and nuance of real-world question-answering tasks. To address this limitation, we developed the Comprehensive RAG (CRAG) Benchmark, a comprehensive evaluation framework designed to assess the capabilities of RAG systems. The benchmark comprises over 4.4K question-answer pairs, accompanied by mock APIs that mimic the process of searching the web and accessing knowledge graphs. CRAG is across five domains (finance, sports, music, movie, and encyclopedia open domain), and a diverse range of question types, including simple-fact questions, questions that require aggregation and reasoning. CRAG further encompasses varied entity popularity from popular to long-tail and temporal spans ranging from seconds to years. As such, it covers a broad spectrum of real user queries, providing a realistic and challenging testbed for RAG systems [52]. By incorporating both web and knowledge graph search, CRAG simulates realistic information retrieval scenarios, reflecting the common practice of users seeking answers from multiple sources. Our evaluations using this benchmark revealed substantial performance gaps in even the most advanced LLMs and RAG systems, highlighting the need for continued research and development in this area.

The CRAG benchmark served as the foundation for the KDD Cup 2024 competition, which drew over 2.5K participants and 5.6K submissions from around the world. The winning solutions demonstrated significant improvements over baseline methods. In this paper, we present key insights and takeaways from the competition, including our learnings on the RAG landscape (Section C), learnings from the CRAG winning solutions (Section D), and learnings from hosting the CRAG challenge (Section E), suggesting directions for future RAG research.

B Challenge Overview

B.1 Challenge Tasks

A RAG QA system takes a question Q as input and outputs an answer A; the answer is generated by LLMs according to information retrieved from external sources or directly from the knowledge internalized in the model. The answer should accurately address the question while avoiding any hallucinations.

We designed three tasks, which share the same set of (question, answer) pairs but differ in the external data accessible for retrieval to augment QA, as shown in Figure 51. Here, we provide the content that can be leveraged in QA to ensure fair comparisons.

Task 1: Retrieval Summarization. In Task 1, we provide up to five web pages for each question. These web pages are likely, but not guaranteed, to be relevant to the question. This task aims to test the answer generation capability of a RAG system.

Task 2: KG and Web Retrieval Augmentation. In Task 2, we in addition provide mock APIs to access information from underlying mock KGs. The mock KGs store structured data relevant to the questions; answers to the questions may or may not exist in the mock KGs. The mock APIs take input parameters, oftentimes parsed from the question, and provide structured data from the mocked KGs to support answer generation. This task tests how well a RAG system 1) queries structured data sources and 2) synthesizes information from different sources.

Task 3: End-to-end RAG. Similar to Task 2, Task 3 also provides both web search results and mock APIs as candidates for retrieval but provides 50 web pages, instead of 5, as candidates. The larger set of web pages increases the likelihood of providing relevant information to answer the question, but they also tend to contain more noise. As such, Task 3 in addition tests how a RAG system ranks a larger number of retrieval results.

The three tasks, each adding upon the previous one, allow testing different capabilities of the end-to-end RAG systems.

B.2 Data and System Requirement

The challenge used the CRAG benchmark as described in Section A. We split the data randomly into validation, public test, and private test at 30%, 30%, and 40%, and released the validation and public test sets for the challenge. The final winners were determined by evaluating against the held-out private test set.

In order to make the systems more comparable and to encourage open source development, we require all submitted systems to be based on Llama 2 [44] or Llama 3 [3] models in this challenge. Participants can also fine-tune their models using publicly available data as long as the data were not generated by proprietary models.

The challenge was hosted on the AICrowd competition platform, and results were posted on a leaderboard. All submissions were run on AWS G4dn.12xlarge instances equipped with 4 NVIDIA T4 GPUs with 16GB GPU memory. Since neither the Llama 2 70B or Llama 3 70B models in full precision can be directly run on these T4 GPUs, participants needed to use quantization or other techniques to make their system runnable on the inference platform. Also, network connection was disabled during the challenge to ensure all participants can access an equal set of retrieved contents, and to prevent leak of the private test set data. Moreover, each example had a time-out limit of 30 seconds and was truncated to 75 BPE tokens [38] in the auto-evaluation. In human-evaluation, graders examined the first 75 bpe tokens to find valid answers, but reviewed the whole response to judge for hallucination. See section B.3 for more details about the evaluation.

B.3 Evaluation

Metrics

We use a scoring method to assess the performance of RAG systems. For each question in the evaluation set, we first label the answer with **perfect, acceptable, missing,** or **hallucination**, according to the following criteria.

Perfect. The response correctly answers the user's question and contains no hallucinated content.

Acceptable. The response provides a useful answer to the user's question but may contain minor errors that do not harm the usefulness of the answer.

Missing. The response is "I don't know", "I'm sorry I can't find ...", a system error such as an empty response, or a request from the system to clarify the original question.

Hallucination. The response provides wrong or irrelevant information to answer the user's question.

We then use a scoring method $\textbf{Score}_h$ with score 1, 0.5, 0, and -1 for each *perfect, acceptable, missing*, and *hallucinated* answer, respectively, where we penalize hallucinated answers because users would prefer the model to admit "I don't know" than providing an hallucinated answer when it does not know how to answer the question. We then define **Truthfulness** as the average score from all examples in the evaluation set for a given RAG system. Concretely,

$$\text{Truthfulness}_h = \text{Perfect rate} + 0.5 * \text{Acceptable rate} - \text{Hallucination rate}.$$

Evaluation Method

We employ a hybrid evaluation system that includes both human evaluation **(human-eval)** and model-based automatic evaluation **(auto-eval)**. In the former, we use manual grading to judge perfect, acceptable, missing, and hallucinated for each answer. In the latter, we merge perfect and acceptable, call it **accurate**, and use a three-way scoring $\textbf{Score}_a$ with $1, -1, 0$ for accurate, hallucinated, and missing answers. And in this case,

$$\text{Truthfulness}_a = \text{Accuracy} - \text{Hallucination rate}.$$

We design a two-step method for automatic evaluation: if the answer matches the ground truth exactly, it is considered accurate; otherwise, we use LLMs to determine whether the response is accurate, hallucinated, or missing. To avoid the self-preference problem [31], we experimented with two families of LLM evaluators: ChatGPT (`gpt-3.5-turbo-0125`) [29] and Llama 3 (`llama-3-70B-instruct`) [3] and reported the average accurate, hallucinated, missing rates, and scores from the two models for benchmarking the straightforward solutions [52]. This two-step method yields an accuracy of 94.5% for ChatGPT and 99% for Llama 3 compared to human-eval. We adopted ChatGPT as the auto-evaluator in the challenge due to its low cost.

B.4 Challenge Schedule and Outcomes

The challenge was announced on March 15, 2024. Submissions were accepted starting on April 1, 2024, simultaneously with the release of the starter kit. Baseline models were released a week after. The challenge had two phases: in Phase 1, each team can make up to six submissions per week for all three tasks together during a two-month period, and check their results directly on the leaderboard based on auto-eval; in Phase 2, each team can submit up to six times for the three tasks together in total. The submitted solutions in Phase 2 were first evaluated by auto-eval. Then the top-15 teams from each task were sent for human-eval to determine the final winners. Phase 2 ended on June 22, 2024, and the final results were announced on July 28, 2024.

The challenge awarded the top-3 teams that obtained the highest Truthfulness scores in each task. It also provided seven additional prizes for teams that achieved the highest scores for each of the seven complex question types. In the end, twelve teams won the prizes (See Table 32 for the list of the winning teams.), among which six winning teams were invited to present in the KDD 2024 CRAG workshop[1]. All participating teams were encouraged to submit a technical report for their solutions.

[1]Please see more information about the workshop at: https://kddcup24.github.io/pages/benchmark.html

The challenge attracted more than 2,500 participants, 384 teams and more than 5,600 submissions. After the challenge was complete, we held a workshop during the ACM KDD 2024 conference. The workshop featured three keynote speeches, six winning-team presentations, and attracted more than 130 onsite audiences. It also published 11 technical reports from the participating teams afterwards.

C Learnings on RAG landscape

	Solution	Accuracy	Hallucination	Missing	Truthfulness	Latency (ms)
LLM-Only	Llama 3	32.3%	28.9%	38.8%	3.4%	
	GPT-4	33.5%	13.5%	53.0%	20.0%	
Straightforward RAG	Llama 3	40.6%	31.6%	27.8%	9.1%	
	GPT-4	43.6%	30.1%	26.3%	13.5%	
CRAG KDD Cup Winners ⋆	Top-3: vsluy-team	43%	24.9%	30.1%	18.1%	
	Top-2: APEX	48.6%	13.7%	36.3%	34.9%	
	Top-1: db3	53.3%	17.1%	28.0%	36.2%	
Industry SOTA ⋆⋆	Perplexity.ai	60.2%	25.3%	10.1%	34.9%	4,634
	Meta SG	57.4%	16.0%	21.8%	41.4%	**3,431**
	ChatGPT Plus	66.5%	25.0%	1.9%	41.5%	6,195
	Gemini Advanced	65.9%	16.6%	12.5%	49.3%	5,246
	Copilot Pro	68.5%	17.9%	7.8%	**50.6%**	11,596

Table 31: Truthfulness scores on CRAG. (⋆: the score is obtained by human evaluation and the others are obtained by automatic evaluation; ⋆⋆: in addition, different accesses to retrieval contents. [52].)

Table 31 summarizes the performance in truthfulness score of baselines (the LLM only and straightforward RAG solutions), the KDD Cup winner solution, and the best-performing industry state-of-the-art (SOTA) system on CRAG. We have the following observations and details can be found in [52].

1. The LLM-only solution performs poorly on CRAG, with truthfulness ranging from 3% to 20%.
2. Although RAG can help answer more questions, adding RAG in a straightforward manner does not necessarily reliably outperform the LLM-only solution (truthfulness ranging from 9% to 13%). This is because careless summarization in baseline RAG solutions can introduce more hallucinations generated from irrelevant retrieval results.
3. The CRAG KDD Cups winning solutions improve the truthfulness significantly to 36%, but still falls short of perfection in terms of answer accuracy and latency. We will describe the solutions in more detail in Section D.
4. Industry SOTA systems, which likely leverages better models and retrieval systems (e.g., web and KG search engines), achieved 51% truthfulness. The systems that achieved the highest truthfulness and that had the lowest latency were different, reflecting the quality-latency tradeoff. There is still much room for improvement on quality and latency.

D Learnings from the winning solutions

More than 2500 participants took part in the CRAG competition. Four teams won top-three positions across three tasks, and seven additional teams scored high points for individual question types (see Table 32).

Table 32: CRAG competition winning teams and team placements. For each task we awarded top-3 teams for overall scores, and top-1 team for each question type.

Team name	**Task 1 placement**	**Task 2 placement**	**Task 3 placement**
db3 [49]	**1st place**	**1st place**	**1st place**
APEX [30]		**2nd place**	**2nd place**
md_dh [11]	**2nd place**	**3rd place**	 Set Aggregation Post-processing
vslyu-team [51]			**3rd place**
ElectricSheep [55]	**3rd place**	Simple with condition Set Aggregation Multi-hop Post-processing	
dRAGonRAnGers [32]	Comparison Post-processing	Comparison	Comparison
ETSLab	False premise		Multi-hop
dummy model [16]	Simple with condition Set Aggregation		
bumblebee7 [56]	Multi-hop		
Future [9]		False premise	
StarTeam [48]			Simple with condition
Riviera4 [43]			False premise

Most winning solutions adapt a general RAG system design that comprises two main stages (see Figure 52): (1) Knowledge Retrieval components that retrieve and process relevant information from web or knowledge graphs, and (2) an Augmented Generation component that leverages an LLM to incorporate retrieved information and provide an answer to the question.

No team trained the overall pipeline in an end-to-end fashion. Instead, each component is optimized separately, where sub-systems are tailored to the specific tasks or question types. The teams leverage both off-the-shelf solutions (e.g., Beautifulsoup for HTML parsing, BGE encoders for ranking) and customized solutions (e.g., fine-tuned LLM to reduce hallucination). In the following we deep dive to learnings from each component.

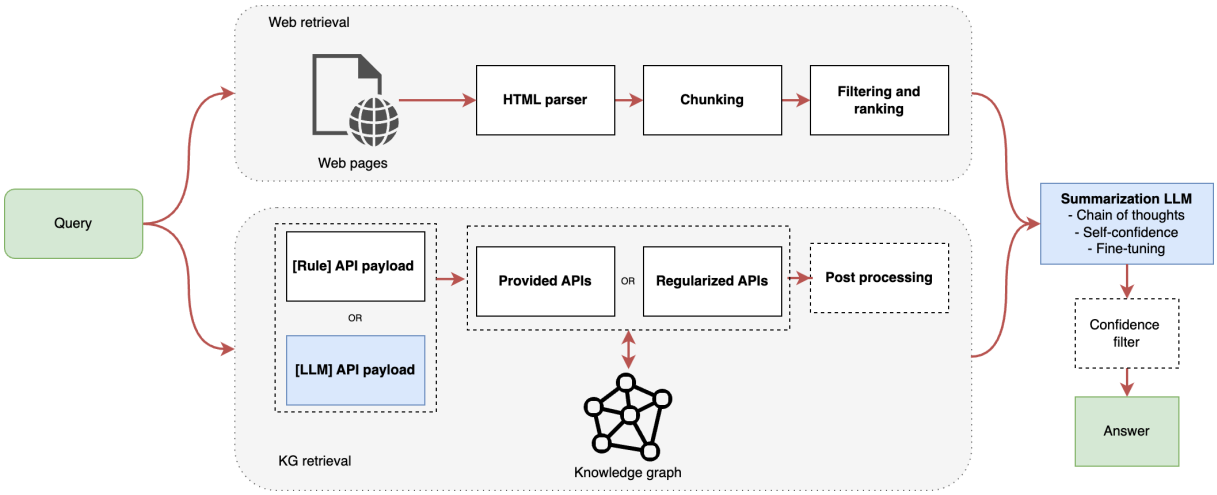

Figure 52: Conceptual overall architecture from winning solutions.

D.1 Knowledge Retrieval from Web

Given a query, this component retrieves supporting material from the web repository (task 1, 2 and 3). An ideal retrieval system finds relevant knowledge efficiently (high recall) without including too many noises (high precision). There are several key challenges: 1) raw HTML contains inconsistent structures, making it challenging to extract clean text; 2) long web pages exceed LLM's input length constraints; 3) webpages often contain contents that are irrelevant to the question that dilute signals and potentially lead to hallucinations from summarizations.

To address these challenges, most winning solutions follow a workflow consisting of four steps to obtain clean relevant information from webpages: HTML parsing, content chunking, relevance scoring, and re-ranking. The teams heavily utilize off-the-shelf libraries as building blocks as listed in Table 33 below.

Table 33: Off-the-shelf libraries the winning solutions leveraged for web retrieval.

Component	Libraries
HTML Parsing	BeautifulSoup [35], Trafilatura [5]
Chunking	CharacterTextSplitter [26] , ParentDocumentRetriever [26]
Relevance scoring	bge-base [50], ms-marco-miniLM [27] , sentence-t5 [28], BM25 [36]
Re-ranking	bge-reranker [8]

The teams leveraged a variety of chunking strategies to partition web content into smaller segments. Beyond the basic token-level chunking method, three solutions stand out in ensuring related contexts are preserved in the same chunk, allowing the summarization model to access coherent information:

1. Db3's [49] solution maintains a parent-child chunk relationship using the ParentDocumentRetriever library. Once they identify smaller children chunks that are relevant to the question, they include their associated parent chunks in the pipeline.

2. Md_dh's [11] solution traverses the tree structure of HTML to identify chunks at node level, minimizing segmentation of texts under a common tag such as header, table, sections.

3. ElectricSheep's [55] solution first identifies questions in web pages by matching sentences that start with "5W1H" key words (i.e. "Who", "What", "Where", "When", "Why" and "How") and end with a question mark, and then ensures subsequent texts are grouped together with the questions.

For relevance scoring and re-ranking, the winning solutions mainly rely on off-the-shelf sparse (e.g. BM25) and dense (e.g. bge-base) retrieval models and re-ranker models (e.g. bge-reranker) without customized fine-tuning. Ablation study (md_dh [11]) shows that dense retrieval via cross-encoders has a slight edge over sparse retrieval in terms of accuracy.

D.2 Knowledge Retrieval from KG

CRAG provides various domain specific APIs to access underlying KG data (task 2,3). The main tasks for efficient retrieval from KG are three folds: 1) identifying the appropriate API(s), 2) generating correct API parameters, and 3) processing API results to find relevant information.

The winning solutions for KG retrieval fall into three broad categories.

1. Build each retrieval step separately. Using APEX's [30] solution as an example, this approach invokes the following steps. It first leverages an LLM to identify named entities, followed by entity linking via string matches or BM25 fuzzy matches. Then it identifies an appropriate API and its parameters via domain-specific rules and calls the API. Afterwards, it filters the API results based on the date of the query. Finally it converts the filtered results to markdown format and sent to the final generation model.

   This approach provides explicit control and insights of each step at the cost of integration complexity.

2. Generate e2e API call directly. A few teams (db3 [49], md_dh [11], ElectricSheep [55] etc.) prompt an LLM to directly generate API payload, where fine-tuned LLMs with API specific data further improve the calling accuracy.

3. Regularized API. Another innovative solution came from team db3 [49]. Instead of building a retrieval system around the original APIs provided by CRAG, the authors develop a new set of "regularized" APIs that enhances the expressiveness of the original APIs. More specifically, the solution built API wrappers to encapsulate filtering conditions (e.g."year" == "2012") and aggregation logics (e.g. maximum, average). The following is an example comparing the original API and the regularized API.

$$get\_movie(movie\_name) \rightarrow get\_movie(movie\_name, condition)[key\_name] \quad (13)$$

   where "condition" is from a set of predefined rules, and "key_name" are indices to search in the results. The expressive regularized APIs allow the team to obtain concise information via a single API call generated by LLM.

D.3 LLM Augmented Generation

LLM Augmented Generation incorporates upstream retrieved knowledge to generate a final answer. All winning solutions put significant efforts in reducing hallucination in this step, partially because hallucination is penalized more severely than a missing (e.g. "I don't know") answer in the CRAG scoring criteria.

The main strategies that the winning solutions adapt to reduce hallucinations are a combination of chain-of-thought prompting, explicit confidence estimation, supervised fine-tuning, and manual rules.

1. Chain-of-thought. Teams (e.g., ElectricSheep [55], APEX [30]) show that prompting the LLM to articulate intermediate thinking steps improve answering accuracy for complex questions and significantly reduce hallucination. For instance, ElectricSheep [55] prompts the model to first identify if a question has false premise, then whether the question can be answered by each retrieved evidence, before generating the final answer.

2. Explicit confidence inference. Another approach is to estimate LLM answer's confidence, then only retain the answer at high confidence, and answer "I don't know" otherwise. Winning solutions either prompt the LLM directly to self produce a confidence level, or sample several answering paths and approximate the confidence as the answer consistency. Both methods are effective in their respective pipeline, but the self-consistency approach demands more computational resources.

3. Supervised fine-tuning. Several teams (e.g., db3 [49], md_dh [11], dRAGonRAnGers [32]) fine-tune an Llama-3-8b model specifically targeting reducing hallucinations. The teams generated training labels by first running the pipeline with a non-fine-tuned LLM generation model, then prompts a separate LLM to automatically identify questions that are not answerable by retrieved knowledge passages and labeled "I don't know" for fine-tuning. The resulting fine-tuned model answers "I don't know" more frequently on questions that it tends to hallucinate originally.

4. Manual rules. Since dynamic questions are much more difficult to answer correctly than static questions. Some teams opt to avoid answering dynamic questions all together to prevent getting penalized with hallucinated answers.

D.4 Observations by Question Categories

CRAG categorizes questions along four dimensions: Topic domain (Finance, Sports, Music, Movie, Open), Dynamism (Real-time, Fast-changing, Slow-changing, Static), Popularity (Web, Head, Torso, Tail) and Question Type (Simple, Simple with condition, Set, Comparison, Aggregation, Multi-hop, Post-processing, False premise). The strategies and performance of winning solutions vary significantly across different dimensions, as shown in Figure 53 from task 3 winning teams.

Domain dimension Since each domain has its own set of APIs, most winning solutions develop router to detect the topic domain of each question then use the results to select appropriate API calls and construct domain specific prompts accordingly.

Among the domains, Finance questions prove to be the most challenging. This is mainly because many financial questions are real-time or fast-changing (e.g., "can you tell me the opening stock price of landp on the last Friday (question asked on 02/28/2024, 07:58:48 PT)?"), thus requiring finding results precisely corresponding to the reference time. To tackle date-time parsing, APEX [30] leveraged regex rules and Python libraries (pytz and datetime) to parse absolute datetime objects. One other challenge is that Finance questions often require numeric reasoning. ElectricSheep [55] delegates numeric calculations to an external Python interpreter, which reduces hallucination from LLM.

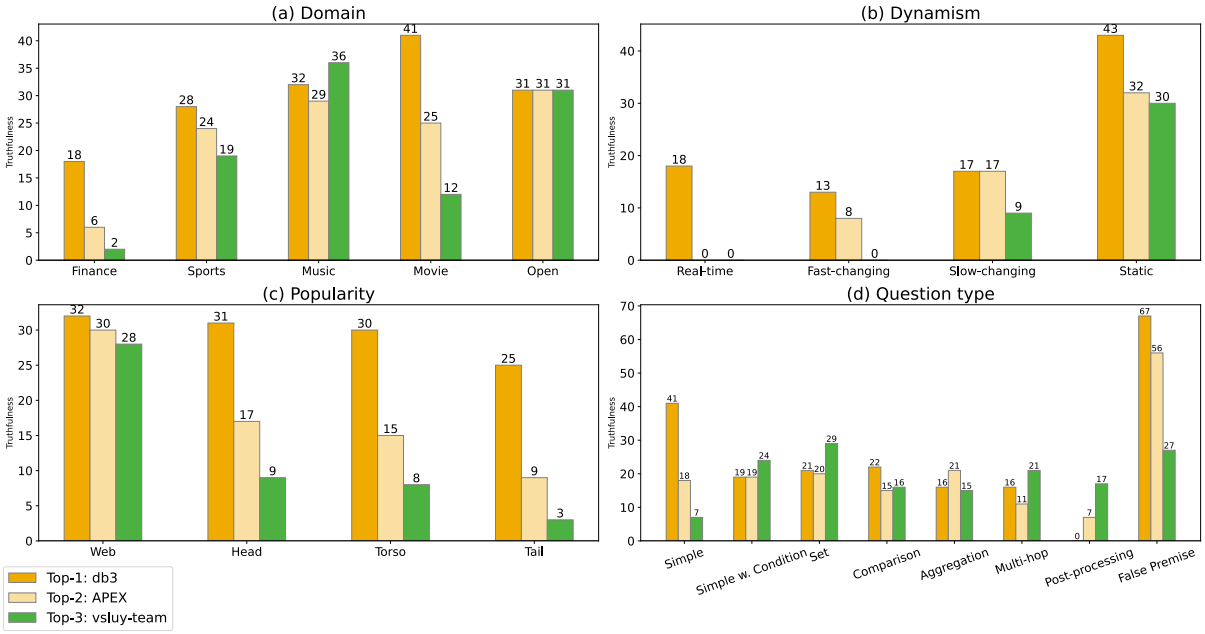

Figure 53: Human-evaluated Truthfulness scores from Task 3 winners on the private test set across the four question categories. The truthfulness scores reflect different strategies taken by different winning teams. First-place winner for all three tasks Db3 [49], as an example, put in significant efforts for improving the results on using the APIs, which gained them large margins in handling those KG-supported questions.

Dynamism dimension Overall, winning solutions perform better in static and slow-changing questions than fast-changing and real-time questions, mainly because of difficulties in temporal reasoning in both the retrieval step and the summarization step. Some teams even implemented solutions to blankly answer "I don't know" for all dynamic questions to reduce hallucination penalties. One promising approach was team db3's [49] "regularized" APIs (see Section D.2), which improved the APIs' expressiveness in handling temporal information.

Dynamic questions are an area where RAG should excel over vanilla LLM. The fact that teams struggle with dynamic questions suggests further opportunities in this area.

Popularity dimension In this dimension, the performance is largely correlated with entity popularity as expected, due to sparsity of information for torso to tail questions.

Question Type dimension In this dimension, comparison questions (e.g., "who started performing earlier, Adele or Ed Sheeran?"), post-processing questions (e.g., "How many days did Thurgood Marshall serve as a Supreme Court justice?"), and multi-hop questions (e.g., "what school does Lebron James's youngest son attend?") are the most difficult types, mainly because they require retrieving information of multiple entities and nontrivial reasoning over the retrieved results.

Team dRAGonRAnGers's [32] solution won first place for comparison questions in all three tasks. The authors hypothesized that their explicit chain-of-thought prompt contributed to the success by generating rationals for each entities involved in the comparison then synthesizing a final answer. Similarly, carefully engineered chain-of-thought prompts also improved reasoning for multi-hop and post-processing questions, as Team ElectricSheep [55] and bumblebee7 [56] demonstrated.

E Learnings from hosting the challenge

In this section, we reflect the experience from creating the benchmark, designing the challenge, and finally hosting the challenge and workshop, and summarize our learnings from this one year journey.

E.1 How can we design a fair, informative and feasible challenge?

We strive to create an informative and engaging challenge, but also need to take into consideration the fairness, accessibility, as well as time and budget constraints. We made several design decisions for the challenge, which turned out to be critical for its success. We narrate these design choices next.

Creating equally accessible retrieval contents. Providing equally accessible retrieval contents eliminates the unfair comparison induced by having access to different resources among the participants. A RAG system theoretically can retrieve from the universe of all public and proprietary information sources. Although it is interesting to test RAG systems in an end-to-end manner by allowing participants to use all accessible APIs, it inevitably creates an advantage for large companies or entities that have more resources. We therefore made a decision to provide equally accessible retrieval contents in the benchmark for the challenge, so that all systems can be evaluated against the same set of facilitating information, eliminating the potential unfairness introduced by having access to different amount of retrieval sources. This design choice, although not completely testing the retrieval of the RAG systems, still allows users to evaluate it by combining all the retrieved webpages as the retrieval pool.

Another advantage of using fixed retrieval contents is to be able to compare over time. Search engines evolve over time. People who use the benchmark questions later on can get better results just because the search engine improves its results. Meanwhile, CRAG retrieval content is available for experiments. Researchers who want to conduct a fair comparison do not need to reproduce the previous results, when the results were reported on CRAG retrieval content. Otherwise, they would need to re-implement existing SOTA systems to make sure the same version of search engines were used. This can be very consuming and even infeasible (if the system involves proprietary implementation or data). We believe providing equally accessible retrieval content can substantially facilitate the research for RAG, and are striving to extend this practice to future studies. See Section G for more discussions.

Setting model constraints. We put the constraints on which model can be used to make the challenge focus on RAG, instead of the base models. The participants can use vanilla or fine-tuned Llama 2 or Llama 3 models as long as the data were publicly available and were not generated from any proprietary model. This decision also ensures the results from the built systems are more comparable.

E.2 How can we ensure the cost is affordable?

There are a lot of cost involved in running a large-scale challenge. To ensure its sustainability, we carefully planned to control two major costs: computation and evaluation.

Setting submission limit and hardware constraints. The computation cost mainly comes from using GPUs to run the inference. We set limits on the number of submissions from each team during the challenge to make sure the computation cost can be controlled. The challenge allows up to six submissions for each team each week in phase 1, and up to six submissions for each team in total in phase 2.

We also selected a more affordable GPU option: the AWS G4dn.12xlarge instance equipped with 4 NVIDIA T4 GPUs with 16GB GPU memory as the supported hardware for the inference. These GPUs

have the limitation that they cannot run the Llama 2 70B or Llama 3 70B full precision models directly. However, the price of those more powerful GPUs were several times higher and would not bring much additional value. Therefore, we decided to choose the more economical option.

These submission limits, along with the selected hardware configurations, ensured that we can provide enough room for the participants to develop and test their solutions while keeping the computation cost under the budget.

Using auto-eval in Phase-1 and in Phase-2 initial selection. Although human-eval is the most comprehensive way for evaluating responses from LLMs, it would be too slow and expensive to support challenges that receive thousands of submissions. Therefore we made an early decision to develop auto-eval solutions for the challenge. Phase 1 of the challenge was purely relying on auto-eval to support the leaderboard. For phase 2, we first used auto-eval to select the top-15 teams for each task, and then employed human manual evaluation to determine the final winners. This design substantially reduced the time and expense (more than 100 times) needed for the evaluation.

E.3 How can we ensure the evaluation is reliable?

In order to guarantee the evaluation quality, we still need to ensure the auto-eval mechanism has high accuracy, which we discuss next.

Ensuring auto-eval quality. We solve the auto-evaluation problem with an LLM task, where the prompt asks the LLM to determine whether the generated answer is accurate, hallucinated, or missing based on the ground truth information. This is a simple version of the NLP task —consistency check.

We conducted extensive experiments to make sure the auto-eval solution can have high accuracy [52]. We have two observations when developing the auto-eval solution. First, it is critical to provide high quality ground truth answers for the auto-evaluator to work effectively. During the experiment, we observed that some examples may not have a unique correct answer (e.g., a question about height can be answered in either feet or meters.). Hence, we also provided alternative ground truth answers for the benchmark questions whenever needed. It turned out that these alternative answers were very helpful for reducing evaluation errors. Second, we don't need the most powerful models to serve as the auto-evaluator. Our experiment shows that the accuracy of using the Llama 3 (`llama-3-70B-instruct`) model and the ChatGPT (`gpt-3.5-turbo`) are 99% and 94.5% respectively (for predicting accurate, missing and hallucinated responses) [52]. See Table 34 for more detailed results.

In the final evaluation, among the nine top-3 winning teams selected by manual evaluation, six of them were also selected by our auto-evaluation. Generally speaking, using LLM-as-a-Judge is a common practice in evaluating language models' performance. Procedures for measuring and resolving the self-enhancement bias, position bias [57] and preventing prompt attacks play vital roles in ensuring the reliability of the evaluation results.

Preventing evaluation attacks. We found three decisions very helpful for protecting the evaluation from hacks. The first one was to keep a held-out test set for the final competition (phase 2). We initially ran the leaderboard in phase 1 using the released public test set, but soon found the leaderboard was filled with 100% accuracy. This was because some teams tried to test the evaluation system and created a map with (query, answer) pairs based on the released data, from which they looked up answers for the test queries during the submission. This issue was remedied by replacing the leaderboard test set with a subset of the held-out dataset. The second decision was to hold out the exact prompt used in the auto-eval to avoid prompt attack. We released an example evaluation prompt to illustrate how the auto-eval works in the starter kit, and later on received questions from the challenge forum calling out

	F1 Score			Accuracy
Model	**Correct**	**Missing**	**Hallucinated**	
ChatGPT-3.5 Turbo	92.0%	100.0%	92.0%	94.5%
Llama 3.0 8B Instruct	94.6%	100.0%	90.7%	95.3%
Llama 3.0 70B Instruct	98.9%	100.0%	97.9%	99.0%
Llama 3.1 70B Instruct	98.2%	100.0%	96.8%	98.4%

Table 34: Auto-eval performance for using different models as the evaluator. The first three columns show the F1 scores on the accurate, missing and hallucinated responses. The last columns shows the overall accuracy on all examples in the CRAG public test set. Llama 3.0 70B and Llama 3.1 70B models achieve accuracy around 99%. Llama 3.0 8B, although being much smaller, attains accuracy 95.3%, on par with ChatGPT-3.5 Turbo. All models have 100% on the missing answers because we require all missing answers to be answered as "I don't know", and we use exact match to judge for missing responses during the evaluation.

ways to hack the prompt for obtaining high scores. During the final competition, we also noticed some submissions that tried to fool the auto-evaluator achieved very high scores. Although the final winners were selected by the manual evaluation, recall that a team needed to be among the top 15 in the auto-eval to be eligible for human-eval. Keeping the evaluation prompt private significantly reduced the risk of prompt attack and also reduced the overhead needed for manual checking after the evaluation. The third one was to disable network connection during the inference. The choice of providing pre-fetched retrieval contents (discussed in Section E.1) allowed us to avoid using live web connection during the challenge. This also helped reduce the risk of leaking the private test data during the evaluation stage.

F Related work

F.1 RAG Benchmarks

We compare and highlight the strengths and limitations of existing retrieval-augmented generation (RAG) benchmarks across several critical dimensions, as demonstrated in table 35 in CRAG [52]. QALD-10 [46] focuses primarily on knowledge graph search over Wikidata but does not incorporate web retrieval, limiting its capacity to test models in unstructured environments. It lacks mock APIs or dynamic question capabilities, reducing its ability to simulate real-world, evolving knowledge scenarios. MS MARCO [4], while widely used for open-domain question answering, primarily emphasizes passage retrieval and lacks integration with knowledge graphs. It also fails to test for long-tail facts, as it predominantly handles popular, factoid-based information sourced from common queries, limiting its coverage of less frequent, tail facts. Natural Questions [19] focuses on more complex and long-form queries but remains heavily constrained to Wikipedia-based knowledge and lacks dynamic or API-driven retrieval. RGB [7], on the other hand, introduces a focus on long-tail facts but does not fully test retrieval beyond specific domains, missing integration with KG searches or dynamic question handling. FreshLLMs [47] excels in testing models' ability to retrieve and update real-time knowledge but remains narrow, lacking domain diversity and focus on structured retrieval such as knowledge graphs.

Despite having smaller question size than MS MARCO and NQ, CRAG [52] stands out by combining web retrieval, knowledge graph search, and mock APIs, simulating diverse retrieval environments. It goes beyond Wikipedia, tackling dynamic questions and ensuring comprehensive coverage of both torso and tail facts across multiple domains, making it a more robust and versatile benchmark for evaluating

Table 35: Comparing CRAG to existing benchmarks for factual question answering.

Benchmark	Web retrieval	KG search	Mock API	Dynamic question	Torso and tail facts	Beyond Wikipedia	Question size
QALD-10 [46]	✗	✓	✗	✗	✗	✗	0.8K
MS MARCO [4]	✓	✗	✗	not explicitly	not explicitly	✓	100K
NQ [19]	✓	✗	✗	not explicitly	not explicitly	✗	323K
RGB [6]	✓	✗	✗	✗	✗	✓	1K
FreshLLM [47]	✗	✗	✗	✓	✗	✓	0.6K
CRAG [52]	✓	✓	✓	✓	✓	✓	4.4K

next-generation RAG systems.

F.2 RAG Systems

RAG systems have evolved significantly over time, focusing on improving large language models (LLMs) by integrating retrieval mechanisms for enhanced knowledge access. Talmor et al. [41] pioneered the use of the web as a knowledge base to answer complex questions, emphasizing retrieval for reasoning tasks. Lewis et al. [21] formally introduced RAG, combining dense retrieval with generative models to solve knowledge-intensive NLP tasks, setting a strong foundation for subsequent models. REALM [14] improved this by incorporating retrieval into pre-training, making retrieval part of both the learning and fine-tuning process, which was further extended by Fusion-in-Decoder (FiD) [17], fusing multiple retrieved documents into the generation process for improved question answering. Mallen et al. [25] explored the effectiveness of parametric and non-parametric memory mechanisms to determine when models should rely on internal versus external knowledge. Similarly, Sun et al. [39] investigated whether LLMs could replace traditional knowledge graphs, assessing how well LLMs store and retrieve structured knowledge. QA-GNN [54] integrated retrieval-augmented methods with reasoning from knowledge graphs, combining structured and unstructured knowledge for better question answering. Meanwhile, Mallen et al. [25] focused on trust in LLMs, analyzing the limitations of parametric knowledge and advocating for non-parametric memory integration. Recently, FreshLLMs [47] refreshed LLM knowledge using search engine augmentation to maintain up-to-date responses. These works collectively contribute to a growing body of research that refines LLM performance by bridging the gap between parametric knowledge and external, retrievable information.

F.3 RAG System Evaluation

In recent developments for evaluating RAG systems, multiple frameworks have proposed solutions aimed at addressing key challenges like retrieval relevance, truthfulness, and the efficiency of the evaluation process. ARES [37] offers an automated evaluation approach by generating synthetic data and fine-tuning lightweight language models, allowing for scalable assessments of retrieval relevance, answer faithfulness and answer relevance. However, ARES may face issues with domain adaptation, as its synthetic data may not always represent the nuances of diverse real-world datasets. RAGAS [12] introduces a reference-free evaluation method, providing metrics for context relevance without relying on ground truth annotations, reducing human labeling costs. Nevertheless, its heuristic-based measures may struggle with capturing deeper semantic nuances and can exhibit biases in complex scenarios. CoFE-RAG [23] performs an exhaustive evaluation across the entire RAG pipeline, employing multi-granularity keyword-based assessment of the retrieved context. RAG-QA Arena [15] proposes a pairwise preference evaluation framework with truthfulness as the primary criterion, leveraging ground truth to gauge the evaluation

system quality.

F.4 RAG Competitions

More recently, the TREC 2024 RAG Track is proposed, featuring Ragnarök [33], an open-source, end-to-end evaluation framework with MS MARCO V2.1 collection and industrial baselines like OpenAI's GPT-4o and Cohere's Command R+. The framework provides a web-based interface for benchmarking pairwise RAG systems through a RAG battle arena.

Various domain-specific competitions, such as the Zindi RAG challenge [2] for public services, the FinanceRAG competition [1], and the Trustbit Enterprise RAG Challenge [45], push the boundaries of RAG applications. These challenges emphasize tailored RAG systems that retrieve domain-specific information and generate context-aware responses, demonstrating the potential of RAG to enhance service delivery in public administration, financial analysis, and business document comprehension.

G Discussion and future work

The KDD Cup CRAG 2024 competition established a benchmark for evaluating Retrieval-Augmented Generation (RAG) systems with three challenging tasks: retrieval summarization, knowledge graph and web retrieval, and end-to-end RAG. Questions covered various domains, dynamism levels, types, and entity popularity. Key learnings from hosting the challenge (Section E) emphasized the need for standardized retrieval content and retrieval access, controlled model constraints, maintaining a private test set, and mixed automated and human evaluation strategies to ensure fairness and manage cost.

CRAG highlighted persistent challenges in RAG systems. First, reducing retrieval noises from the web and from KG is critical as irrelevant or contradicting information degrades answer quality. Second, handling real-time or fast-changing questions remains difficult due to the need for nuanced temporal reasoning. Third, questions that require aggregation, multi-hop reasoning or post-processing are challenging as they require complex logic to integrate knowledge across multiple sources. Lastly, reducing hallucinations in summarization remains essential. To address the above challenges, winning teams leveraged a variety of solutions (Section D). For example, developing multi-step pipelines to parse, chunk, and rank web contents reliably, creating regularized APIs to enhance KG retrieval expressiveness, incorporating external libraries to improve temporal reasoning, leveraging chain-of-thought techniques to break down complex queries, and fine-tuning LLM and implementing confidence estimation component to reduce hallucinations. The practical applications of these findings have significant implications for improvements in industry-grade RAG systems.

The CRAG competition highlights several promising directions for future RAG research: Enhancing web retrieval quality via better handling of structured and semi-structured data, improving temporal reasoning for real-time queries, and refining methods for multi-hop and aggregation tasks can improve response reliability. Additionally, targeted fine-tuning may be essential to effectively reduce hallucinations. Looking ahead, future CRAG-style competitions could expand the question sets to include multimodal, multi-turn questions, integrating text with images or structured data for a more realistic RAG environment. As researchers pursue these directions, they will also need to address key scalability challenges in building RAG systems that can handle real-time and multimodal data, including ensuring data consistency across diverse sources and achieving low-latency processing of real-time multimodal data. By tackling these challenges, researchers can unlock the full potential of RAG systems and improve response reliability.

References

[1] Financerag challenge: Retrieval-augmented generation (rag) for financial documents. https://finance-rag.com/. Accessed: 2024-11-10.

[2] Retrieval-augmented generation (rag) for public services and administration tasks. https://zindi.africa/competitions/retrieval-augmented-generation-rag-for-public-services-and-administration-tasks. Accessed: 2024-11-10.

[3] AI@Meta. Llama 3 model card. 2024.

[4] P. Bajaj, D. Campos, N. Craswell, L. Deng, J. Gao, X. Liu, R. Majumder, A. McNamara, B. Mitra, T. Nguyen, et al. Ms marco: A human generated machine reading comprehension dataset. ArXiv preprint, abs/1611.09268, 2016.

[5] A. Barbaresi. Trafilatura: A web scraping library and command-line tool for text discovery and extraction. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations, pages 122–131, 2021.

[6] J. Chen, H. Lin, X. Han, and L. Sun. Benchmarking large language models in retrieval-augmented generation. arXiv preprint arXiv:2309.01431, 2023.

[7] J. Chen, H. Lin, X. Han, and L. Sun. Benchmarking large language models in retrieval-augmented generation. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, pages 17754–17762, 2024.

[8] J. Chen, S. Xiao, P. Zhang, K. Luo, D. Lian, and Z. Liu. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. arXiv preprint arXiv:2402.03216, 2024.

[9] X. Chen, L. Wang, W. Wu, Q. Tang, and Y. Liu. Honest ai: Fine-tuning "small" language models to say "i don't know", and reducing hallucination in rag. 2024 KDD Cup Workshop for Retrieval Augmented Generation, 2024.

[10] Z. Chen, Z. Gu, L. Cao, J. Fan, S. Madden, and N. Tang. Symphony: Towards natural language query answering over multi-modal data lakes. In CIDR, 2023.

[11] M. DeHaven. Marags: A multi-adapter system for multi-task retrieval augmented generation question answering. 2024 KDD Cup Workshop for Retrieval Augmented Generation, 2024.

[12] S. Es, J. James, L. Espinosa Anke, and S. Schockaert. RAGAs: Automated evaluation of retrieval augmented generation. In N. Aletras and O. De Clercq, editors, Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations, pages 150–158, St. Julians, Malta, Mar. 2024. Association for Computational Linguistics.

[13] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, Q. Guo, M. Wang, and H. Wang. Retrieval-augmented generation for large language models: A survey. 2024.

[14] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang. Retrieval augmented language model pre-training. In Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event, volume 119 of Proceedings of Machine Learning Research, pages 3929–3938. PMLR, 2020.

[15] R. Han, Y. Zhang, P. Qi, Y. Xu, J. Wang, L. Liu, W. Y. Wang, B. Min, and V. Castelli. Rag-qa arena: Evaluating domain robustness for long-form retrieval augmented question answering. arXiv preprint arXiv:2407.13998, 2024.

[16] L. He, R. Li, S. Shen, J. Lu, L. Zhu, Y. Su, and Z. Huang. A simple yet effective retrieval-augmented generation framework for the meta KDD cup 2024. In 2024 KDD Cup Workshop for Retrieval Augmented Generation, 2024.

[17] G. Izacard and E. Grave. Leveraging passage retrieval with generative models for open domain question answering. In Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, pages 874–880, Online, 2021. Association for Computational Linguistics.

[18] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung. Survey of hallucination in natural language generation. ACM Comput. Surv., 55(12), mar 2023.

[19] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, K. Toutanova, L. Jones, M. Kelcey, M.-W. Chang, A. M. Dai, J. Uszkoreit, Q. Le, and S. Petrov. Natural questions: A benchmark for question answering research. Transactions of the Association for Computational Linguistics, 7:452–466, 2019.

[20] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. tau Yih, T. Rocktäschel, S. Riedel, and D. Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021.

[21] P. S. H. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, and D. Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, 2020.

[22] V. Liévin, C. E. Hother, A. G. Motzfeldt, and O. Winther. Can large language models reason about medical questions? Patterns, 5(3), 2024.

[23] J. Liu, R. Ding, L. Zhang, P. Xie, and F. Huang. Cofe-rag: A comprehensive full-chain evaluation framework for retrieval-augmented generation with enhanced data diversity. arXiv preprint arXiv:2410.12248, 2024.

[24] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. ACM Computing Surveys, 55(9):1–35, 2023.

[25] A. Mallen, A. Asai, V. Zhong, R. Das, D. Khashabi, and H. Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. ArXiv preprint, abs/2212.10511, 2022.

[26] V. Mavroudis. Langchain. 2024.

[27] ms-marco MiniLM-L-12-v2.

[28] J. Ni, G. H. Abrego, N. Constant, J. Ma, K. B. Hall, D. Cer, and Y. Yang. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. arXiv preprint arXiv:2108.08877, 2021.

[29] OpenAI. ChatGPT. https://openai.com/index/chatgpt/, 2023. Accessed: 2024-06-04.

[30] J. Ouyang, Y. Luo, M. Cheng, D. Wang, S. Yu, Q. Liu, and E. Chen. Revisiting the solution of meta kdd cup 2024: Crag. 2024 KDD Cup Workshop for Retrieval Augmented Generation, 2024.

[31] A. Panickssery, S. R. Bowman, and S. Feng. Llm evaluators recognize and favor their own generations. ArXiv preprint, abs/2404.13076, 2024.

[32] S. Park, J. Seok, J. Lee, J. Yun, and W. Lee. KDD cup meta CRAG 2024 technical report: Three-step question-answering framework. In 2024 KDD Cup Workshop for Retrieval Augmented Generation, 2024.

[33] R. Pradeep, N. Thakur, S. Sharifymoghaddam, E. Zhang, R. Nguyen, D. Campos, N. Craswell, and J. Lin. Ragnar\" ok: A reusable rag framework and baselines for trec 2024 retrieval-augmented generation track. arXiv preprint arXiv:2406.16828, 2024.

[34] V. Rawte, S. Chakraborty, A. Pathak, A. Sarkar, S. Tonmoy, A. Chadha, A. P. Sheth, and A. Das. The troubling emergence of hallucination in large language models–an extensive definition, quantification, and prescriptive remediations. arXiv preprint arXiv:2310.04988, 2023.

[35] L. Richardson. Beautiful soup documentation, 2007.

[36] S. Robertson, H. Zaragoza, et al. The probabilistic relevance framework: Bm25 and beyond. Foundations and Trends® in Information Retrieval, 3(4):333–389, 2009.

[37] J. Saad-Falcon, O. Khattab, C. Potts, and M. Zaharia. Ares: An automated evaluation framework for retrieval-augmented generation systems, 2023.

[38] R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1715–1725, Berlin, Germany, 2016. Association for Computational Linguistics.

[39] K. Sun, Y. E. Xu, H. Zha, Y. Liu, and X. L. Dong. Head-to-tail: How knowledgeable are large language models (llm)? aka will llms replace knowledge graphs? ArXiv preprint, abs/2308.10168, 2023.

[40] Y. Sun, H. Xin, K. Sun, Y. E. Xu, X. Yang, X. L. Dong, N. Tang, and L. Chen. Are large language models a good replacement of taxonomies? Proc. VLDB Endow., 17(11):2919–2932, aug 2024.

[41] A. Talmor and J. Berant. The web as a knowledge-base for answering complex questions. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 641–651, New Orleans, Louisiana, 2018. Association for Computational Linguistics.

[42] N. Tang, C. Yang, J. Fan, L. Cao, Y. Luo, and A. Y. Halevy. Verifai: Verified generative AI. In CIDR, 2024.

[43] Y. Taya, D. Ito, S. Maeda, and Y. Hamano. RAG approach enhanced by category classification with BERT. In 2024 KDD Cup Workshop for Retrieval Augmented Generation, 2024.

[44] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[45] Trustbit. Enterprise rag challenge, 2024. Accessed: 2024-11-10.

[46] R. Usbeck, X. Yan, A. Perevalov, L. Jiang, J. Schulz, A. Kraft, C. Möller, J. Huang, J. Reineke, A.-C. Ngonga Ngomo, et al. Qald-10–the 10th challenge on question answering over linked data. Semantic Web, (Preprint):1–15, 2023.

[47] T. Vu, M. Iyyer, X. Wang, N. Constant, J. Wei, J. Wei, C. Tar, Y.-H. Sung, D. Zhou, Q. Le, et al. Freshllms: Refreshing large language models with search engine augmentation. ArXiv preprint, abs/2310.03214, 2023.

[48] C. Wu, T. Shen, R. Yan, H. Wang, Z. Liu, Z. WANG, D. Lian, and E. Chen. Knowledge graph integration and self-verification for comprehensive retrieval-augmented generation. In 2024 KDD Cup Workshop for Retrieval Augmented Generation, 2024.

[49] Y. Xia, J. Chen, and J. Gao. Winning solution for meta kdd cup' 24. 2024 KDD Cup Workshop for Retrieval Augmented Generation, 2024.

[50] S. Xiao, Z. Liu, P. Zhang, and N. Muennighoff. C-pack: Packaged resources to advance general chinese embedding, 2023.

[51] W. Xie, X. Liang, Y. Liu, K. Ni, H. Cheng, and Z. Hu. Weknow-rag: An adaptive approach for retrieval-augmented generation integrating web search and knowledge graphs, 2024.

[52] X. Yang, K. Sun, H. Xin, Y. Sun, N. Bhalla, X. Chen, S. Choudhary, R. D. Gui, Z. W. Jiang, Z. Jiang, et al. Crag–comprehensive rag benchmark. ArXiv preprint, abs/2406.04744, 2024.

[53] M. Yasunaga, H. Ren, A. Bosselut, P. Liang, and J. Leskovec. QA-GNN: Reasoning with language models and knowledge graphs for question answering. Association for Computational Linguistics, 2021.

[54] M. Yasunaga, H. Ren, A. Bosselut, P. Liang, and J. Leskovec. QA-GNN: Reasoning with language models and knowledge graphs for question answering. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 535–546, Online, 2021. Association for Computational Linguistics.

[55] Y. Yuan, C. Liu, J. Yuan, G. Sun, S. Li, and M. Zhang. A hybrid rag system with comprehensive enhancement on complex reasoning. 2024 KDD Cup Workshop for Retrieval Augmented Generation, 2024.

[56] J. Zhao and X. Liu. TCAF: a multi-agent approach of thought chain for retrieval augmented generation. In 2024 KDD Cup Workshop for Retrieval Augmented Generation, 2024.

[57] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. Advances in Neural Information Processing Systems, 36:46595–46623, 2023.

[58] Y. Zhu, S. Du, B. Li, Y. Luo, and N. Tang. Are large language models good statisticians? In NeurIPS, 2024.