CRAG – Comprehensive RAG Benchmark

Xiao Yang*[1], Kai Sun*[1], Hao Xin*[3], Yushi Sun*[3], Nikita Bhalla[1], Xiangsen Chen[4], Sajal Choudhary[1], Rongze Daniel Gui[1], Ziran Will Jiang[1], Ziyu Jiang[4], Lingkun Kong[1], Brian Moran[1], Jiaqi Wang[1], Yifan Ethan Xu[1], An Yan[1], Chenyu Yang[4], Eting Yuan[1], Hanwen Zha[1], Nan Tang[3,4], Lei Chen[3,4], Nicolas Scheffer[1], Yue Liu[1], Nirav Shah[1], Rakesh Wanga[1], Anuj Kumar[1], Wen-tau Yih[2], and Xin Luna Dong[1]

[1]Meta Reality Labs, [2] FAIR, Meta, [3] HKUST, [4] HKUST (GZ)

Abstract

Retrieval-Augmented Generation (RAG) has recently emerged as a promising solution to alleviate Large Language Model (LLM)'s deficiency in lack of knowledge. Existing RAG datasets, however, do not adequately represent the diverse and dynamic nature of real-world Question Answering (QA) tasks. To bridge this gap, we introduce the **Comprehensive RAG Benchmark (CRAG)**, a factual question answering benchmark of 4,409 question-answer pairs and mock APIs to simulate web and Knowledge Graph (KG) search. CRAG is designed to encapsulate a diverse array of questions across five domains and eight question categories, reflecting varied entity popularity from popular to long-tail, and temporal dynamisms ranging from years to seconds. Our evaluation of this benchmark highlights the gap to fully trustworthy QA. Whereas most advanced LLMs achieve $\leqslant 34\%$ accuracy on CRAG, adding RAG in a straightforward manner improves the accuracy only to 44%. State-of-the-art industry RAG solutions only answer 63% of questions without any hallucination. CRAG also reveals much lower accuracy in answering questions regarding facts with higher dynamism, lower popularity, or higher complexity, suggesting future research directions. The CRAG benchmark laid the groundwork for a KDD Cup 2024 challenge and attracted thousands of participants and submissions. We commit to maintaining CRAG to serve research communities in advancing RAG solutions and general QA solutions. CRAG is available at `https://github.com/facebookresearch/CRAG/`.

1 Introduction

Large Language Models (LLMs) have transformed the landscape of Natural Language Processing (NLP) tasks, especially in Question Answering (QA) [20, 22, 38, 39]. Despite the advancements, the issue of hallucination persists as a significant challenge; LLMs may generate answers that lack factual accuracy or grounding [14, 27, 30, 32]. Studies have shown that GPT-4's accuracy in answering questions referring to slow-changing or fast-changing facts is below 15% [36]; even for stable (never-changing) facts, GPT-4's accuracy in answering questions referring to torso-to-tail (less popular) entities is below 35% [29]. Overcoming hallucinations thus becomes a priority in building reliable QA systems [13, 14].

Retrieval-Augmented Generation (RAG) [6, 8, 12, 19] has recently emerged as a promising solution to alleviate LLM's deficiency in lack of knowledge and attracted a lot of attention from both academia research and industry. Given a question, a RAG system searches external sources to retrieve relevant information and then provides grounded answers [7, 12, 19] (see Figure 1 for an illustration). Despite

*Equal contribution. Correspondence to: Xiao Yang (xiaoyangfb@meta.com).

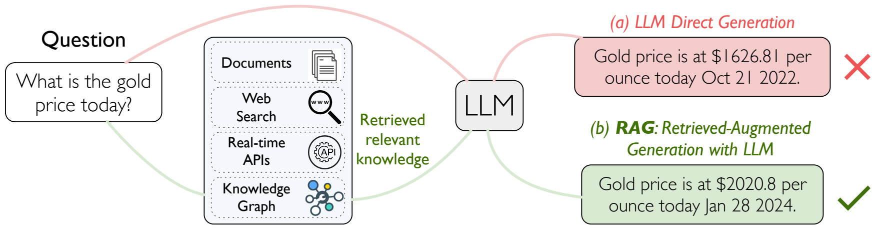
Question
What is the gold price today?
Documents
Web Search
Real-time APIs
Knowledge Graph
Retrieved relevant knowledge
LLM
(a) LLM Direct Generation
Gold price is at $1626.81 per ounce today Oct 21 2022.
(b) RAG: Retrieved-Augmented Generation with LLM
Gold price is at $2020.8 per ounce today Jan 28 2024.

Figure 1: QA using LLMs (a) without RAG vs. (b) with RAG.

its potential, RAG still faces many challenges, such as selecting the most relevant information, reducing question answering latency, and synthesizing information to answer complex questions.

A comprehensive benchmark is currently missing to advance continued research efforts in this field. Our goal is to build a benchmark that can provide a holistic view of the important capabilities and fast but reliable evaluation for RAG to propel the area forward. What is a good benchmark for QA over LLMs? We consider five critical features.

1. **Realism:** First and foremost, a good benchmark shall best reflect real use cases. In other words, a solution that achieves high metrics in the benchmark shall also perform very well in real scenarios. For example, the questions in a RAG benchmark shall be similar to questions people ask in real-world QA scenarios.
2. **Richness:** The benchmark shall contain a diverse set of instance types, covering both common use cases and some complex and advanced use cases, to represent real-world challenges and reveal possible limitations of existing solutions.
3. **Insightfulness:** The benchmark shall allow for an easy understanding of performance on different slices of the data, reflecting the capability of the solution in addressing different types of challenges.
4. **Reliability:** The benchmark shall allow reliable assessment of metrics: the ground truths shall be accurate; the metrics shall well capture the performance of the model; the evaluation shall be easy and reliable, and the computed metrics shall hold statistical significance.
5. **Longevity:** Finally, to enable research and experimental comparison in a long term, the scenarios and the data in the benchmark shall not quickly expire and ideally shall be refreshed and improved over time.

We strive to create a benchmark that have all of the aforementioned features, and we call it *CRAG – Comprehensive benchmark for RAG*. Our work makes three contributions.

Our first contribution is the dataset itself (Section 3). CRAG contains a *rich* set of 4,409 QA pairs from five domains: *Finance, Sports, Music, Movie,* and *Open domain*. In addition to simple-fact questions (asking for an attribute of an entity), CRAG contains seven types of complex questions to cover real user queries: questions with *Conditions*, *Comparison* questions, *Aggregation* questions, *Multi-hop* questions, *Set queries*, *Post-processing-heavy* questions, and *False-premise* questions. CRAG reflects varied entity popularity from popular to long-tail and temporal spans ranging from seconds to years, allowing easy deep dives for *insights*. As we generated the questions, we referred to smart assistant use cases to make sure the questions are *realistic*, paraphrased the questions to increase the *diversity* of expressions, and manually verified ground truths to ensure *reliability*.

In addition to QA pairs, CRAG provides mock APIs to simulate retrieval from a diverse range of available information. This includes up to 50 full HTML pages for each question returned from a real-world search engine—the Brave Search API [5], and mock KGs with 2.6 million entities. For the mock KGs, we deliberately make sure that the retrieval candidates reflect noises in a *realistic* setting.

Our second contribution is the evaluation mechanism to allow for *reliable* comparisons. We designed 3 tasks to test different components in RAG solutions: retrieval summarization, knowledge graph and web retrieval, and end-to-end retrieval-augmented generation (Section 2). Instead of computing the percentage of correctly answered questions, our score system distinguishes hallucinated answers and missing answers, and gives the former a higher penalty as it can be more harmful to ruin user

Table 1: 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 [35]	✗	✓	✗	✗	✗	✗	0.8K
MS MARCO [4]	✓	✗	✗	not explicitly	not explicitly	✓	100K
NQ [18]	✓	✗	✗	not explicitly	not explicitly	✗	323K
RGB [6]	✓	✗	✗	✗	✗	✓	1K
FreshLLM [36]	✗	✗	✗	✓	✗	✓	0.6K
CRAG	✓	✓	✓	✓	✓	✓	4.4K

trust. We also design an effective automatic evaluation mechanism to allow for fast evaluations and iterations (Section 4).

Our third contribution is a comprehensive evaluation of straightforward RAG solutions and industry state-of-the-art solutions on RAG (Section 5). Whereas most advanced LLMs achieve $\leqslant 34\%$ accuracy on CRAG, adding RAG in a straightforward manner improves the accuracy only to 44%. State-of-the-art industry RAG solutions answer only 63% questions without any hallucination, still having much lower accuracy in answering questions regarding facts with higher dynamism, lower popularity, or higher complexity. These evaluations serve two roles: first, they demonstrate that CRAG has appropriate level of difficulty and allows insights drawn from different dimensions of diversities the benchmark has incorporated; second, they highlight the gaps and research directions to a fully trustworthy QA system.

The CRAG benchmark laid the groundwork for a KDD Cup 2024 challenge[2], has attracted thousands of participants and submissions within the first 50 days of the competition. We commit to maintaining CRAG to serve research communities in advancing RAG solutions and general QA solutions.

Comparison with existing benchmarks. Table 1 compares CRAG with existing benchmarks for factual question answering. Traditional QA benchmarks such as Natural Questions (NQ) [18], TriviaQA [16], MS MARCO [4], and QALD-10 [35] have advanced QA in the past decade but consider only web retrieved *or* KG retrieved contents, and do *not* adequately represent the diverse and dynamic challenges that RAG is facing. New benchmarks for LLM or RAG usually target certain capabilities of the QA system. Researchers created benchmarks to evaluate how well the systems can answer simple knowledge questions [23, 29, 30] and handle more advanced scenarios. These include answering questions with changing answers [36], integrating information from multiple documents [6], addressing multi-hop questions [33], and answering questions with long texts [26]. Moreover, traditional QA benchmarks usually adopt matching-based metrics such as ROUGE [21] or F1 to evaluate the quality of the responses [16, 18]. These metrics, although working well for extractive methods, are known to not perform very effectively for LLMs that generate free-form responses [11].

Despite CRAG being smaller than MS MARCO and NQ, it offers several distinctive advantages: comprehensive coverage, realistic testing with mock APIs, dynamic question handling, diverse fact popularity, extensive content beyond Wikipedia, and fast yet reliable evaluation. These features make CRAG a robust and versatile benchmark for testing RAG systems and broadly QA systems, providing a shared testbed to evaluate how these systems handle real-world, dynamic, and diverse information retrieval and synthesis challenges for reliable LLM-based question answering.

2 Problem Description

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 provide useful information to answer the question without adding any hallucination.

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

Table 2: Definition of CRAG question types.

Question type	**Definition**
Simple	Questions asking for simple facts that are unlikely to change overtime, such as the birth date of a person and the authors of a book.
Simple w. Condition	Questions asking for simple facts with some given conditions, such as stock prices on a certain date and a director's recent movies in a certain genre.
Set	Questions that expect a set of entities or objects as the answer (e.g., "*what are the continents in the southern hemisphere?*").
Comparison	Questions that compare two entities (e.g., "*who started performing earlier, Adele or Ed Sheeran?*").
Aggregation	Questions that require aggregation of retrieval results to answer (e.g., "*how many Oscar awards did Meryl Streep win?*").
Multi-hop	Questions that require chaining multiple pieces of information to compose the answer (e.g., "*who acted in Ang Lee's latest movie?*").
Post-processing-heavy	Questions that need reasoning or processing of the retrieved information to obtain the answer (e.g., "*how many days did Thurgood Marshall serve as a Supreme Court justice?*").
False Premise	Questions that have a false preposition or assumption (e.g., "*What's the name of Taylor Swift's rap album before she transitioned to pop?*" (Taylor Swift has not yet released any rap album)).

We designed three tasks. They share the same set of (question, answer) pairs but differ in the external data accessible for retrieval to augment QA. Here, we provide the content that can be leveraged in QA to ensure fair comparisons. We describe how we generated the data in Section 3.

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 are more likely to provide necessary information to answer the question, but meanwhile are more likely to contain noises. 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 RAG systems. The only component of a RAG system not covered by these tasks is search retrieval. One may easily extend the tasks to use all 220K webpages in our benchmark as the search corpus for fully end-to-end testing.

3 Dataset Description

CRAG contains two parts of data: the QA pairs and the contents for retrieval. We now describe each part of the data. Data generation details can be found in Appendix A.1.1–A.1.6.

Table 3: The numbers and percentages (%, in parenthesis) of questions for each category of dynamism, decided manually. Finance and Sports domain have the most Real-time and Fast-changing questions.

Dynamism	Finance	Sports	Music	Movie	Open	Total
Real-time	434 (42)	0 (0)	2 (0)	0 (0)	1 (0)	437 (10)
Fast-changing	204 (20)	275 (33)	40 (6)	17 (2)	28 (4)	564 (13)
Slow-changing	183 (18)	215 (26)	152 (24)	253 (22)	204 (26)	1,007 (23)
Static	218 (21)	343 (41)	430 (69)	855 (76)	555 (70)	2,401 (54)
All	1,039	833	624	1,125	788	4,409

Table 4: The number and percentages (%, in parenthesis) of questions for each question type, decided manually. Simple and simple with condition questions constitute 43% of all questions.

Question type	Finance	Sports	Music	Movie	Open	Total
Simple	466 (45)	23 (3)	112 (18)	519 (46)	85 (11)	1,205 (27)
Simple w. condition	113 (11)	250 (30)	92 (15)	112 (10)	122 (15)	689 (16)
Set	48 (5)	93 (11)	72 (12)	104 (9)	86 (11)	403 (9)
Comparison	146 (14)	85 (10)	102 (16)	105 (9)	98 (12)	536 (12)
Aggregation	69 (7)	137 (16)	96 (15)	71 (6)	116 (15)	489 (11)
Multi-hop	86 (8)	64 (8)	55 (9)	90 (8)	87 (11)	382 (9)
Post-processing heavy	26 (3)	24 (3)	26 (4)	28 (2)	76 (10)	180 (4)
False Premise	85 (8)	157 (19)	69 (11)	96 (9)	118 (15)	525 (12)
All	1,039	833	624	1,125	788	4,409

3.1 Question answering pairs

CRAG covers five domains: Finance, Sports, Music, Movie, and Open domain, and eight types of questions, all in English. The question types are listed in Table 2. We constructed the question-answer pairs both from underlying KGs and web contents.

QA pairs constructed from KGs. We constructed QA pairs from KGs by collecting a set of entities based on publicly available data and then creating 600+ question templates based on selected entity types and relations. Next, we sampled entities with different popularities (head, torso and tail) following [29] from the KGs to fill in the templates and generate the full question and answer.

QA pairs constructed from web contents. We asked annotators to write down possible questions that users may ask (e.g., "*most popular action movies in 2023*") and created QA pairs from the corresponding web search results.

Using the above methods, we collected 2,425 *Web Questions* and 1,984 *KG Questions*, with 661, 658, and 665 *KG Questions* containing *head*, *torso*, and *tail* entities respectively. Tables 3 and 4 summarize the distribution of the questions across different dimensions. The size of each dimension slice (e.g., fast-changing facts) allows us to get metrics with $< 5\%$ margin-of-error (with 95% confidence level) for most of the cases. The dynamism distribution roughly reflects the nature of the domain (e.g., much more real-time questions for *Finance* than for other domains). See Appendix A.1.2 for the definition of the dynamism categories.

3.2 Contents for retrieval

We included two types of contents for retrieval to simulate the practical scenario for RAG: web search and KG search.

Web search results. For each question, we used the question text as the search query and stored up to 50 HTML pages from the Brave search API [5]. See Table 8 in Appendix A.1.5 for an example.

We estimated the web search recall with a heuristic-based method: first check whether the ground truth answer URL was found among the pages; if not, decide whether the fact in the ground truths is contained in the page snippet or content with an LLM. In particular, we pass the question, ground truth, and the pages to Llama 3 70B Instruct and ask it to judge if the context is sufficient to answer the question.

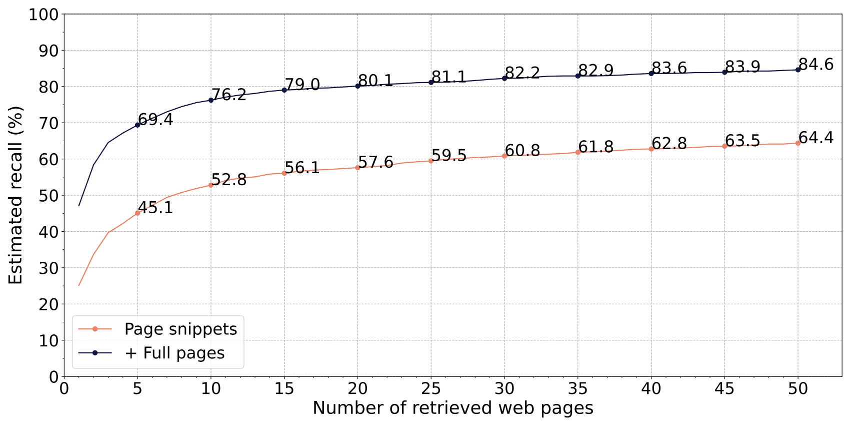

Figure 2: For 85% of CRAG questions, the web search results are estimated to contain the ground truth facts. The curve shows that the retrieval recall grows sharply at the beginning and flattens out later on.

Figure 2 shows the estimated web search recall curve for all CRAG questions, with an overall recall of 85% when using all 50 pages. It reflects multiple advantages of the benchmark by design. First, the recall curves are sharp at the beginning and flatten out later, and the recall for the top 5 pages is about 69%. This is comparable to what we observe in practice when developing a RAG system. The non-perfect coverage, especially for Task 1, allows us to test whether the RAG solutions admit "I don't know" when the retrieval results do not contain the necessary information. Second, compared to the recall from web snippets, full web pages increase the recall by about 20%, emphasizing the importance of extracting and understanding HTML contents. Moreover, the estimated web search recall (50 web pages) is 93% for *Web Questions* and 74% for *KG Questions*, indicating significantly lower recall for KG questions than web questions. This aligns with our observations that web search recall for torso and tail entities is typically lower, underlining the crucial role of leveraging KGs in Task 2 and 3.

Mock KGs. We created mock KGs that contain publicly available KG data used to generate the questions, randomly selected entities of the same type, and also "hard negative" entities with similar names (e.g., *"phantom"* for *"phantom of the opera"*).

Mock APIs. We created mock APIs with pre-defined parameters to support structured search in the mock KGs. For example, for queries asking for stock prices, an example mock API is in the form of `get_price_history(ticker)`.

We collected snapshots of the KG and web search data concurrently while posing real-time and fast-changing questions. This approach ensures that we capture the "snapshot" of the information world at the time of question answering. A RAG solution that performs well on the benchmark should also be capable of reasoning over time and generalizing to evolving questions.

In total, the resulting data contains 220K webpages, a KG of 2.6M entities, and 38 mock APIs. See Table 9 in the Appendix for a complete list of the mock APIs.

4 Metrics and Evaluation

In this section, we present the metrics for evaluating the RAG systems and briefly describe the 2024 Meta KDD Cup challenge in Appendix A.2.3.

4.1 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 **incorrect**, 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.

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

We use a scoring method with score 1, 0.5, 0, and -1 for each *perfect, acceptable, missing*, and *incorrect* answer, respectively, where we penalize hallucinated answers and prefer *missing* answers to *incorrect* ones. We then define **truthfulness** as the average score from all examples in the evaluation set for a given RAG system.

4.2 Evaluation

Similar to previous work [37], we employ 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 *incorrect* for each answer. In the latter, we merge *perfect* and *acceptable*, call it **accurate**, and use a three-way scoring system with $1, -1, 0$ for *accurate*, *incorrect*, and *missing* answers.

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, incorrect*, or *missing*. To avoid the *self-preference* problem [25], we use two LLM evaluators: ChatGPT (`gpt-3.5-turbo-0125`) [24] and Llama 3 (`llama-3-70B-instruct`) [2] and report the average *accurate, hallucination, missing* rates, and *truthfulness* scores from the two models for each RAG system. Our offline experiment shows that this two-step method yields an average F1 score of 94.7% for ChatGPT and 98.9% for Llama 3 compared to human-eval. See Appendix A.2.2 for more details.

Test data split. We split the data randomly into *validation* (30%), *public test* (30%), and *private* (40%), and released the validation and public test sets (Appendix A.2.3). Participants of the KDD Cup challenge can use the validation and public test sets to develop and test their models, and the submitted solutions were evaluated on the private test set. Future offline users of CRAG can use the validation set for development, fine-tuning, and validation, and the public test set for testing and result reporting.

5 Benchmarking

In this section, we present the performance of LLMs and RAG systems on CRAG, demonstrating that CRAG has a reasonable level of difficulty and can help draw insights and show directions in developing RAG techniques.

5.1 Straightforward RAG solutions

Experiment setup: We started with running LLM-only solutions on the CRAG public test set with $1,335$ questions, using simple prompts that encourage brief answers and *"I don't know"* answers when the confidence is low (Appendix A.3.1). We employed Llama 2 Chat (`llama-2-7b-chat` and `llama-2-70b-chat`) [34], Llama 3 Instruct (`llama-3-8B-instruct` and `llama-3-70B-instruct`) [2], Mixtral (`Mixtral-8x7B-Instruct-v0.1`) [15], Falcon (`40B`) [3], FLAN-T5 (`FLAN-T5-XXL`) [9], and GPT-4 Turbo (`gpt-4-turbo-2024-04-09`) [1]. The web-only RAG solutions we evaluated (Task 1) used a fixed-length web context window (1K tokens for Falcon and FLAN-T5, 2K for Llama 2 Chat, and 4K for Llama 3 Instruct and GPT-4 Turbo); we concatenated webpage snippets using the original order from the data as the reference text, until filling up the window (similar to [17,23,36]). Our KG-based solutions (Tasks 2, 3) additionally used a fixed-length KG context window (0.5K tokens for Falcon and FLAN-T5, 1K for Llama 2 Chat, and 2K for Llama 3 Instruct and GPT-4 Turbo) to include the results from the mock APIs; we extracted the relevant query entities using `llama-3-8B-instruct` with in-context learning (similar to [28]) detailed in Appendix A.3.1 and concatenated the results returned from all applicable mock APIs (based on the extracted entities), until filling up the window. We provide an extensive comparison of all LLMs in Appendix A.3.2 and focus on the best-performing LLMs (i.e., Llama 3 70B Instruct and GPT-4 Turbo) in this section.

Table 5: Performance of straightforward RAG solutions. All numbers are in percentage. LLM-only solutions has up to 34% accuracy and straightforward RAG solutions has up to 44% accuracy. The subscript in "truthfulness$_a$" denotes the result is reported by auto-eval.

	Model	**Accuracy**	**Hallucination**	**Missing**	**Truthfulness$_a$**
LLM only	Llama 3 70B Instruct	32.3	28.9	38.8	3.4
	GPT-4 Turbo	**33.5**	**13.5**	53.0	**20.0**
Task 1	Llama 3 70B Instruct	35.6	31.1	33.3	4.5
	GPT-4 Turbo	**35.9**	**28.2**	35.9	**7.7**
Task 2	Llama 3 70B Instruct	37.5	29.2	33.3	8.3
	GPT-4 Turbo	**41.3**	**25.1**	33.6	**16.2**
Task 3	Llama 3 70B Instruct	40.6	31.6	27.8	9.1
	GPT-4 Turbo	**43.6**	**30.1**	26.3	**13.4**

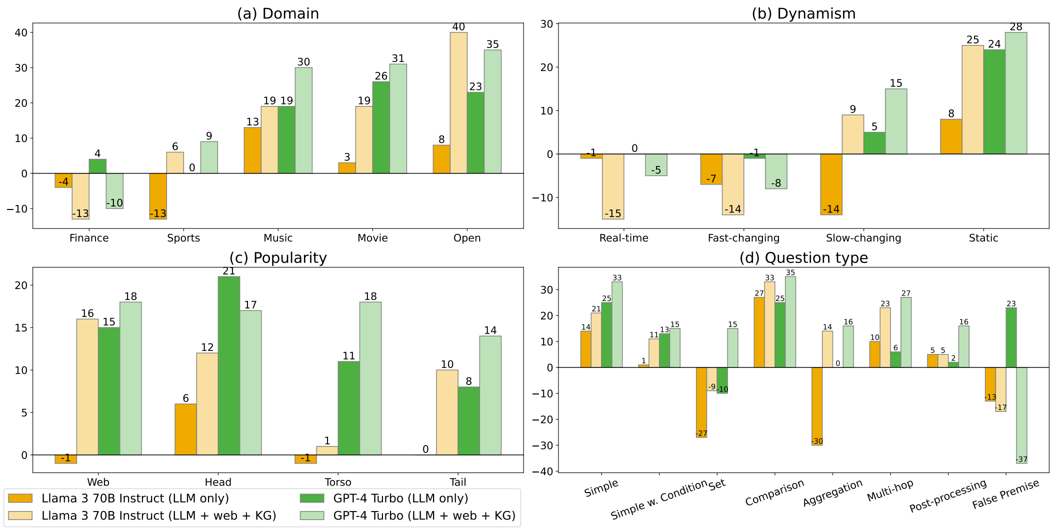

Figure 3: LLM-only and Task 3 solution auto-eval truthfulness (in percentage) across domain, dynamism, popularity, and question type.

Table 5 shows the average evaluation results from the two auto-evaluators (ChatGPT and Llama 3) and illustrates that the CRAG benchmark is ***non-trivial***. First, the best LLM-only solutions (GPT-4 Turbo) obtained an accuracy of only 34%, with truthfulness of 20%, showing a big room for improvement. Second, straightforward RAG solutions obtained up to 44% accuracy, showing that extra information *does* help answer more questions reliably. Interestingly, none of the RAG solutions obtain truthfulness higher than 20%; this is because all RAG solutions introduce more hallucinations generated from irrelevant retrieval results, showing a big challenge in RAG—*How to judiciously use retrieval results without being distracted by retrieval noises?* Third, we found that Task 2 truthfulness scores are higher than Task 1, showing that the KG knowledge helps improve accuracy, with a similar or even lower hallucination rate, because the KG knowledge is typically brief but precise. Unfortunately, the improvement is mediocre, showing a second challenge in RAG—*How to best leverage the power of KG data?* Finally, the truthfulness for Task 3 are also higher than Task 2, because of better search ranking (recall that Task 1 and 2 provide five pages randomly selected from the top-10 search results) and better search recall. In particular, we found that the ground truths of over 30% of questions are available in the web retrieval results but are not included in the prompt due to the context window limitation. This shows *the importance of search ranking* in RAG.

Figure 3 shows the auto-eval results across the domain, dynamism, popularity, and question type dimension. The results reveal a lot of interesting observations and show that the CRAG benchmark allows more ***insightful*** conclusions. First, it shows *which slices of the benchmark are harder*. For example, we found much lower RAG truthfulness on the *Finance* and *Sports* domains, for *real-*

Table 6: Benchmarking CRAG questions with industry SOTA RAG systems. Perfect, acceptable (Acc.), hallucination (Hall.), missing rates (Miss.), and truthfulness$_h$ reported by human-eval (Truth$_h$) are in percentages. The best system achieves truthfulness of 51% and provides perfect answers for up to 63% of questions.

	System	Perfect	Acc.	Hall.	Miss.	Truth$_h$	Latency (ms)
Equal weighted	Copilot Pro	**62.6**	11.7	17.9	7.8	**50.6**	11,596
	Gemini Advanced	60.8	10.1	16.6	12.5	49.3	5,246
	ChatGPT Plus	59.8	**13.3**	25.0	1.9	41.5	6,195
	Meta SG	52.5	9.7	**16.0**	21.8	41.4	**3,431**
	Perplexity.ai	55.8	8.8	25.3	10.1	34.9	4,634
Traffic weighted	Copilot Pro	**70.0**	9.5	14.3	6.1	**60.5**	-
	Gemini Advanced	67.1	10.0	**12.7**	10.2	59.3	-
	ChatGPT Plus	61.8	**11.4**	25.7	1.3	41.8	-
	Meta SG	61.0	7.1	14.1	17.8	50.5	-
	Perplexity.ai	63.7	6.3	20.9	9.1	45.9	-

time and *fast-changing* facts, for *tail* entities, and for complex questions requiring *set answers, post-processing,* and with *false premises*. Second, it shows *where it is harder to leverage retrieval results*. Take the popularity slices as an example, we observed that GPT-4 Turbo's truthfulness dropped from head (21%) to torso (11%) to tail (8%), consistent with past observations [29]; however, the straightforward RAG solution based on GPT-4 Turbo improved QA quality regarding torso (+7%) and tail entities (+6%) but lowered the quality regarding head (-4%). Finally, although our goal is *not* to compare different LLMs, the different dimensions allow us to understand the strengths and weaknesses of each method. For example, although the RAG system based on Llama 3 70B Instruct has a lower overall truthfulness score than the one based on GPT-4 Turbo, it has a similar or slightly higher truthfulness in answering *simple* and *comparison* questions, whereas much lower truthfulness in answering *set* and *post-processing* questions, suggesting investigations on the reasoning capabilities.

5.2 State-of-the-art industry solutions

Next, we evaluated industry state-of-the-art (SOTA) RAG solutions on CRAG public test set. We selected five RAG systems built upon SOTA LLMs and search engines, queried them with CRAG questions, collected the responses, and applied manual grading (details in Appendix A.4).

In addition, we applied traffic weights to the questions to understand the solutions in real-world use cases. The traffic weights reflect the frequency of each question type, as defined in Table 2, in real QA traffic. We gave the same weights to all domains and reported the macro average across domains. This is because we have observed quite different domain-level distributions in different use cases, but have been observing similar distributions at the query-type level.

Table 6 and Figure 4 show the overall performance of the SOTA systems and their performance across different dimensions. The evaluation results confirm our belief that the CRAG benchmark *reveals interesting insights and shows room for improvement for existing RAG solutions.* First, the results from SOTA solutions achieve much better truthfulness (highest 51%) compared to the straightforward solutions. However, the hallucination rate ranges from 16% to 25%, so the answers are still *not* trustworthy. Note that the truthfulness scores between the SOTA solutions and the straightforward solutions are not completely comparable, as they have different accesses to retrieval contents (Appendix A.3 and A.4.1), and the former used auto-eval, while the latter used human-eval; however, the trend is valid. Second, we observed very different latency, ranging from 3.4s to 11.6s, reflecting the different design options in trading off latency and quality; for example, Copilot Pro has the highest truthfulness, but meanwhile highest latency, whereas Meta SG [10] has mid-tier truthfulness but lowest latency. (See Appendix A.4.2 for additional results and how we measured latency.) Third, most difficult slices we see in the straightforward solutions remain to be difficult for SOTA solutions: *real-time* and *fast-changing* queries, and questions regarding *torso* and *tail* entities, showing the improvement needed for handling retrieval noises when the system relies on retrieval results to answer the question; as another example, we see lower truthfulness for queries requiring *aggregation, multi-hop reasoning* or *post-processing*, showing the improvement space for reasoning

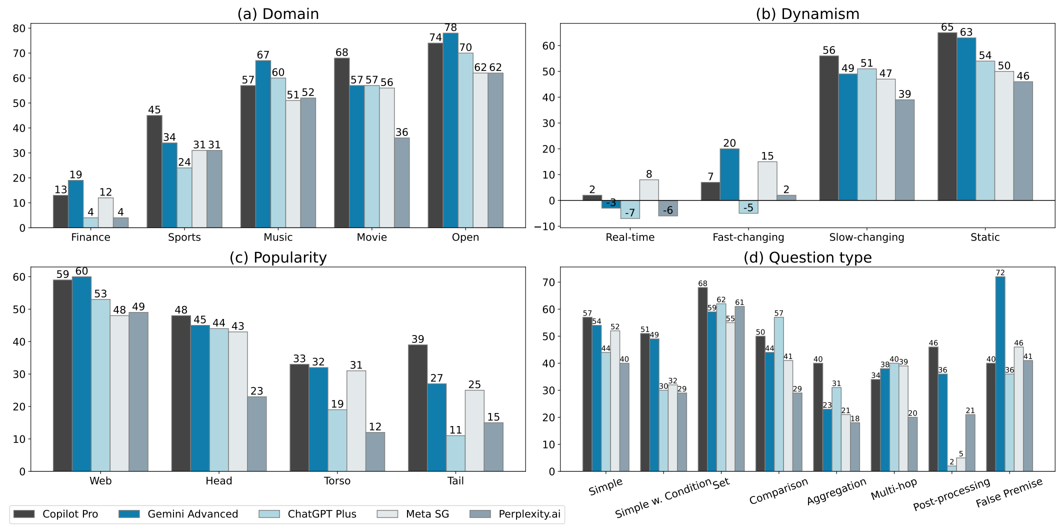

Figure 4: SOTA systems human-eval truthfulness scores (in percentage) across different dimensions.

in question answering. Last, truthfulness on *set* and *false premise* questions improved significantly in the SOTA solutions compared to the straightforward solutions, showing advancement in RAG systems in providing accurate and complete set answers and detecting *false premises*.

6 Conclusion

This paper proposes CRAG, a rich and comprehensive benchmark designed to advance research in retrieval-augmented generation (RAG). With detailed empirical studies, CRAG reviewed gaps in existing RAG solutions and provided valuable insights for future improvement. We plan to continue improving and expanding the benchmark for multi-lingual questions, multi-modal questions, multi-turn conversations, etc., to ensure CRAG stays at the forefront to push RAG research, adapts to emerging challenges, and evolves for new research needs.

Acknowledgements

We would like to thank Jinsong Yu for his advice on the data strategy and help in connecting with the partner teams. We thank Alex Boesenberg for enabling us to create web search results. We thank Sam Wexler for coordinating this project with various partners. We thank Hejia Zhang, Rares Bostan, Eryk Helenowski, Nanshu Wang, Sinong Wang, and Denis Savenkov for the discussion regarding LLM and RAG evaluation. We thank Katie Manlove for coordinating with the budget and annotation needs. We thank our annotation team for creating many great examples: David Vu, Florian Gawlitta, Rani Salomi Pitta, Lauren Gregory Mikus, Tara Welch, Nidhi Modi, Ray De Leon, Jader Ricarte, Joshua Aranzaso. We thank Apoorva Srinivas and Courtnee Parker for coordinating the annotation tasks. We would like to acknowledge the support from Jackie Leader, Mindy Abern, Heather Nolan, Ty Toledano, George Lan, Ben Edgar, Sy Choudhury, Rodrick Shepard, Katie Manlove, Mavis Hu, Jen Vinz, Taha Masood, Parkin Kent, Rebekkah Hogan, Tony Nelli, Jennifer Pak, Jonathan Torres, and Amy Lee.

Lei Chen's work is partially supported by National Key Research and Development Program of China Grant No. 2023YFF0725100, National Science Foundation of China (NSFC) under Grant No. U22B2060, Guangdong-Hong Kong Technology Innovation Joint Funding Scheme Project No. 2024A0505040012, the Hong Kong RGC GRF Project 16213620, RIF Project R6020-19, AOE Project AoE/E-603/18, Theme-based project TRS T41-603/20R, CRF Project C2004-21G, Guangdong Province Science and Technology Plan Project 2023A0505030011, Guangzhou municipality big data intelligence key lab, 2023A03J0012, Hong Kong ITC ITF grants MHX/078/21 and PRP/004/22FX, Zhujiang scholar program 2021JC02X170, Microsoft Research Asia Collaborative Research Grant, HKUST-Webank joint research lab and 2023 HKUST Shenzhen-Hong Kong Collaborative Innovation Institute Green Sustainability Special Fund, from Shui On Xintiandi and the InnoSpace GBA.

References

[1] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

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

[3] E. Almazrouei, H. Alobeidli, A. Alshamsi, A. Cappelli, R. Cojocaru, M. Debbah, É. Goffinet, D. Hesslow, J. Launay, Q. Malartic, et al. The falcon series of open language models. *arXiv preprint arXiv:2311.16867*, 2023.

[4] P. Bajaj, D. Campos, N. Craswell, L. Deng, J. Gao, X. Liu, R. Majumder, A. McNamara, B. Mitra, T. Nguyen, M. Rosenberg, X. Song, A. Stoica, S. Tiwary, and T. Wang. MS MARCO: A human generated machine reading comprehension dataset, 2018.

[5] Brave Software. Brave Search API.

[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] W. Chen, H. Hu, X. Chen, P. Verga, and W. Cohen. MuRAG: Multimodal retrieval-augmented generator for open question answering over images and text. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Dec. 2022.

[8] 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.

[9] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

[10] X. L. Dong. The journey to a knowledgeable assistant with retrieval-augmented generation (rag). In *Companion of the 2024 International Conference on Management of Data*, SIGMOD/PODS '24, page 3, New York, NY, USA, 2024. Association for Computing Machinery.

[11] M. Gao, X. Hu, J. Ruan, X. Pu, and X. Wan. Llm-based nlg evaluation: Current status and challenges. *arXiv preprint arXiv:2402.01383*, 2024.

[12] 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.

[13] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232*, 2023.

[14] 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.

[15] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[16] M. Joshi, E. Choi, D. Weld, and L. Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. Association for Computational Linguistics, July 2017.

[17] N. Kandpal, H. Deng, A. Roberts, E. Wallace, and C. Raffel. Large language models struggle to learn long-tail knowledge. In *International Conference on Machine Learning*, pages 15696–15707. PMLR, 2023.

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

[19] 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.

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

[21] C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[22] 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.

[23] 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. In *ACL*, 2023.

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

[25] A. Panickssery, S. R. Bowman, and S. Feng. Llm evaluators recognize and favor their own generations. *arXiv preprint arXiv:2404.13076*, 2024.

[26] 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.

[27] 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.

[28] T. Schick, J. Dwivedi-Yu, R. Dessi, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv*, 2023.

[29] K. Sun, Y. E. Xu, H. Zha, Y. Liu, and X. L. Dong. Head-to-Tail: How knowledgeable are large language models (llms)? a.k.a. will llms replace knowledge graphs? *arXiv preprint arXiv:2308.10168*, 2024.

[30] 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.

[31] A. Talmor and J. Berant. The web as a knowledge-base for answering complex questions, 2018.

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

[33] Y. Tang and Y. Yang. Multihop-rag: Benchmarking retrieval-augmented generation for multi-hop queries. *arXiv preprint arXiv:2401.15391*, 2024.

[34] 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.

[35] 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.

[36] T. Vu, M. Iyyer, X. Wang, N. Constant, J. Wei, J. Wei, C. Tar, Y.-H. Sung, D. Zhou, Q. Le, and T. Luong. FreshLLMs: Refreshing large language models with search engine augmentation, 2023.

[37] F. Xu, Y. Song, M. Iyyer, and E. Choi. A critical evaluation of evaluations for long-form question answering. In *Association of Computational Linguistics*, 2023.

[38] 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.

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

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Section 3.
   (b) Did you describe the limitations of your work? [Yes] See Appendix A.5.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments (e.g. for benchmarks)...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The dataset, KGs, mock APIs, and starter kits for building baseline systems and performing auto-eval are available at `https://github.com/facebookresearch/CRAG/`.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4.2, Appendix A.2.3, and Appendix A.4.1
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No] Evaluating the performance of the same LLM on different platforms is out of the scope of this work.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 5.1.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] We only use publicly available and shareable data.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

A.1 Dataset

A.1.1 Constructing QA pairs from KGs

We first collected a set of entities based on publicly available data. Then we created question-answer pairs in three steps for *Simple static and dynamic questions*.

Step 1. For each domain, we first selected an entity type and a meaningful relation (e, r) and created a question template. For example, for *(music artist, first album)*, we create a template "*what is the first album of [music artist]?*".

Step 2. We then sampled entities from the KGs to fill in the templates and generate the full question. We adopted the method described in [29] and sampled entities of top, middle, and bottom popularity. We defined popularity based on heuristics for each entity type and created an equal number of questions for each bucket.

Step 3. Last, we took the associated attribute values as the answer to the question to create question-answer pairs.

We created the *Comparison, Aggregation, Set, Post-processing*, and *False-premise questions* in a similar way but 1) made sure the template allows for crisp and deterministic answers and 2) sampled the subject entities that fit the question. We used heuristics to select entity types for these question categories.

Finally, we created multi-hop questions in three steps, similar to those described in [31]. We first sampled an entity e_1 from the KG and selected two relation triplets following a two-hop path: (e_1, r_1, e_2) and (e_1, r_2, e_3). We then created a question template describing the path. For example, for path *(company$_1$, is_parent, company$_2$)* followed by *(company$_1$, ceo, person)*, we created the template *"who is the CEO of the parent company of [company$_2$]?"*. The answer to the new question will be e_3 in the second triplet.

A.1.2 Definition of dynamism categories

Table 7: Definition of dynamism categories.

Dynamism	Definition
Real-time	The answer to the question changes over seconds (e.g., "*What's Costco's stock price today?*").
Fast-changing	The answer to the question changes no more than daily (e.g., "*When is Laker's game tonight?*").
Slow-changing	The answer to the question changes no more than yearly (e.g., "*Who won the Grammy award last year?*").
Static	The answer to the question does not change over time, such as the birth date of a person.

A.1.3 Constructing QA pairs from web contents

Step 1. Ask annotators to write down a list of questions that could possibly be answered by web search based on a general guideline (e.g., "*what is the most popular action movie in 2023?*").

Step 2. Generate the web search results to answer the question.

Step 3. Finally, annotators reviewed the web search results to determine the ground truth answers to the questions: 1) If the search results successfully provided the necessary information, annotators recorded the ground truth answer text and the URL associated with it based on the retrieved content. Note that the answer is determined by the *query_time* at which the web search happened, especially for the *Fast-changing* and *Real-time* questions. 2) Otherwise, annotators conducted further web searches to document the correct answers.

Besides the QA pairs, the annotators will also provide labels for the domain, dynamism, question types, and an *answer URL* (a URL that contains the answer to the question) for *Web Questions*.

A.1.4 Validation for the QA pairs

We conducted two phases of dataset validation with our in-house Linguist team.

Phase 1. Question and meta-label validation. After the initial round of QA pair generation, an audit session was conducted, where expert annotators reviewed the question template, questions, and meta-labels (domain, question type, etc), applying edits as necessary with 2x human review (agreement rate 90%+). All problematic questions (e.g., a wrong false-premise question) were revised, and all conflicting labels were resolved by a third more experienced auditor.

Phase 2. Answer validation. To ensure the answers in the benchmark are correct, we further conducted an auditing process for all the answers. For the web questions, an annotation team reviewed each question and conducted an extensive search to make sure the answer is factually correct and includes comprehensive information (such as for set questions) with 2x human review (agreement rate 90%+). A third more experienced auditor then reviewed all conflicting answers and provided a resolution. For the KG questions, a team of five engineers carefully checked the questions and queried the mock APIs manually to validate the answers. This step resulted in a 5% answer correction.

In both phases, we paid special attention to examples where the straightforward solutions output different answers from the ground truth answers and asked the auditing team to double-check those examples. This step yielded an additional 2% answer updates.

A.1.5 An example of retrieved web search results

Table 8: An example of web search results.

Key	Value
"page name"	"A Short History Of ChatGPT: How We Got To Where We Are Today"
"page url"	"https://www.forbes.com/sites/bernardmarr/2023/05/19/a-short-history-of-chatgpt-how..."
"page snippet"	"OpenAI released an early demo of ChatGPT on <strong>November 30, 2022</strong>..."
"page last modified"	"2024-1-18 15:32:24"
"html page"	"<!DOCTYPE html><html lang="en"><head><link rel="preload" as="font" href="https..."

A.1.6 The mock data and mock APIs

CRAG provides mock APIs to simulate retrieval from web, KG, and real-time APIs in the ***real*** retrieval environment, allowing ***accessible*** facilitating data and fair comparison.

CRAG provides both structured (through mock KGs) and unstructured (web search results) information to test the effectiveness of RAG systems in leveraging a diverse range of available information. First, for each question in the benchmark, CRAG provides up to 50 web search results from a real-world search engine—the Brave Search API [5]. Different from existing benchmarks that use snippets or selected text chunks [4, 6], CRAG provides full HTML pages, containing more information and potentially more noises as in a realistic setting. Second, CRAG provides mock KG search APIs to test structured search for RAG. The mock KGs, though much smaller in size, contain both information necessary to answer a subset of questions in the benchmark and noises that have similar entity or attribute names, again simulating real settings. Our mock KGs contain about 2.6M entities and have a signal-to-noise ratio of less than 1/30.

Table 9: Mock APIs and Descriptions.

APIs	Descriptions
open_search_entity_by_name(query: str) -> dict	Search for entities by name in the Open domain.
open_get_entity(entity: str) -> dict	Retrieve detailed information about an entity in the Open domain.
movie_get_person_info(person_name: str) -> dict	Get information about a person related to movies.
movie_get_movie_info(movie_name: str) -> dict	Get information about a movie.
movie_get_year_info(year: str) -> dict	Get information about movies released in a specific year.
movie_get_movie_info_by_id(movie_id: int) -> dict	Get movie information by its unique ID.
movie_get_person_info_by_id(person_id: int) -> dict	Get person information by their unique ID.
finance_get_company_name(query: str) -> dict	Search for company names in the finance domain.
finance_get_ticker_by_name(query: str) -> dict	Retrieve the ticker symbol for a given company name.
finance_get_price_history(ticker_name: str) -> dict	Get the price history for a given ticker symbol.
finance_get_detailed_price_history(ticker_name: str) -> dict	Get detailed price history for a ticker symbol.
finance_get_dividends_history(ticker_name: str) -> dict	Get dividend history for a ticker symbol.
finance_get_market_capitalization(ticker_name: str) -> dict	Retrieve market capitalization for a ticker symbol.
finance_get_eps(ticker_name: str) -> dict	Get earnings per share (EPS) for a ticker symbol.
finance_get_pe_ratio(ticker_name: str) -> dict	Get the price-to-earnings (PE) ratio for a ticker symbol.
finance_get_info(ticker_name: str) -> dict	Get financial information for a ticker symbol.
music_search_artist_entity_by_name(artist_name: str) -> dict	Search for music artists by name.
music_search_song_entity_by_name(song_name: str) -> dict	Search for songs by name.
music_get_billboard_rank_date(rank: int, date: str = None) -> dict	Get Billboard ranking for a specific rank and date.
music_get_billboard_attributes(date: str, attribute: str, song_name: str) -> dict	Get attributes of a song from Billboard rankings.
music_grammy_get_best_artist_by_year(year: int) -> dict	Get the Grammy Best New Artist for a specific year.
music_grammy_get_award_count_by_artist(artist_name: str) -> dict	Get the total Grammy awards won by an artist.
music_grammy_get_award_count_by_song(song_name: str) -> dict	Get the total Grammy awards won by a song.
music_grammy_get_best_song_by_year(year: int) -> dict	Get the Grammy Song of the Year for a specific year.
music_grammy_get_award_date_by_artist(artist_name: str) -> dict	Get the years an artist won a Grammy award.
music_grammy_get_best_album_by_year(year: int) -> dict	Get the Grammy Album of the Year for a specific year.
music_grammy_get_all_awarded_artists() -> dict	Get all artists awarded the Grammy Best New Artist.
music_get_artist_birth_place(artist_name: str) -> dict	Get the birthplace of an artist.
music_get_artist_birth_date(artist_name: str) -> dict	Get the birth date of an artist.
music_get_members(band_name: str) -> dict	Get the member list of a band.
music_get_lifespan(artist_name: str) -> dict	Get the lifespan of an artist.
music_get_song_author(song_name: str) -> dict	Get the author of a song.
music_get_song_release_country(song_name: str) -> dict	Get the release country of a song.
music_get_song_release_date(song_name: str) -> dict	Get the release date of a song.
music_get_artist_all_works(artist_name: str) -> dict	Get all works by an artist.
sports_soccer_get_games_on_date(team_name: str, date: str) -> dict	Get soccer games on a specific date.
sports_nba_get_games_on_date(team_name: str, date: str) -> dict	Get NBA games on a specific date.
sports_nba_get_play_by_play_data_by_game_ids(game_ids: List[str]) -> dict	Get NBA play by play data for a set of game ids.

A.2 Evaluation

A.2.1 Human evaluation

We run human evaluation to score each answer with respect to the metrics defined in Section 4.1. We score each *perfect, acceptable, missing*, or *incorrect* answer with a score s_p, s_a, s_m, and s_{in}, respectively and define **truthfulness**$_h$ as the score of the answer by setting $s_p = 1$, $s_a = 0.5$, $s_m = 0$, and $s_{in} = -1$. We then compute the average truthfulness for all examples in the evaluation set as the truthfulness score for the RAG solution.

Human-eval instructions. The human-eval instructions are as follows.
Given a query, a query day and time at which the query was made, the chatbot's response, grade the *accuracy* for the response according to the criteria below:
Using an external search engine, please evaluate the factual accuracy of the response based on the grading rubric below. An accurate answer should be factually correct and provide useful information to answer the user's question.

- Accuracy = 0 (Missing). It covers the following situations. There is no response. There is a failure to provide a response to the request (e.g. "I'm sorry ......, I can't do ...") due to inability. E.g., Query: "Latest news about the Nobel prize today." Response: "I can't find specific information regarding the Nobel prize ..." The response fails to answer and asks follow-up questions.
- Accuracy = 1 (Incorrect). It covers the following situations. The answer is unintelligible. The answer is poorly formed. The answer contains a major hallucination (e.g. wrong date,

Table 10: Accuracy and F1 for the ChatGPT and Llama 3 auto-evaluation models.

	Accuracy		**Precision**		**Recall**		**F1 score**	
	ChatGPT	Llama 3	ChatGPT	Llama 3	ChatGPT	Llama 3	ChatGPT	Llama 3
Accurate	94.1	**98.6**	**98.8**	98.5	92.2	**99.3**	92.0	**98.9**
Incorrect	94.1	**98.6**	86.8	**98.7**	**97.8**	97.2	92.0	**97.9**
Missing	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Average	96.1	**99.1**	95.2	**99.1**	96.7	**98.8**	94.7	**98.9**

wrong numbers, or other significant factual errors). The answer is irrelevant to the user's request. Answers in a category, location, or time window that is significantly different from the user's request, if any. There is a significant structural/formatting error. The response is otherwise a total structural/functional failure and does not contain sufficient well-formed content that can be used to determine accuracy

- Accuracy = 2 (Acceptable). It covers the following situations. The answer is acceptably correct and relevant to the user's request, but may miss some information, i.e. accurate but not complete, or mostly accurate with minor issues. The answer may contain some minor hallucination that doesn't significantly alter the overall meaning. "Minor hallucination" means the answer addressed the user's question but might be off on some additional details. The rule of thumb is to see if the answer serves the purpose of the user's question, and whether the hallucination could mislead the users on what they were asking.
- Accuracy = 3 (Accurate). The answer is factually correct, contains all the relevant information requested, responds appropriately to the query, and does not contain any hallucination.

We requested two human graders to grade each question (agreement rate 94%), and when there is a conflict, a third more experienced grader will resolve it.

A.2.2 Automatic evaluation

In auto-eval, we merge *perfect* and *acceptable* as *accurate* and consider only three scores: 1 for *accurate*, 0 for *missing*, and -1 for *incorrect*. The **truthfulness**$_a$ is calculated by the average truthfulness for all the examples in the evaluation set, and is effectively

$$\text{accuracy} - \text{hallucination},$$

where **accuracy**, **hallucination**, and **missing** are the percentage of *accurate*, *incorrect*, and *missing* answers in the test set. These score choices penalize *incorrect* answers, award *correct*, and assign a value of 0 to *missing* answers.

Auto-evaluators. We computed the accuracy and F1 for the two auto-evaluators for the *accurate, incorrect*, and *missing* examples in the public test set, respectively. Here, we considered human-eval labels as the ground truth. Table 10 shows both models attain reasonable accuracy and F1 scores as an evaluator compared to human evaluation.

Auto-eval prompt. The prompt we used in the auto-eval is similar to the following. We did not release the exact prompt used in the challenge to avoid prompt attack.
PROMPT = """# Task: You are given a Question, a model Prediction, and a list of Ground Truth answers, judge whether the model Prediction matches any answer from the list of Ground Truth answers. Follow the instructions step by step to make a judgement.
1. If the model prediction matches any provided answers from the Ground Truth Answer list, "Accuracy" should be "True"; otherwise, "Accuracy" should be "False".
2. If the model prediction says that it couldn't answer the question or it doesn't have enough information, "Accuracy" should always be "False".
3. If the Ground Truth is "invalid question", "Accuracy" is "True" only if the model prediction is exactly "invalid question".
# Output:
Respond with only a single JSON string with an "Accuracy" field which is "True" or "False".
# Examples:
Question: how many seconds is 3 minutes 15 seconds?
Ground truth: ["195 seconds"]

Prediction: 3 minutes 15 seconds is 195 seconds.
Accuracy: True
Question: Who authored The Taming of the Shrew (published in 2002)?
Ground truth: ["William Shakespeare", "Roma Gill"]
Prediction: The author to The Taming of the Shrew is Roma Shakespeare.
Accuracy: False
Question: Who played Sheldon in Big Bang Theory?
Ground truth: ["Jim Parsons", "Iain Armitage"]
Prediction: I am sorry I don't know.
Accuracy: False
"""

A.2.3 KDD Cup 2024 Meta CRAG challenge

The KDD Cup 2024 Meta CRAG challenge has two stages. Stage 1 is designed for the participants to develop their RAG solutions by submitting their systems against the leaderboard, whereas Stage 2 determines the final winners. We split our benchmark data into three sets with similar distributions: *validation, public test*, and *private test* at 30%, 30%, and 40%, respectively. We shared the validation and public test sets, in total, 2,706 examples in Stage 1, and held out the private test set to select the final winners in Stage 2. We used auto-eval for Stage 1, and selected top teams with auto-eval in Stage 2 to conduct manual evaluation.

A.3 Evaluating straightforward RAG solutions

We send each CRAG question in a prompt shown below. This prompt is designed to include the original *Query Time* for the question and ask the LLM to answer the question **based on the query time and the retrieved result**. Note that the retrieved result was also from the same *query time* and was provided in CRAG. Moreover, this prompt encourages brief answers and *"I don't know"* answers when confidence is low.

A.3.1 Prompts used in straightforward RAG solutions

Vanilla LLM. PROMPT = """ You are given a Question and the time when it was asked in the Pacific Time Zone (PT), referred to as "Query Time". The query time is formatted as "mm/dd/yyyy, hh:mm:ss PT". Your task is to answer the question in as few words as possible.
Please follow these guidelines when formulating your answer:
1. If the question contains a false premise or assumption, answer "invalid question".
2. If you are uncertain or don't know the answer, respond with "I don't know".
### Question
{query}
### Query Time
{query_time}
### Answer
"""

RAG with web search results (Task 1). PROMPT = """ You are given a Question, References and the time when it was asked in the Pacific Time Zone (PT), referred to as "Query Time". The query time is formatted as "mm/dd/yyyy, hh:mm:ss PT". The references may or may not help answer the question. Your task is to answer the question in as few words as possible.
Please follow these guidelines when formulating your answer:
1. If the question contains a false premise or assumption, answer "invalid question".
2. If you are uncertain or don't know the answer, respond with "I don't know".
### Question
{query}
### Query Time
{query_time}
### References
{references}

### Answer
"""

RAG with KG and web search results (Tasks 2 and 3). PROMPT = """ You are given a Question, References and the time when it was asked in the Pacific Time Zone (PT), referred to as "Query Time". The query time is formatted as "mm/dd/yyyy, hh:mm:ss PT". The references may or may not help answer the question. Your task is to answer the question in as few words as possible.
Please follow these guidelines when formulating your answer:
1. If the question contains a false premise or assumption, answer “invalid question”.
2. If you are uncertain or don’t know the answer, respond with “I don’t know”.
### Question
{query}
### Query Time
{query_time}
### References
# web
{web_results}
# knowledge graph
{kg_response}
### Answer
"""

Query entity extraction. PROMPT = """ You are an agent that only outputs JSON. You are given a Query and Query Time. Do the following:

1) Determine the domain the query is about. The domain should be one of the following: "finance", "sports", "music", "movie", "encyclopedia". If none of the domains apply, use "other". Use "domain" as the key in the result json.

2) Extract structured information from the query. Include different keys into the result json depending on the domains, and put them DIRECTLY in the result json. Here are the rules:

For ‘encyclopedia’ and ‘other’ queries, these are possible keys:
- ‘main_entity’: extract the main entity of the query.

For ‘finance’ queries, these are possible keys:
- ‘market_identifier’: stock identifiers including individual company names, stock symbols.
- ‘metric’: financial metrics that the query is asking about. This must be one of the following: ‘price’, ‘dividend’, ‘P/E ratio’, ‘EPS’, ‘marketCap’, and ‘other’.
- ‘datetime’: time frame that the query asks about. When datetime is not explicitly mentioned, use ‘Query Time’ as default.

For ‘movie’ queries, these are possible keys:
- ‘movie_name’: name of the movie
- ‘movie_aspect’: if the query is about a movie, which movie aspect the query asks. This must be one of the following: ‘budget’, ‘genres’, ‘original_language’, ‘original_title’, ‘release_date’, ‘revenue’, ‘title’, ‘cast’, ‘crew’, ‘rating’, ‘length’.
- ‘person’: person name related to moves
- ‘person_aspect’: if the query is about a person, which person aspect the query asks. This must be one of the following: ‘acted_movies’, ‘directed_movies’, ‘oscar_awards’, ‘birthday’.
- ‘year’: if the query is about movies released in a specific year, extract the year

For ‘music’ queries, these are possible keys:
- ‘artist_name’: name of the artist
- ‘artist_aspect’: if the query is about an artist, extract the aspect of the artist. This must be one of the following: ‘member’, ‘birth place’, ‘birth date’, ‘lifespan’, ‘artist work’, ‘grammy award count’, ‘grammy award date’.
- ‘song_name’: name of the song
- ‘song_aspect’: if the query is about a song, extract the aspect of the song. This must be one of the

following: ‘author’, ‘grammy award count’, ‘release country’, ‘release date’.

For ‘sports’ queries, these are possible keys:
- ‘sport_type’: one of ‘basketball‘, ‘soccer‘, ‘other‘
- ‘tournament’: NBA, World Cup, Olympic.
- ‘team’: teams that users are interested in.
- ‘datetime’: time frame that the user is interested in. When datetime is not explicitly mentioned, use ‘Query Time’ as default.

Return the results in a FLAT json.

NEVER include ANY EXPLANATION or NOTE in the output, ONLY OUTPUT JSON!!!
"""

A.3.2 Performance of straightforward RAG solutions

Table 11 summarizes the results of straightforward RAG solutions.

Table 11: Performance of straightforward RAG solutions on CRAG.

	Model	**Accuracy (%)**	**Hallucination (%)**	**Missing (%)**	**Truthfulness (%)**
LLM only	Llama 2 7B Chat	14.8	78.4	**6.7**	-63.6
	Llama 2 70B Chat	22.3	28.7	49.0	-6.4
	Llama 3 8B Instruct	23.7	33.8	42.6	-10.1
	Llama 3 70B Instruct	32.3	28.9	38.8	3.4
	Falcon 40B	10.8	41.9	47.3	-31.1
	FLAN-T5-XXL 11B	9.4	8.7	81.9	0.7
	Mixtral-8x7B-Instruct-v0.1	20.8	27.0	52.1	-6.2
	GPT-4 Turbo	**33.5**	**13.5**	53.0	**20.0**
Task 1	Llama 2 7B Chat	16.4	83.1	**0.5**	-66.7
	Llama 2 70B Chat	29.3	61.0	9.7	-31.7
	Llama 3 8B Instruct	28.5	45.6	25.9	-17.1
	Llama 3 70B Instruct	35.6	31.1	33.3	4.5
	Falcon 40B	21.9	55.5	22.5	-33.6
	FLAN-T5-XXL 11B	27.5	36.5	36.0	-9.0
	Mixtral-8x7B-Instruct-v0.1	33.6	44.4	22.0	-10.8
	GPT-4 Turbo	**35.9**	**28.2**	35.9	**7.7**
Task 2	Llama 2 7B Chat	16.4	83.1	**0.5**	-66.7
	Llama 2 70B Chat	29.1	61.1	9.7	-32.0
	Llama 3 8B Instruct	28.6	45.5	25.9	-16.9
	Llama 3 70B Instruct	37.5	29.2	33.3	8.3
	Falcon 40B	21.9	55.4	22.7	-33.5
	FLAN-T5-XXL 11B	27.4	36.6	36.0	-9.2
	Mixtral-8x7B-Instruct-v0.1	33.4	44.6	22.0	-11.2
	GPT-4 Turbo	**41.3**	**25.1**	33.6	**16.2**
Task 3	Llama 2 7B Chat	16.0	83.6	**0.4**	-67.6
	Llama 2 70B Chat	31.9	65.7	2.4	-33.7
	Llama 3 8B Instruct	32.1	56.3	11.6	-24.1
	Llama 3 70B Instruct	40.6	31.6	27.8	9.1
	Falcon 40B	22.0	56.6	21.3	-34.6
	FLAN-T5-XXL 11B	27.8	37.1	35.1	-9.3
	Mixtral-8x7B-Instruct-v0.1	33.5	44.1	22.4	-10.6
	GPT-4 Turbo	**43.6**	**30.1**	26.3	**13.4**

A.4 Evaluating state-of-the-art industry solutions

A.4.1 Quality

We send the CRAG public test set question as input to each of the SOTA RAG systems and collect the responses for human grading. Note that the original *Query Time* and the provided retrieval results in CRAG are **not** used in this setting. We simply test the questions and ask human graders to grade the responses based on when the query was made to the SOTA system. We called Copilot Pro, Gemini Advanced, and ChatGPT Plus through their web interfaces and Perplexity.ai through its API. Meta SG, designed as a smart glasses (SG) assistant, includes default on-device components such as Automatic Speech Recognition (ASR) and Text-to-Speech (TTS), which are not typically enabled by default in other systems. To ensure a fair comparison, we excluded these on-device components, ensuring that answer quality was not affected. We called each system on the following dates in Pacific Time: 05/12/2024∼05/16/2024 (Copilot Pro), 05/20/2024∼05/28/2024 (Gemini Advanced), 05/27/2024∼06/02/2024 (ChatGPT Plus), 05/15/2024∼05/16/2024 (Perplexity.ai), and 07/02/2024 (Meta SG). We set the conversation style to "Precise" when calling Copilot Pro and the temperature to 0 when calling Perplexity.ai. We select `GPT-4o` and `llama-3-sonar-large-32k-online` as the base LLM when calling ChatGPT Plus and Perplexity.ai, respectively.

A.4.2 Latency

We quantified the latency by calculating the time difference between the timestamp of the query submission to the system and the timestamp when the complete response was received.

The latency of Perplexity.ai measured via API call is 2,455ms. Since latency measured by API call and web interface interactions are not directly comparable, we further called Perplexity.ai through its web interface and reported the latency under this setting in Table 6. Note that this latency may not correspond to the accuracy numbers from the API calls. For Meta SG, we estimated a latency comparable to other web interface interactions by excluding on-device components such as ASR and TTS from the overall end-to-end latency measurement.

A.5 Limitations

The three tasks in our benchmark do not directly evaluate the construction of a first-stage retrieval candidate pool, a demanding retrieval task in its own right. This design decision ensures that the competition remains both challenging and achievable within the KDD Cup's required three-month timeframe. Despite the limitation, users of our dataset have the option to use the union of all 220K web pages as a corpus to build a retriever. While this corpus does not match the entire web, it allows for fair comparisons and manageable costs.