KERAG: Knowledge-Enhanced Retrieval-Augmented Generation for Advanced Question Answering

Yushi Sun[1], **Kai Sun**[2], **Yifan Ethan Xu**[2], **Xiao Yang**[2],
Xin Luna Dong[2], **Nan Tang**[3*], **Lei Chen**[3,1]
[1]HKUST, [2]Meta Reality Labs, [3]HKUST(GZ)
[1]{ysunbp, leichen}@cse.ust.hk,
[2]{sunkaicn, ethanxu, xiaoyangfb, lunadong}@meta.com,
[3]{nantang, leichen}@hkust-gz.edu.cn

Abstract

Retrieval-Augmented Generation (RAG) mitigates hallucination in Large Language Models (LLMs) by incorporating external data, with Knowledge Graphs (KGs) offering crucial information for question answering. Traditional Knowledge Graph Question Answering (KGQA) methods rely on semantic parsing, which typically retrieves knowledge strictly necessary for answer generation, thus often suffer from low coverage due to rigid schema requirements and semantic ambiguity. We present KERAG[1], a novel KG-based RAG pipeline that enhances QA coverage by retrieving a broader subgraph likely to contain relevant information. Our retrieval-filtering-summarization approach, combined with fine-tuned LLMs for Chain-of-Thought reasoning on knowledge sub-graphs, reduces noises and improves QA for both simple and complex questions. Experiments demonstrate that KERAG surpasses state-of-the-art solutions by about 7% in quality and exceeds GPT-4o (Tool) by 10-21%.

1 Introduction

Retrieval-Augmented Generation (RAG) has been extensively explored recently in both academia and industry, as a means to mitigate the hallucination issues of Large Language Models (LLMs) (Sun et al., 2024d; Yang et al., 2024; Fan et al., 2024; Ni et al., 2025; Sun et al., 2025). RAG enhances LLM outputs by incorporating relevant data from external sources, thereby grounding the generated content with factual knowledge. Knowledge graphs serve as a crucial data source, providing efficient access to rich and precise information for question answering (Dong, 2024; Hogan et al., 2025).

Knowledge Graph Question Answering (KGQA) is a well-established field that has been extensively studied long before the advent of LLMs. Most solutions utilize the technique of *semantic parsing*, where a natural language question is converted into a structured query, in the form of either a SPARQL query (Yih et al., 2016; Das et al., 2021; Gu and Su, 2022; Shu et al., 2022; Hu et al., 2022; Xie et al., 2022; Xu et al., 2023; Zhang et al., 2023), or a retrieval path that first identifies the queried

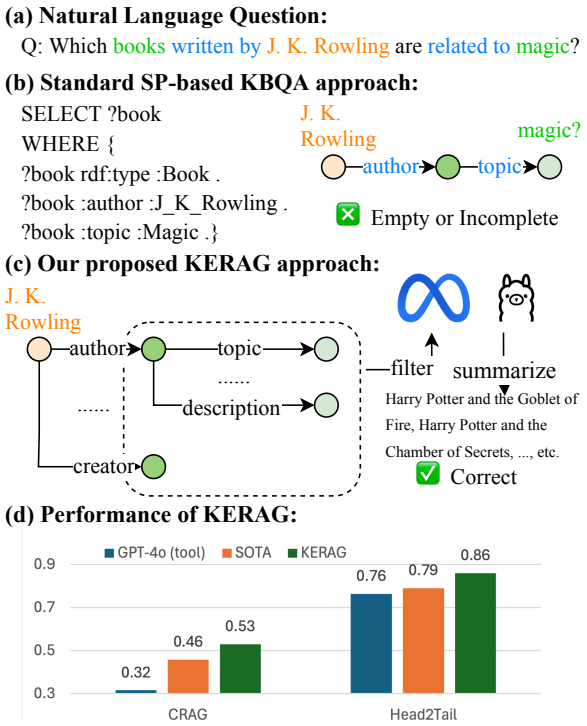

Figure 1: (a) An exemplar query. (b) On the left, we show the SPARQL query generated by the triple-level SP-based approach. On the right, we display the linked KG content. This design leads to empty or incomplete answers since: 1) The book entities do not necessarily have an attribute named "topic"; 2) "magic" may not be explicitly presented in the topic attributes, but may be embedded in other attributes such as description. (c) The rationale behind our KERAG approach: entity-level SP with LLM-based post-processing of the retrieved KG content. (d) Truthfulness of KERAG against state-of-the-arts (truthfulness=accuracy-hallucination).

*Corresponding author.

[1]All the code and data were done by HKUST: https://github.com/ysunbp/KERAG.

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 6194–6216
November 4-9, 2025

entity and then follows a particular predicate to a neighboring entity or attribute value as the answer (Yih et al., 2015; Luo et al., 2018; Lan et al., 2019; Yu et al., 2022). When successful, these solutions produce precise and concise answers. However, they require rigorous schemas (ontologies) and have low tolerance to parsing errors (see Figure 1 for an example). As a result, whereas KGQA generally provides highly accurate answers, it often suffers from low coverage, missing answers for many questions. With LLMs emerging, recent works (Sun et al., 2024a; Ma et al., 2025) leverage the reasoning ability of LLMs to generate *multiple* retrieval paths; however, the small number of paths (e.g., up to 3 in (Ma et al., 2025)) does not fully address the retrieval recall issue.

LLMs, with their strong summarization and reasoning capabilities, offer a new perspective to pursue KGQA and allow us to push *reasoning* from retrieval-path generation to the summarization step. At retrieval time, instead of following up to a few retrieval *paths*, we gather from the neighborhood of the topic entity *all* information potentially relevant to the question. At answer generation time, we leverage LLM's reasoning capabilities to identify relevant information and generate accurate answers. By focusing on the whole neighborhood rather than a few KG paths, the complexity of generating logical forms for KG querying is significantly reduced and the retrieval recall can significantly increase.

Although the idea appears straightforward, there are three prominent challenges. First, entities, particularly head entities, can be associated with a substantial amount of knowledge (2 million triples per head entity (Dong et al., 2014)). All knowledge may not fit within the context window, and even if it does, the noises can lead to confusion during summarization. Second, some questions require information from multi-hop neighbors, complicating the determination of retrieval boundaries, and further increasing the volume of retrieved information. Lastly, complex questions that necessitate aggregation and reasoning remain challenging.

We proposed KERAG, a novel Knowledge-Enhanced RAG pipeline that employs a *retrieval-filter-summarization* paradigm. The *retrieval* part gathers information about the topic entity mentioned in the question; the *filtering* part eliminates information irrelevant to the question; and the *summarization* part reasons over the remaining information to generate the answer. Rather than retrieving the entities and triples strictly necessary for generating an answer, our method retrieves a broader subgraph that is likely to contain relevant information for answer generation.

There are two key innovations that ensure high-quality answers. First, in our planning phase we interleave multi-hop neighborhood retrieval with filtering according to the KG schema, allowing us to filter at the predicate level rather than overwhelming the summarizer with irrelevant triples, and meanwhile ensuring that all information necessary for question answering is retrieved from the neighborhood. Second, we fine-tune the LLM for Chain-of-Thought (CoT) reasoning over the neighborhood sub-graph, enabling it to generate accurate answers for both simple questions and complex questions. An additional advantage of our method is its applicability not only to graph databases that support SPARQL, but also to API-based knowledge-providing systems, which are commonly used in production (Yang et al., 2024).

To summarize, we make three contributions.

- We introduce a novel KG-based RAG pipeline, KERAG, which retrieves information at the entity level, rather than at the triple level as in traditional semantic parsing solutions. This approach significantly enhances QA coverage.
- We design a retrieval-filter-summarization paradigm, complemented by fine-tuning LLMs for Chain-of-Thought-based summarization. This solution reduces the noises in summarization and improves answer generation for both simple and complex questions.
- We conducted extensive experiments on two KG benchmarks (API-based and SPARQL-based), showing that KERAG outperforms state-of-the-art solutions by 7% in quality, and outperforms GPT-4o (Tool) by 10-21%.

2 Related Work

One major line of research of KGQA is semantic-parsing-based (SP-based) KGQA approaches, which primarily transform natural language queries into logical forms and initiate structured queries to access the knowledge graph content. The SP-based KGQA approaches can be further divided into multi-step and seq2seq approaches. The multi-step approaches generally formulate the semantic parsing problem as a multi-step search problem: identifying the core entity, generating and expanding the query graph based on the relationships between the attributes of the entities and the

predicates of the queries (Lan et al., 2019; Lan and Jiang, 2020; Oguz et al., 2022). The seq2seq approaches primarily consider the semantic parsing problem in a seq2seq manner with the help of fine-tuned language models: directly generating the complete semantic expression based on the query and retrieving the relevant knowledge graph content (Das et al., 2021; Shu et al., 2022; Hu et al., 2022; Xie et al., 2022; Yu et al., 2022; Zhang et al., 2023; Gu et al., 2023; Xu et al., 2023). Specifically, Pangu (Gu et al., 2023) presents an innovative framework for grounded language understanding that integrates a symbolic agent with a neural language model. This system enables the stepwise formulation of valid plans while employing the language model to evaluate the viability of these logical plans in an online manner. WikiSP is a novel few-shot seq2seq semantic parser developed by (Xu et al., 2023), which fine-tunes a Llama model on modified SPARQL queries.

Another line of research in KGQA is the information-retrieval-based KGQA approaches, where a subgraph is generated based on the entities and relations in the query, graph matching and retrieval techniques are applied to retrieve relevant graphs from the KG to answer the query (Saxena et al., 2020; He et al., 2021; Sen et al., 2021; Shi et al., 2021; Mavromatis and Karypis, 2022; Zhang et al., 2022). For instance, (Zhang et al., 2022) introduces a novel approach with a trainable subgraph retriever (SR) designed to operate separately from the reasoning process, identifying relevant subgraphs and prioritizing smaller and more relevant subgraphs. The subgraphs identified by our approach are more relaxed compared to (Zhang et al., 2022), which provide a broader context for the summarizer to generate accurate answers.

Thanks to the strong reasoning ability of language models, an increasing number of works start to seek or analyze the assistance of language models (Sun et al., 2023, 2024d; Li et al., 2022, 2024a, 2025; Wang et al., 2024; Chen et al., 2025). The KGQA community also focuses on LLM-based approaches recently, which utilize large language models (LLMs) to perform KGQA (Sun et al., 2024a; Ma et al., 2025; Jiang et al., 2023). As for the LLM-based KGQA approaches, several state-of-the-art LLMs such as Llama-3.1s (Dubey et al., 2024) support tool calling, which provides the opportunity to directly prompt the LLMs to generate API calls or SPARQL queries. However, retrieving directly with the tool call plan given by the LLM would propagate planning errors into the retrieval process, resulting in insufficient KG content being retrieved. We provide a comparison of performance between our KERAG pipeline and the Llama-3.1-70B-Instruct (tool) model (which can freely plan and initiate multiple rounds of API calls) in Tables 1 and 2.

Beyond naive LLM tool calling, ToG (Sun et al., 2024a) introduces beam search on the knowledge graph to iteratively explore reasoning paths for LLMs. ToG-2 (Ma et al., 2025) links to external Wikipedia pages to get additional context, the retrieval and reasoning are conducted between KG and external pages. StructGPT (Jiang et al., 2023) designs an interface for data retrieval and a unique process that involves invoking, linearizing, and generating outputs. Our method is fundamentally different: these approaches retrieve information from KGs based on retrieval paths, whereas KERAG retrieves all potentially relevant information in a neighborhood, thus improving on the retrieval recall (~11% over ToG, Section 5.4). As such, KERAG achieves a higher accuracy on all benchmarks, improving over ToG and ToG-2 by up to 20% and 4% respectively (Table 3), and outperforming StructGPT by 7% in truthfulness (Table 2). Additionally, KERAG has the flexibility to apply both on KGs and APIs, allowing experiments on datasets like Head2Tail (Sun et al., 2024b) and CRAG (Yang et al., 2024). We make detailed comparisons in Appendix B.

3 Problem Statement

Knowledge Graph: A *Knowledge Graph (KG)* K can be defined as $K = (E, R, D)$, where E is a set of *entities*, R is a set of *relations* and optionally D is a set of *domains* covered by K. Each entity $e_i \in E$ can have properties. Each relation $r_j \in R$ can be represented as a binary relation $R : E \times E \to \mathbb{B}$. A KG is normally accompanied with an *ontology*, describing the entity types and relationships between different types of entities.

KG-based RAG: Given a natural language question Q with a KG K, *KG-based RAG* aims to answer the question Q leveraging knowledge in K.

4 Methodology

4.1 Solution overview

We start with an overview of our solution. Given a question Q and a knowledge graph K, KERAG retrieves knowledge from K and uses it to generate

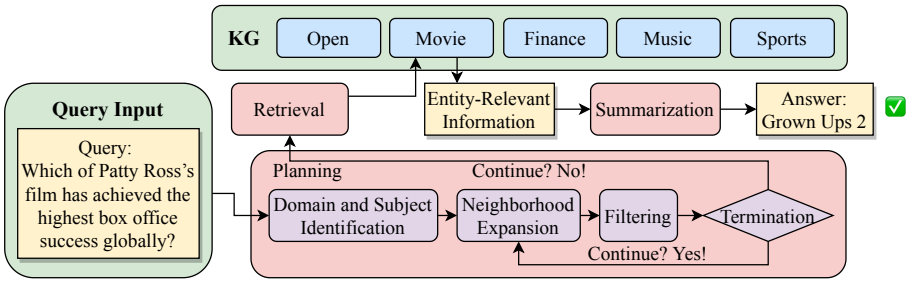

Figure 2: The overall pipeline of our KERAG approach.

the answer for Q. As illustrated in Figure 2, the pipeline contains three major steps.

1. **Planning:** The *Planning* step formulates a retrieval plan by selecting a topic entity E as a starting point and determining the scope of its neighborhood for retrieval. The scope includes the number of hops h for neighborhood exploration and relations $\bar{R}$ to filter.

2. **Retrieval:** The *Retrieval* step translates the retrieval plan to KG queries or API calls to retrieve relevant knowledge.

3. **Summarization:** Finally, the *Summarization* step reasons over the retrieved knowledge and produces the final answer.

The Retrieval step is fairly easy and varies based on the underlying KG system. In the rest of the section, we describe the planning and summarization steps in detail, with a focus on how they address the three main challenges: managing *knowledge overloading*, determining *multi-hop retrieval boundaries*, and handling *complex questions*.

4.2 Planning

The planning step takes a question Q as input, and decides a retrieval plan as a quadruplet $(D, E, \bar{R}, h)$, where D denotes the domain, E denotes the topic entity as the starting point, $\bar{R}$ denotes the relations to filter, and h denotes the number of hops to explore in the neighborhood.

We distinguish the retrieval plan from a retrieval query (e.g., SPARQL) or a retrieval path generated by traditional semantic parsing. Normal retrieval queries return a particular node (or a set of nodes) strictly necessary to answer the question. Our retrieval plan, instead, returns a neighborhood scope that might be needed for answering a question; only relations that are absolutely irrelevant to a question are excluded (e.g., *Oscar awards* for a question regarding movie *box office*).

Our planning proceeds in four steps.

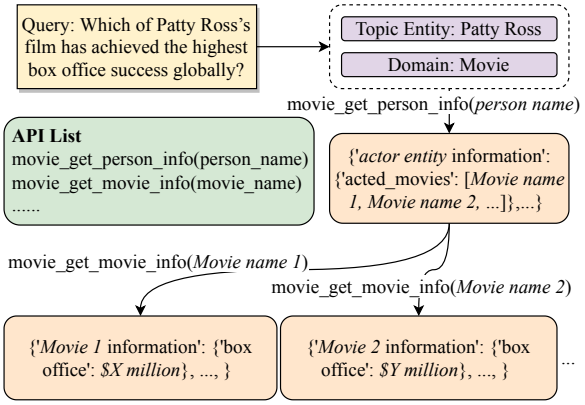

Figure 3: An exemplar query illustrating the planning.

- *Step 1:* We start with prompt to decide the topic entity E and the domain D from the input question Q. As shown in Figure 3, *"Which of Patty Ross's film has achieved the highest box office success globally?"*. The topic entity is *Patty Ross* and the queried domain is *Movie*.
- *Step 2:* Knowing the type of the topic entity, we leverage the schema to explore one hop in the neighborhood, identifying the relations and the types of the neighbors. From our example entity *Patty Ross*, the neighborhood includes relations such as acted movies, birthday, etc., with neighbors of types *entity (movie)*, *attribute (date)*, etc.
- *Step 3:* We feed the neighborhood information into an LLM to decide which relations are irrelevant, and whether enough information has been collected to answer question Q. For example, facts regarding *birth_date* and *birth_place* are irrelevant to answer the question; just knowing the movie neighbors without further information about these movies is not enough to answer the question.
- *Step 4:* We terminate if LLM decides that there is enough information to answer the question, updating h (#hops) accordingly; oth-

erwise, we go to Step 2 to explore the next hop of the neighborhood, unless we have exhausted relevant relations or reached a predetermined maximum number of hops. In our example, we will continue to retrieve information about movies, including their box office, necessary to answer the question.

Next, we describe each step in more detail.

Domain and Subject Identification: We leverage LLMs for this step. We apply few-shot learning and include necessary schema information to guide LLM, including the explanation and examples of domains. We use the following template (Planning Prompts Skeleton in Appendix H, details in the Appendix I.1).

Neighborhood Expansion: This step simply explores the schema of the knowledge graph, to understand relations and entity types in the next hop of the neighborhood to prepare the next step.

Filtering: This step decides 1) which relations in the obtained neighborhood content are absolutely irrelevant to answer the question Q thus can be filtered, and 2) whether we have obtained enough information for question answering, addressing both the *knowledge overloading* challenge and *multi-hop boundary* challenge. We prepare both LLM-based filtering and similarity-based filtering (Karpukhin et al., 2020) to filter according to the semantics of the question and the ontology. We apply the following LLM-based filtering template (Filtering Prompts Skeleton in Appendix H, details in the Appendix I.2):

Termination: This step takes the LLM output to decide whether to continue planning or terminate. Once the LLM decides all necessary information is collected for question answering, we can proceed to the retrieval step.

Our planning step has the following advantages. First, we enlarge the scope of retrieval compared to traditional semantic parsing, thus allowing us to lift the retrieval recall. Second, we allow multi-hop retrieval, such that we are able to answer complex questions regarding multi-hop reasoning. Third, we apply filtering to remove irrelevant content to avoid distracting the final summarization step. Finally, hopping and filtering decisions are made according to the ontology, significantly reduced the cost for full data retrieval (in case ontology is absent, we can easily adjust the algorithm to retrieve and reason about the data in each hop of neighborhood). On the CRAG dataset, multi-hop retrieval improved truthfulness by 5.5% and filtering improved by 3.9% (Table 4).

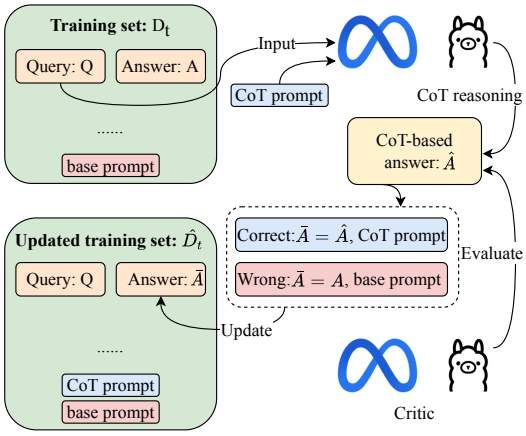

Figure 4: Generation of the fine-tuning data.

4.3 CoT Summarization

The final *Summarization* step takes the question Q and the retrieval results, applies CoT reasoning to generate the answer. The key challenge is to answer complex questions leveraging the retrieved knowledge, oftentimes still in large volume.

Complex questions, such as aggregation questions (e.g., *What is the total points scored by the New Orleans Pelicans in 2022-01?*) and reasoning questions (e.g., *Is memory management in Python handled through garbage collection?*), often require cross-referencing multiple pieces of KG information. We employ CoT prompting for the reasoning, the template is as follows (Summarization Prompts Skeleton in Appendix H, details in the Appendix I.3).

To further enhance performance, we fine-tune the CoT-based summarizer. To avoid the difficulties of manually labeling data (Sun et al., 2024c), we proposed the following method to generate the fine-tuning data and present the details in Figure 4.

1. For each query Q in the training set, we first prompt an LLM model to generate CoT-style reasoning together with the answer $\hat{A}$.
2. We then prompt LLM to decide the correctness of $\hat{A}$ by comparing with the ground truth answer A.
3. Based on the correctness judgment, we enhance the training data as follows. If the generated answer $\hat{A}$ is correct, we will take $\hat{A}$ together with the generated reasoning as the

ground truth; otherwise, we retain the original ground truth and replace the CoT prompt with stand non-CoT prompt.

4. The queries, the updated prompts, and the updated answers formulate our updated training set for Supervised Fine Tuning (SFT).

Our proposed CoT-based summarizer together with the fine-tuning data generation scheme is the biggest contributor to the end-to-end answer quality improvement. On the CRAG dataset, using plain CoT reasoning refrained from generating hallucinated answers for 18% of questions (mostly saying "I don't know" instead), and our fine-tuning further increased accuracy by 10% (Table 4).

5 Experiments

5.1 Datasets

To evaluate the performance of our method, we consider the widely used CRAG (Yang et al., 2024) benchmark, where the KGs are accessed through API functions. We use the KG questions in CRAG for conducting experiments.

We further adapted our method to the Open domain question set of the Head-to-tail benchmark (Head2Tail) (Sun et al., 2024b), where the DBPedia KG is utilized, and the content is accessed through SPARQL queries to showcase the generalizability of our approach under different KGQA settings.

Both the CRAG and Head2Tail categorize the questions into head, torso, and tail based on the popularity of the topic entity. We adopt the public validation set of CRAG as the training set and use the private testing set as the testing set. The statistics of the training and testing sets in the CRAG KG subset are presented in Table 9 of Appendix A. As for the Head2Tail dataset, we randomly sampled 375 questions from each of the head, torso, and tail categories to formulate the testing set, while the rest can be treated as the training set. We present the detailed statistics of the datasets used in Appendix A.

We also included other SPARQL-based datasets used by the state-of-the-art approaches: QALD-10-en (Usbeck et al., 2023), WebQSP (Yih et al., 2016), AdvHotpotQA (Ye and Durrett, 2022), and CWQ (Talmor and Berant, 2018).

5.2 Baselines

We compare our approach against the following baselines on both the CRAG and Head2Tail datasets. Details of baselines are in Appendix C.

- *Base models:* GPT-4o (Hurst et al., 2024), Llama-3.1-70B-Instruct (Dubey et al., 2024), GPT-4o (tool), and Llama-3.1-70B-Instruct (tool). The GPT-4o and Llama-3.1-70B-Instruct models, along with their tool-use variants (prompts are in Appendix I.4), serve as representatives for closed-source and open-source LLM solutions.
- *KDD Cup winners:* For CRAG, we include the top-2 KDD Cup CRAG winning solutions, db3 (Xia et al., 2024) and apex (Ouyang et al., 2024), which are state-of-the-art solutions on the CRAG benchmark.
- *Other state-of-the-art models:* We consider the following state-of-the-art methods on the SPARQL-based Head2Tail dataset: WikiSP (Xu et al., 2023), StructGPT (Jiang et al., 2023), and ToG (Sun et al., 2024a). Note that since CRAG does not support SPARQL querying, so we ran them only on Head2Tail. We further compared with the state-of-the-art results mentioned in ToG-2 (Ma et al., 2025) on QALD-10-en, WebQSP, AdvHotpotQA, and CWQ datasets. We did not evaluate ToG-2 on Head2Tail because ToG-2 is designed based on Wikidata and requires external pages to provide context, which is not compatible with the DBPeida KG used by Head2Tail. Besides, the extra information introduced by external pages would influence fairness of evaluation.

Finally, for fair comparison with KDD Cup winning solutions, which may use 8B models, we also included KERAG (8B), where all modules are developed based on Llama-3.1-8B-Instruct.

5.3 Evaluation

To evaluate the performance of the approaches on CRAG and Head2Tail, we adopted a similar auto-evaluation template used by CRAG (detailed in Appendix I.5). The Llama-3.1-70B-Instruct model is employed as the critic. Following the settings in CRAG, we classify the answers into accurate, missing, and hallucination, corresponding to correct, missing, and incorrect answers. We report the accuracy (A), miss rate (M), hallucination rate (H), and additionally the *truthfulness* score (T), which is defined as $T = 1 * A + 0 * M - 1 * H$, penalizing the cases where the methods hallucinate.

For QALD-10-en, WebQSP, AdvHotpotQA, and CWQ datasets, we align with their respective evaluation methods and metrics.

Table 1: KERAG improves over state-of-the-art by 7.1% on CRAG benchmark.

Model	Accu.	Hall.	Miss.	Truth.
GPT-4o	0.341	0.090	0.569	0.251
Llama	0.306	0.080	0.614	0.227
GPT-4o (tool)	0.362	**0.047**	0.592	0.315
Llama (tool)	0.220	0.057	0.723	0.163
apex (Ouyang et al., 2024)	0.652	0.194	0.154	0.458
db3 (Xia et al., 2024)	0.510	0.173	0.317	0.337
KERAG	**0.732**	0.202	**0.066**	**0.529**

The configurations are presented in Appendix E.

5.4 Experimental Results

We present the experimental results for the CRAG dataset in Table 1. We first notice that KERAG outperforms all baselines in CRAG, with an overall improvement of 7.1% in terms of the truthfulness score. Specifically, we observe that the LLM baselines (GPT-4o and Llama) perform poorly (accuracy < 40%) on the CRAG benchmark, which highlights the difficulty of the questions. In addition, although the tool-use variants of LLM have slightly lower hallucination rates, they fail to achieve high accuracy and have >55% missing rates. We attribute the high missing rate of tool-use variants to the fact that direct function chaining is likely to propagate planning errors to the retrieval stage, leading to low recall of relevant KG content. The KDD Cup winning solutions significantly outperformed baseline LLM solutions, obtaining higher accuracy (40-65%), at the price of much higher hallucination rate (~20%). Our KERAG solution achieves much higher accuracy (>70%), with very minor increase of hallucination rate; showcasing the superiority of multi-hop retrieval and filtering design by providing high recall of relevant content, thereby reducing the miss rate.

We observed similar trends on the Head2Tail benchmark, shown in Table 2. First, KERAG outperforms all the baselines in terms of overall accuracy and truthfulness scores, in particular significantly outperforming GPT-4o and Llama-3.1-70B-Instruct across all data splits. Furthermore, although we allow LLMs to write SPARQL queries to access the knowledge graph (tool variants), KERAG still shows overall truthfulness improvements of 10-15% compared to GPT-4o (tool) and Llama-3.1-70B-Instruct (tool). Additionally, KERAG outperforms WikiSP, StructGPT, and ToG by ~8% on truthfulness scores, with lower hallucinations. We report that the retrieval recall of our approach is 0.952, while the recall of ToG is 0.844, validating the advantages of our neighborhood-based exploration against the path-based exploration of ToG. Finally, we observed 91% accuracy on this dataset, which mainly contains simple questions, showing that KERAG has mostly conquered simple-question answering.

Table 2: KERAG improves over state-of-the-art by 7% on Head2Tail benchmark.

Model	Accu.	Hall.	Miss.	Truth.
GPT-4o	0.502	0.160	0.338	0.342
Llama	0.452	0.163	0.385	0.290
GPT-4o (tool)	0.799	0.035	0.166	0.764
Llama (tool)	0.752	**0.031**	0.217	0.721
WikiSP (Xu et al., 2023)	0.858	0.066	0.076	0.782
StructGPT (Jiang et al., 2023)	0.895	0.105	**0.000**	0.790
ToG (Sun et al., 2024a)	0.833	0.063	0.104	0.770
KERAG	**0.908**	0.049	0.043	**0.860**

Table 3: Experimental results on QALD-10-en, WebQSP, AdvHotpotQA, and CWQ datasets. The prior SOTA include the best known methods on each dataset: α: (Borroto et al., 2022); β: (Yu et al., 2022); γ: (Li et al., 2024b); δ: (Das et al., 2021). The baseline performance is directly taken from ToG and ToG-2. ToG-2 additionally links to external Wikipedia pages to get context information.

Model	QALD-10-en	WebQSP	AdvHotpotQA	CWQ
Prior SOTA	0.454^{α}	0.821^{β}	0.354^{γ}	$\mathbf{0.704}^{\delta}$
ToG	0.502	0.762	0.263	0.695
ToG-2	0.541	0.811	0.429	-
KERAG	**0.558**	**0.843**	**0.461**	0.702

We further present the results on QALD-10-en, WebQSP, AdvHotpotQA, and CWQ datasets in Table 3: our approach outperforms SOTA methods on QALD-10-en, WebQSP, and AdvHotpotQA datasets and achieves comparable performance against SOTAs on CWQ dataset.

5.5 Ablation Study

We conducted an ablation study to explore the effectiveness of each component in KERAG: 1) -multihop: considers only one-hop neighbors in the retriever; 2) -filter: without performing the filtering step; 3) -finetuning: without fine-tuning the CoT-based summarizer; 4) -CoT: replaces the CoT-based summarizer with a plain tuned summarizer.

We observed that the overall truthfulness of these four variants decreases relative to KERAG by 7.4%, 10.4%, 14.4%, 43.1%. As expected, multi-hop retrieval increased missing rate by 25%, with the price of increased hallucination rate (by 9%). Filtering reduced hallucination rate by removing noises in the retrieval results. The Chain-of-Thought prompting of the summarizer and the fine-tuning process improved the quality most, increasing accuracy by 9% and reducing hallucinations by 15%;

Table 4: Ablation Study on CRAG.

Model	Accu.	Hall.	Miss.	Truth.
KERAG	0.732	0.202	0.066	0.529
- multihop	0.587	0.112	0.301	0.474
- filter	0.721	0.232	0.047	0.490
- finetuning	0.626	0.173	0.201	0.453
- CoT	0.649	0.348	0.003	0.301

fine-tuning played an important role in reducing missing answers, especially for complex questions. Combining the four designs maximizes the performance of KERAG.

We further explored the effect of multi-hop retrieval on KGs by comparing the performance difference of our multi-hop retrieval strategy for one-hop, two-hop, and even n-hop on CRAG[2]. We present the results in Figure 5 of the Appendix. We observe that the accuracy increased significantly (15%) from one-hop to exhaustive retrieval; correspondingly, missing rate reduced significantly from 30% for one-hop to 7% for exhaustive. On the other hand, we also observe an increased hallucination rate from one-hop to two-hop (9%), brought by retrieval noises, thereby validating the necessity of a proper content filter and a summarizer with high reasoning ability in the pipeline.

5.6 Robustness

We explore the robustness of KERAG on head/torso/tail categorization. As shown in Tables 12 and 13 in Appendix F, GPT-4o and Llama-3.1-70B-Instruct models present a significant decreasing trend in terms of accuracy and truthfulness going from head to torso to tail categories, even with KG-based RAG. In contrast, as shown in Table 7, KERAG presents a relatively stable performance, narrowing down the gaps and demonstrating robustness. Interestingly, on Head2Tail KERAG has lower performance on head entities, because of challenges like knowledge overloading on head entities and ambiguous entity names.

To further analyze the generalizability of KERAG, we adapt our KERAG approach to a multi-entity settings, where the planner is asked to identify a list of entities instead of a single entity and hops from multiple topic entities to retrieve relevant information. In this way, our solution is generalized to the cases where retrieving information of multiple entities is necessary. In order to evaluate the performance of the generalized solution on multi-entity cases, we experiment the adapted KERAG model on comparison questions, which is the only question type that involves multiple entities in CRAG. As shown in Table 5, before applying the adaptation the KERAG model achieve truthfulness score (accuracy-hallucination rate) and accuracy of 0.7 and 0.817 respectively. We achieve truthfulness score and accuracy of 0.75 and 0.85 respectively with multi-entity adaptation, validating the feasibility and effectiveness of the adaptation.

Table 5: Multi-entity Generalizability.

Model	A	H	M	T
KERAG (single)	0.817	0.117	0.066	0.700
KERAG (multiple)	0.850	0.100	0.050	0.750

We present the experimental results by replacing the LLM backbones of KERAG in Table 6. We observe that the performance drops slightly when we replace the Llama-3.1 70B with Llama-3.1 8B. We additionally replace the backbone with DeepSeek-chat model (DeepSeek-V3) to see if a more advanced model brings performance uplift on CRAG dataset. The performance of KERAG (DeepSeek) further boosts the performance of KERAG (70B) by 2.4% and 11.8% in terms of overall accuracy and truthfulness scores (accuracy-hallucination rate). We believe that an insight is that advanced LLMs would reduce the hallucination risk in the planning step and thus bring performance uplift.

Table 6: Robustness of LLM Backbones.

Model	A	H	M	T
KERAG (8B)	0.713	0.208	0.080	0.505
KERAG (70B)	0.732	0.202	0.066	0.529
KERAG (DeepSeek)	0.756	0.109	0.135	0.647

We perform the error analysis on each component of KERAG as follows: For CRAG, The overall recall of content retrieval is 0.967 and the retrieval miss rate is 0.033. The summarization accuracy is 0.757, indicating that the main error lies in the summarization step. For Head2Tail, the entity linking accuracy is 0.957, the retrieval recall of the answer entity is 0.952, and the overall accuracy is 0.908, which indicates that the main errors occur in the entity linking and summarization.

5.7 Latency

We report the mean inference time of KERAG against the baselines in Table 8. Specifically, although our approach presents a slightly larger latency than apex on CRAG, it demonstrates a smaller latency than StructGPT and ToG on

[2]In our case, the API schema of CRAG limits $n \leq 3$.

Table 7: Head/torso/tail results of KERAG.

Data	A	H	M	T
CRAG-head	0.754	0.219	0.027	0.535
CRAG-torso	0.734	0.207	0.059	0.527
CRAG-tail	0.707	0.181	0.112	0.527
CRAG-overall	0.732	0.202	0.066	0.529
Head2Tail-head	0.883	0.067	0.051	0.816
Head2Tail-torso	0.917	0.045	0.037	0.872
Head2Tail-tail	0.925	0.035	0.040	0.891
Head2Tail-overall	0.908	0.049	0.043	0.860

Head2Tail dataset. In summary, we believe the overall latency of KERAG is acceptable.

Table 8: Mean Inference Time of KERAG and baselines.

Model	CRAG	Head2Tail
apex (Ouyang et al., 2024)	6.96s	-
StructGPT (Jiang et al., 2023)	-	4.46s
ToG (Sun et al., 2024a)	-	4.96s
KERAG	8.55s	3.18s

6 Conclusion

In conclusion, our proposed KERAG pipeline addresses key challenges in KG-based RAG by leveraging entity-level information, significantly reducing reliance on triple-level semantic parsing. By integrating a novel data generation scheme for fine-tuning chain-of-thought summarizers, KERAG enhances the ability to handle complex queries effectively. Its flexibility across various knowledge graphs, including those accessed via SPARQL or APIs, demonstrates broad applicability and generalizability. Experimental results confirm that KERAG outperforms existing state-of-the-art methods, making it a promising advancement in KGQA.

Limitations

Although our proposed KERAG approach demonstrates remarkable performance on both API-based and SPARQL-based knowledge graphs, we must acknowledge that the evaluation only covered CRAG, Head2Tail, QALD-10-en, WebQSP, AdvHotpotQA, and CWQ datasets. It is unknown how our proposed approach would perform in other KGs with different characteristics or domains. Furthermore, due to the multi-stage design of our approach, we have to admit that there may exist potential risks regarding error propagation in the pipeline. Our error analysis indicates that the main error might occur in entity linking and our current method does not focus on deploying a specific entity-linking mechanism. We believe that existing state-of-the-art entity linking approaches can be used as a pre-step to effectively integrate with our approach to improve the overall performance. We would like to point out that these limitations do not diminish the importance of our work; instead, they highlight potential areas for future research and enhancement.

Ethics Statement

In this section, we discuss the ethical considerations of our work from two aspects: 1) The knowledge graphs (KGs) utilized in our experiments include real-world named entities, such as individuals and organizations, along with their publicly accessible information, which cannot be anonymized when evaluating QA performance. The datasets referenced in this work (Yang et al., 2024; Sun et al., 2024b; Usbeck et al., 2023; Yih et al., 2016; Ye and Durrett, 2022; Talmor and Berant, 2018) were collected from publicly available online sources and have been extensively used in prior research. However, it is crucial to note that the use of these public datasets does not guarantee the absence of privacy violations. 2) Environmental Impact: The process of training and inferring models leads to energy consumption and carbon emissions.

Acknowledgements

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, Key Areas Special Project of Guangdong Provincial Universities 2024ZDZX1006, 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.

Nan Tang's work is partially supported by Guangdong provincial project 2023CX10X008.

References

Manuel Borroto, Francesco Ricca, Bernardo Cuteri, and Vito Barbara. 2022. Sparql-qa enters the qald challenge. In *Proceedings of the 7th Natural Language Interfaces for the Web of Data (NLIWoD) co-located with the 19th European Semantic Web Conference, Hersonissos, Greece*, volume 3196, pages 25–31.

Zixin Chen, Sicheng Song, Kashun Shum, Yanna Lin, Rui Sheng, and Huamin Qu. 2025. Unmasking deceptive visuals: Benchmarking multimodal large language models on misleading chart question answering. *arXiv preprint arXiv:2503.18172*.

Rajarshi Das, Manzil Zaheer, Dung Thai, Ameya Godbole, Ethan Perez, Jay-Yoon Lee, Lizhen Tan, Lazaros Polymenakos, and Andrew Mccallum. 2021. Case-based reasoning for natural language queries over knowledge bases. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9594–9611.

Xin Luna Dong. 2024. 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. Association for Computing Machinery.

Xin Luna Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Kevin Murphy, Shaohua Sun, and Wei Zhang. 2014. From data fusion to knowledge fusion. *Proceedings of the VLDB Endowment*, 7(10):881–892.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6491–6501.

Yu Gu, Xiang Deng, and Yu Su. 2023. Don't generate, discriminate: A proposal for grounding language models to real-world environments. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4928–4949.

Yu Gu and Yu Su. 2022. Arcaneqa: Dynamic program induction and contextualized encoding for knowledge base question answering. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1718–1731.

Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Improving multi-hop knowledge base question answering by learning intermediate supervision signals. In *Proceedings of the 14th ACM international conference on web search and data mining*, pages 553–561.

Aidan Hogan, Xin Luna Dong, Denny Vrandečić, and Gerhard Weikum. 2025. Large language models, knowledge graphs and search engines: A crossroads for answering users' questions. *arXiv preprint arXiv:2501.06699*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Xixin Hu, Xuan Wu, Yiheng Shu, and Yuzhong Qu. 2022. Logical form generation via multi-task learning for complex question answering over knowledge bases. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1687–1696.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Structgpt: A general framework for large language model to reason over structured data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9237–9251.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781.

Yunshi Lan and Jing Jiang. 2020. Query graph generation for answering multi-hop complex questions from knowledge bases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 969–974.

Yunshi Lan, Shuohang Wang, and Jing Jiang. 2019. Knowledge base question answering with topic units.(2019). In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 5046–5052.

Haobo Li, Eunseo Jung, Zixin Chen, Zhaowei Wang, Yueya Wang, Huamin Qu, and Alexis Kai Hon Lau. 2025. Pipe: Physics-informed position encoding for alignment of satellite images and time series. *arXiv preprint arXiv:2506.14786*.

Haobo Li, Zhaowei Wang, Jiachen Wang, YueYa Wang, Alexis Kai Hon Lau, and Huamin Qu. 2024a. Cllmate: A multimodal benchmark for weather and climate events forecasting. *arXiv preprint arXiv:2409.19058*.

Tianle Li, Yushi Sun, Shang-ling Hsu, Yanjia Li, and Raymond Chi-Wing Wong. 2022. Fake news detection with heterogeneous transformer. *arXiv preprint arXiv:2205.03100*.

Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Shafiq Joty, Soujanya Poria, and Lidong Bing. 2024b. Chain-of-knowledge: Grounding large language models via dynamic knowledge adapting over heterogeneous sources. In *The Twelfth International Conference on Learning Representations*.

Kangqi Luo, Fengli Lin, Xusheng Luo, and Kenny Zhu. 2018. Knowledge base question answering via encoding of complex query graphs. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2185–2194.

Shengjie Ma, Chengjin Xu, Xuhui Jiang, Muzhi Li, Huaren Qu, Cehao Yang, Jiaxin Mao, and Jian Guo. 2025. Think-on-graph 2.0: Deep and faithful large language model reasoning with knowledge-guided retrieval augmented generation. In *The Thirteenth International Conference on Learning Representations*.

Costas Mavromatis and George Karypis. 2022. Rearev: Adaptive reasoning for question answering over knowledge graphs. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2447–2458.

Bo Ni, Zheyuan Liu, Leyao Wang, Yongjia Lei, Yuying Zhao, Xueqi Cheng, Qingkai Zeng, Luna Dong, Yinglong Xia, Krishnaram Kenthapadi, and 1 others. 2025. Towards trustworthy retrieval augmented generation for large language models: A survey. *arXiv preprint arXiv:2502.06872*.

Barlas Oguz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Scott Yih. 2022. Unik-qa: Unified representations of structured and unstructured knowledge for open-domain question answering. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1535–1546.

Jie Ouyang, Yucong Luo, Mingyue Cheng, Daoyu Wang, Shuo Yu, Qi Liu, and Enhong Chen. 2024. Revisiting the solution of meta kdd cup 2024: Crag. *arXiv preprint arXiv:2409.15337*.

Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 4498–4507.

Priyanka Sen, Armin Oliya, and Amir Saffari. 2021. Expanding end-to-end question answering on differentiable knowledge graphs with intersection. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8805–8812.

Jiaxin Shi, Shulin Cao, Lei Hou, Juanzi Li, and Hanwang Zhang. 2021. Transfernet: An effective and transparent framework for multi-hop question answering over relation graph. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4149–4158.

Yiheng Shu, Zhiwei Yu, Yuhan Li, Börje Karlsson, Tingting Ma, Yuzhong Qu, and Chin-Yew Lin. 2022. Tiara: Multi-grained retrieval for robust question answering over large knowledge base. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8108–8121.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. 2024a. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *The Twelfth International Conference on Learning Representations*.

Kai Sun, Yifan Xu, Hanwen Zha, Yue Liu, and Xin Luna Dong. 2024b. Head-to-tail: How knowledgeable are large language models (llms)? aka will llms replace knowledge graphs? In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 311–325.

Yushi Sun, Kai Sun, Xiao Yang, and Nan Tang. 2025. Knowledge internalized in llms. In *Handbook on Neurosymbolic AI and Knowledge Graphs*, pages 230–255. IOS Press.

Yushi Sun, Jiachuan Wang, Peng Cheng, Libin Zheng, Lei Chen, and Jian Yin. 2024c. Cross-domain-aware worker selection with training for crowdsourced annotation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 249–262. IEEE.

Yushi Sun, Hao Xin, and Lei Chen. 2023. Reca: Related tables enhanced column semantic type annotation framework. *Proceedings of the VLDB Endowment*, 16(6):1319–1331.

Yushi Sun, Hao Xin, Kai Sun, Yifan Ethan Xu, Xiao Yang, Xin Luna Dong, Nan Tang, and Lei Chen. 2024d. Are large language models a good replacement of taxonomies? *Proceedings of the VLDB Endowment*, 17(11):2919–2932.

Alon Talmor and Jonathan Berant. 2018. 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. Association for Computational Linguistics.

Ricardo Usbeck, Xi Yan, Aleksandr Perevalov, Longquan Jiang, Julius Schulz, Angelie Kraft, Cedric Möller, Junbo Huang, Jan Reineke, Axel-Cyrille Ngonga Ngomo, Muhammad Saleem, and

Andreas Both. 2023. Qald-10 – the 10th challenge on question answering over linked data. *Semantic Web*.

Xingbo Wang, Samantha L Huey, Rui Sheng, Saurabh Mehta, and Fei Wang. 2024. Scidasynth: Interactive structured knowledge extraction and synthesis from scientific literature with large language model. *arXiv preprint arXiv:2404.13765*.

Yikuan Xia, Jiazun Chen, and Jun Gao. 2024. Winning solution for meta kdd cup'24. *arXiv preprint arXiv:2410.00005*.

Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I Wang, and 1 others. 2022. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 602–631.

Silei Xu, Shicheng Liu, Theo Culhane, Elizaveta Pertseva, Meng-Hsi Wu, Sina Semnani, and Monica Lam. 2023. Fine-tuned llms know more, hallucinate less with few-shot sequence-to-sequence semantic parsing over wikidata. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5778–5791.

Xiao Yang, Kai Sun, Hao Xin, Yushi Sun, Nikita Bhalla, Xiangsen Chen, Sajal Choudhary, Rongze Daniel Gui, Ziran Will Jiang, Ziyu Jiang, Lingkun Kong, Brian Moran, Jiaqi Wang, Yifan Ethan Xu, An Yan, Chenyu Yang, Eting Yuan, Hanwen Zha, Nan Tang, and 8 others. 2024. Crag – comprehensive rag benchmark. *arXiv preprint arXiv:2406.04744*.

Xi Ye and Greg Durrett. 2022. The unreliability of explanations in few-shot prompting for textual reasoning. *Advances in neural information processing systems*, 35:30378–30392.

Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the Joint Conference of the 53rd Annual Meeting of the ACL and the 7th International Joint Conference on Natural Language Processing of the AFNLP*.

Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206.

Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Yang Wang, Zhiguo Wang, and Bing Xiang. 2022. Decaf: Joint decoding of answers and logical forms for question answering over knowledge bases. In *The Eleventh International Conference on Learning Representations*.

Jing Zhang, Xiaokang Zhang, Jifan Yu, Jian Tang, Jie Tang, Cuiping Li, and Hong Chen. 2022. Subgraph retrieval enhanced model for multi-hop knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5773–5784.

Lingxi Zhang, Jing Zhang, Yanling Wang, Shulin Cao, Xinmei Huang, Cuiping Li, Hong Chen, and Juanzi Li. 2023. Fc-kbqa: A fine-to-coarse composition framework for knowledge base question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1002–1017.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, and 1 others. 2023. Pytorch fsdp: Experiences on scaling fully sharded data parallel. *Proceedings of the VLDB Endowment*, 16(12):3848–3860.

A Dataset Details

The statistics of training and testing sets in the CRAG KG subset is presented in Table 9.

Table 9: Training and testing sets statistics of CRAG.

	Training	**Testing**
head	219	187
torso	197	203
tail	188	188
overall	604	578

The CRAG benchmark includes KGs from open, movie, finance, sports, and music domains. The question types include simple, simple with condition, set, comparison, aggregation, multi-hop, post-processing, and false premise questions. There are also four dynamism considered, including real-time, fast-changing, slow-changing, and static questions. The proportions of hops required for questions in CRAG are: 43% requires one hop, 49% requires two hops, and 8% requires three hops. A detailed statistics about each question type is presented in Table 10.

The Open domain question set of the Head2Tail benchmark consists of in total 9219 questions, each of the head, torso, tail data split consist of 3073 questions. We randomly sampled 375 questions from each data split to formulate the testing set, which guarantees a margin of error of less than 5%.

The licenses of CRAG and Head2Tail are Attribution-NonCommercial 4.0 International.

Table 10: statistics of each question type in different question splits.

	head	torso	tail
open	48	48	48
finance	125	134	112
movie	121	120	118
music	35	41	14
sports	77	57	84
simple	205	218	187
simple_w_condition	59	43	57
set	19	12	13
comparison	41	44	34
aggregation	33	35	38
multi-hop	14	13	11
post-processing	6	6	6
false premise	29	29	30
real-time	78	85	72
fast-changing	67	59	76
slow-changing	46	51	45
static	215	205	183

The licenses of QALD-10-en, WebQSP, AdvHotpotQA, and CWQ are MIT license, CC-BY-SA-4.0, CC-BY-SA-4.0, and Apache-2.0.

B Advantages of KERAG over State-of-the-arts.

We further discuss the advantages of KERAG over the state-of-the-arts: ToG (Sun et al., 2024a) and ToG-2 (Ma et al., 2025): 1) Our retrieval process is more relax: instead of prompting the LLMs to specify the retrieval paths, we gather all relevant neighborhood and rely more on the strong summarization ability of LLMs to perform filtering and CoT-based summarization. 2) Our method is more efficient: First, our filtering and expansion decisions are made at a schema level whereas the existing approaches make instance-level decisions, so we are much more efficient. Second, assuming D hops, we invoke LLM $D+2$ times, whereas ToG and ToG-2 invoke LLM $ND + D + 2$ times (N is #paths investigated). 3) Our method is more generally applicable. It applies not only to SPARQL-based KGs, but also to API-based KGs, where each entity node may contain long text, with attributes in raw text, JSON style, or tabular format. As such, we can run KERAG on the complex CRAG benchmark, but cannot easily adapt ToG, or ToG-2 on CRAG.

C Baseline Details

- GPT-4o (Hurst et al., 2024): The Generative Pre-trained Transformers series consists of advanced large language models developed by OpenAI. We chose GPT-4o as the representative for our evaluation. The model is proprietary and can only be accessed via API functions.

- GPT-4o (tool): The tool-use variant of the GPT-4o model. We designed a prompt with KG access interfaces to allow the GPT-4o model writes its own API function calls or SPARQL queries. In-context examples are provided to the model. Please check Appendix I.4 for detailed prompts.

- Llama-3.1-70B-Instruct (Dubey et al., 2024): The Llama-3.1 series is a new collection of large language models released by Meta in July 2024. We adopted the Llama-3.1-70B model with instruct settings to conduct the experiments.

- Llama-3.1-70B-Instruct (tool): Similar to GPT-4o (tool), we allow the Llama-3.1-70B-Instruct model to write API function calls or SPARQL queries.

- db3 (Xia et al., 2024): db3 is one of the winning solutions of Meta KDD Cup 2024, which represents the state-of-the-art performance on CRAG benchmark. The design of db3 jointly considers the inputs from KG and web content. We enter "<EMPTY>" into the web content module of db3 to adapt it to our KGQA settings.

- apex (Ouyang et al., 2024): apex is one of the winning solutions of Meta KDD Cup 2024, which represents the state-of-the-art performance on CRAG benchmark. A router-based adaptive RAG pipeline is introduced to retrieve relevant KG content for answer generation. Similar to db3, we input "<EMPTY>" as the web content input to the model to adapt the apex solution to our KGQA settings.

- WikiSP (Xu et al., 2023): WikiSP is a semantic parsing KGQA approach. Following the instructions given by WikiSP, we trained the semantic parser on Head2Tail benchmark and replaced the original GPT-3 QA module with

the advanced GPT-4o module in the experiments. WikiSP serves as the state-of-the-art representative of semantic parsing KGQA approaches.

- StructGPT (Jiang et al., 2023): StructGPT is a general framework that facilitate the reasoning over structured data. Structured data content such as KGs is retrieved and linearized for the reasoning of LLMs. We selected StructGPT as the state-of-the-art representative for LLM-based KGQA research. Similar to WikiSP, we replace the original ChatGPT core model used by StructGPT with the up-to-date GPT-4o model for fair comparison.
- sparql-qa (Borroto et al., 2022): sparql-qa uses a neural architecture combining NMT and NER with input processing and QQT format to translate natural language to SPARQL.
- Decaf (Yu et al., 2022): Decaf jointly generates answers and logical forms for knowledge base question answering, combines their advantages, uses text retrieval instead of entity linking to enhance generality.
- CoK (Li et al., 2024b): CoK enhances LLMs by dynamically incorporating heterogeneous knowledge sources through three stage reasoning preparation, dynamic knowledge adapting with an adaptive query generator, and answer consolidation to reduce hallucination and improve factual accuracy.
- CBR (Das et al., 2021): CBR is a neuro-symbolic method. It retrieves similar question cases, reuses their logical form components, and revises the generated form via KB embeddings to handle complex KBQA and unseen relations.
- ToG (Sun et al., 2024a): ToG integrates large language models (LLMs) with knowledge graphs (KGs). Through beam search, LLMs are used to iteratively explore reasoning paths on KGs, so as to enhance the deep reasoning ability of LLMs, improve knowledge traceability and correctness, and achieve state-of-the-art performance on multiple datasets.
- ToG-2 (Ma et al., 2025): Following ToG, ToG-2 is a hybrid RAG framework that tightly couples knowledge graphs and documents for iterative retrieval, enabling deep and faithful reasoning in LLMs with state-of-the-art performance on multiple datasets.

D Evaluation Details

We adopted the auto-evaluation scheme used in CRAG (Yang et al., 2024) to evaluate the correctness of the answers on both CRAG and Head2Tail benchmarks. According to (Yang et al., 2024), the detailed criterion for accurate, missing, and hallucination is defined as follows:

- Correct: The response correctly answers the user's question and contains no hallucinated content, or 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 adopted the Llama-3.1-70B-Instruct model as the auto-evaluation critic to evaluate all the responses. The prompts used for auto-evaluation are presented in Appendix I.5.

We present the auto-evaluation performance of our Llama-3.1-70B-Instruct model in Table 11.

Table 11: The Auto-evaluation Performance of Llama-3.1 70B Instruct Model.

	Llama-3.1 70B
correct answers	98.2%
incorrect answers	96.8%
missing	100%
overall reliability	98.4%

E Configuration Details

All experiments were performed on four NVIDIA A100 GPUs (80GB).

For the CRAG benchmark, the Llama-3.1-70B-Instruct model is used as the base model for planning. We adopted FSDP (Zhao et al., 2023) and LoRA (Hu et al., 2021) to efficiently fine-tune the summarizer, which is based on the Llama-3.1-8B-Instruct model. The temperatures are set to 0 for

all modules. The summarizer is fine-tuned for 20 epochs (in total 6 GPU hours), with a learning rate of 5e-5. The proportion of CoT data in the training set is 81.6%. The non-CoT data is merged with CoT data for joint training and the training process is one-off.

For the configurations of KERAG on Head2Tail dataset, Llama-3-8B-Instruct is the base model used for planning, and summarization. Chain-of-Thought reasoning and summarizer finetuning are not applied to the Head2Tail dataset. Temperatures of the models are set to 0 for stable generation of the answer. We include a 3-gram similarity based method to first identify the potential (entity, predicate) pairs (maximum three candidates per query) based on the entity and relation lists in Head2Tail and then apply the Main Entity Extraction prompts to identify the main entity of the query. Since there is only one DBPedia KG in Head2Tail and the questions in Head2Tail are not time-sensitive, we don't need to perform domain and time information extraction. The similarity-based filtering is applied. Specifically, we use DPR (Karpukhin et al., 2020) to compute and rank the similarity between the candidate predicates in the retrieved triples and the query with the main entity removed. Top-30 predicates are preserved in the retrieval step for the summarizer to generate responses. Specifically, the average input number of predicates is 53.2, and up to 30 predicates are selected for each question.

The configurations of KERAG on the QALD-10-en, WebQSP, AdvHotpotQA, and CWQ datasets are as follow: We use Llama-3-8B-Instruct as the base model for the planning step. The llama-7b-wikiwebquestions-qald7 model[3] (Xu et al., 2023) is used to aid the process of main entity extraction. The plain GPT-4o model is used as the summarizer of the model. Chain-of-Thought reasoning and summarizer finetuning are not applied. Similarity-based DPR filtering is used to filter irrelevant predicates. The planning and summarization prompts are similar to those used on Head2Tail.

F Head-Torso-Tail Experiment Results

We present the detailed results regarding the head/torso/tail categorization of CRAG and Head2Tail datasets in Tables 12 and 13.

[3]https://huggingface.co/stanford-oval/llama-7b-wikiwebquestions-qald7

Table 12: Head/torso/tail experimental results on CRAG.

	Model	A	H	M	T
head	GPT-4o	0.401	**0.086**	0.513	0.316
	Llama	0.390	0.096	0.513	0.294
	KERAG (8B)	0.738	0.225	0.037	0.513
	KERAG	**0.754**	0.219	**0.027**	**0.535**
torso	GPT-4o	0.355	**0.089**	0.557	0.266
	Llama	0.305	**0.089**	0.606	0.217
	KERAG (8B)	0.714	0.192	0.094	0.522
	KERAG	**0.734**	0.207	**0.059**	**0.527**
tail	GPT-4o	0.266	0.096	0.638	0.170
	Llama	0.223	**0.053**	0.723	0.170
	KERAG (8B)	0.686	0.207	**0.106**	0.479
	KERAG	**0.707**	0.181	0.112	**0.527**

Table 13: Head/torso/tail experimental results on Head2Tail.

	Model	A	H	M	T
head	GPT-4o	0.613	0.181	0.205	0.432
	Llama	0.629	0.197	0.173	0.432
	KERAG	**0.883**	**0.067**	**0.051**	**0.816**
torso	GPT-4o	0.517	0.133	0.349	0.384
	Llama	0.440	0.139	0.421	0.301
	KERAG	**0.917**	**0.045**	**0.037**	**0.872**
tail	GPT-4o	0.376	0.165	0.459	0.211
	Llama	0.288	0.152	0.560	0.136
	KERAG	**0.925**	**0.035**	**0.040**	**0.891**

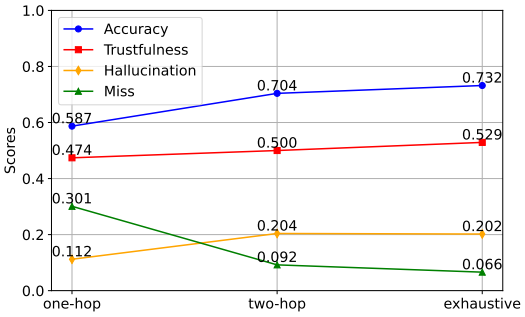

Figure 5: Comparison between one-hop, two-hop, and exhaustive retrieval on CRAG.

G Full Ablation Study Results

The full ablation study results with head/torso/tail categorization on CRAG are presented in Table 14.

H Templates

Planning Prompts Skeleton

You are given a Query [Q], please extract the topic entity [E] from the Query.
Determine the domain the query is about. The domain should be one of the following: [Set of KG domains].
[Examples] (Optional)
[Further Instructions] (Optional)

Table 14: Ablation Study on CRAG.

	Model	A	H	M	T
head	KERAG	0.754	0.219	0.027	0.535
	- multihop	0.626	0.150	0.225	0.476
	- filter	0.738	0.246	0.016	0.492
	- finetuning	0.663	0.203	0.134	0.460
	- CoT	0.658	0.342	0.000	0.316
torso	KERAG	0.734	0.207	0.059	0.527
	- multihop	0.596	0.094	0.310	0.502
	- filter	0.704	0.246	0.049	0.458
	- finetuning	0.640	0.172	0.187	0.468
	- CoT	0.640	0.360	0.000	0.281
tail	KERAG	0.707	0.181	0.112	0.527
	- multihop	0.537	0.096	0.367	0.441
	- filter	0.703	0.202	0.074	0.501
	- finetuning	0.574	0.144	0.282	0.431
	- CoT	0.649	0.340	0.010	0.309
overall	KERAG	0.732	0.202	0.066	0.529
	- multihop	0.587	0.112	0.301	0.474
	- filter	0.721	0.232	0.047	0.490
	- finetuning	0.626	0.173	0.201	0.453
	- CoT	0.649	0.348	0.003	0.301

Filtering Prompts Skeleton

Given the user query [Q] with detected topic entity [E] and the current [retrieval plan], you will also have access to the following functions [a set of functions/predicates with ontology] to perform additional hops based on the current plan.
Decide which functions/predicates are relevant to answer the query based on the ontology and if the composed retrieval plan sufficient for the question answering.
[Examples] (Optional)
[Further Instructions] (Optional)

Summarization Prompts Skeleton

Please provide a brief answer to the question [Q] based on your knowledge and the following content. Answer "I don't know" if you are not confident of your answer. Please think step by step.
[Retrieved Content (Triple form)]

I Prompts

We present the detailed prompts used in our experiments. Italicized text enclosed in pointed brackets represents the input variables.

I.1 Planning Prompts

Planning Prompts of CRAG

Main Entity and Domain Extraction

You are an agent 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 domain applies, 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, amd 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'.
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: 'auther', 'grammy award count', 'release country', 'release date'.
For 'sports' queries, these are possible keys: - 'sport_type': one of 'basketball', 'soccer', 'other'
- 'tournament': such as NBA, World Cup, Olympic.
- 'team': teams that user interested in.
Return the results in a FLAT json.
NEVER include ANY EXPLANATION or NOTE in the output, ONLY OUTPUT JSON!!!
Here are some examples:
Query
what was the volume of trades for rcm on the last day?
Your extracted JSON should be: "domain": "finance", "market_identifier": "rcm", "metric": "volume of trades"
Query
on 2022-10-11, how many points did bulls put up in their game?
Your extracted JSON should be: "domain": "sports", "sport_type": "basketball", "team": "chicago bulls"

CoT-based Time Extraction

Please extract the time frame that user interested in. When datetime is not explicitly mentioned, use 'Query Time' as default. Enclose your datetime extracted with <>! Use '∼' to indicate time period.
Here are some examples:
Query
what was the volume of trades for rcm on the last day?
Query Time

02/28/2024, 08:04:08 PT
Question: What is the user interested time frame of the Query? Please think step by step.
Your output: Since the query is asking about the last day of the Query Time, the time frame should be <02/27/2024>.
Query
which team did boston celtics take on in their matchup on 2023-05-29?
Query Time
03/15/2024, 16:05:17 PT
Question: What is the user interested time frame of the Query? Please think step by step.
Your output: The time frame should be <05/29/2023>, which is explicitly stated in the Query.
Query
on which date did sgml distribute dividends the first time?
Query Time
02/28/2024, 08:25:10 PT
Question: What is the user interested time frame of the Query? Please think step by step.
Your output: The time frame should be <02/28/2024>. Since there is no datetime explicitly mentioned and we take the Query Time as the default answer.
Query
what's the schedule looking like for west ham's next game in eng-premier league?
Query Time
03/15/2024, 15:48:32 PT
Question: What is the user interested time frame of the Query? Please think step by step.
Your output: The time frame should be <03/15/2024 $\sim$ future>. Since there is no datetime that can be confidently inferred from the query, we take the Query Time to ($\sim$) future time period as the answer.
Query
on average, what was the daily high stock price of xpev over the past week?
Query Time
02/28/2024, 08:17:40 PT
Question: What is the user interested time frame of the Query? Please think step by step.
Your output: The time frame should be <02/21/2024 $\sim$ 02/28/2024>. Since the question is asking for the value over the past week. We take 02/21/2024 $\sim$ 02/28/2024 time period as the answer.

Planning Prompts of Head2Tail

Main Entity Extraction

You are a professional semantic parsing expert. Please provide your answer by giving a brief (entity, predicate) pair. If no appropriate (entity, predicate) pair can be extracted, please respond (No, No).
[Instruction]: The task is to determine the most appropriate (entity, predicate) pair based on a given question and a list of potential (entity, predicate) pair list extracted by a fuzzy-matching-based semantic parser. MUST NOT provide answer to the question directly.
[EXAMPLE]: Question: What aircraft bomber was used by the South Vietnam Air Force? Pair List: [('Vietnam', 'Air_Force'), ('South_Vietnam_Air_Force', 'aircraftBomber')]; Your Answer: ('South_Vietnam_Air_Force', 'aircraftBomber')
[TASK]: Select the most appropriate (entity, predicate) based on this question and entity pair list: Question: <*Input Question*> Pair List: <*Candidate List*>.

I.2 Filtering Prompts

We present an exemplar filtering prompt used in the finance KG of CRAG as an example. The filtering prompts on other KGs can be generated by replacing the function descriptions and in context example accordingly.

Filtering Prompts of CRAG

Tool-based Filtering on Finance KG

You have access to the following functions:
Use the function 'finance_get_price_history' to 'Return daily Open price, Close price, High price, Low price and trading Volume.': {"type": "function", "function": {"name": "finance_get_price_history", "description": "Return daily Open price, Close price, High price, Low price and trading Volume.", "parameters": {"type": "object", "properties": {"ticker_name": {"type": "string", "description": "The ticker name of the stock interested."}}, "required": ["ticker_name"]}}}
Use the function 'finance_get_detailed_price_history' to 'Return minute-level Open price, Close price, High price, Low price and trading Volume.': {"type": "function", "function": {"name": "finance_get_detailed_price_history", "description": "Return minute-level Open price, Close price, High price, Low price and trading Volume.", "parameters": {"type": "object", "properties": {"ticker_name": {"type": "string", "description": "The ticker name of the stock interested."}}, "required": ["ticker_name"]}}}
Use the function 'finance_get_dividends_history' to 'Return dividend history of a ticker.': {"type": "function", "function": {"name": "finance_get_dividends_history", "description": "Return dividend history of a ticker.", "parameters": {"type": "object", "properties": {"ticker_name": {"type": "string", "description": "The ticker name of the stock interested."}}, "required": ["ticker_name"]}}}
Use the function 'finance_get_market_capitalization' to 'Return the market capitalization of a ticker.': {"type": "function", "function": {"name": "finance_get_market_capitalization", "description": "Return the market capitalization of a ticker.", "parameters": {"type": "object", "properties": {"ticker_name": {"type": "string", "description": "The ticker name of the stock interested."}}, "required": ["ticker_name"]}}}
Use the function 'finance_get_eps' to 'Return earnings per share of a ticker.': {"type": "function", "function": {"name": "finance_get_eps", "description": "Return earnings per share of a ticker.", "parameters": {"type": "object", "properties": {"ticker_name": {"type": "string", "description": "The ticker name of the stock interested."}}, "required": ["ticker_name"]}}}
Use the function 'finance_get_pe_ratio' to 'Return price-to-earnings ratio of a ticker.': {"type": "function", "function": {"name": "finance_get_pe_ratio", "description": "Return price-to-earnings ratio of a ticker.", "parameters": {"type": "object", "properties": {"ticker_name": {"type": "string", "description": "The ticker name of the stock interested."}}, "required": ["ticker_name"]}}}
Use the function 'finance_get_info' to 'Return rough meta data of a ticker.': {"type": "function", "function": {"name": "finance_get_info", "description": "Return rough meta data of a ticker.", "parameters": {"type": "object", "properties": {"ticker_name": {"type": "string", "description": "The ticker name of the stock interested."}}, "required": ["ticker_name"]}}}

If you choose to call a function ONLY reply in the following format with no prefix or suffix:
Question: What is the price of Meta currently? Your answer: <function=detailedPriceHistoryTool></function>
Reminder:
- Function calls MUST follow the specified format, start with <function= and end with </function>
- Only call one function at a time
- Put the entire function call reply on one line
- If there is no function call available, answer I don't know.'
Question: <*Question*>

Multi-hop Prompts of Head2Tail

Multi-hop Retrieval Boundary Prompts

You will be given a QUESTION and a set of retrieved TRIPLEs from DBpedia. Your task is to indicate whether the currently retrieved content is sufficient for you to answer the QUESTION. If you need to have more retrieved triples, respond <NO>. If you think the subject entity is wrongly linked, respond <NA>. Otherwise, if you think the current information is sufficient, respond <YES>. Only answer <NO>/<NA>/<YES>!!!
QUESTION: <*Input Question*>
DBpedia TRIPLEs: <*Retrieved triples*>

I.3 Summarization Prompts

Summarization Prompts of CRAG

CoT-based Prompt

Please provide a brief answer as short as possible to the question based on your own knowledge and the following relevant CONTENT extracted from Knowledge Base. Answer "I don't know" if you are not confident of your answer. Please think step by step.
The current query time is: <*Query Time*>.
<*Question*>
CONTENT: <*Refined KG Content*>

Base Prompt

Please provide a brief answer as short as possible to the question based on your own knowledge and the following relevant CONTENT extracted from Knowledge Base. Answer "I don't know" if you are not confident of your answer.
The current query time is: <*Query Time*>.
<*Question*>
CONTENT: <*Refined KG Content*>

Summarization Prompts of Head2Tail

Summarization Prompt

Please provide a brief answer as short as possible to the question based on your own knowledge and the following relevant TRIPLEs (subject, predicate, object) from DBpedia. Answer "I don't know" if you are not confident of your answer.
TRIPLE: <*triple_1*>
TRIPLE: <*triple_2*>
.....
TRIPLE: <*triple_n*>
<*Question*>

I.4 Baselines Tool-use Prompts

Tool-use Prompts for GPT-4o and Llama

Tool-use Prompts of CRAG

Please select the proper APIs to retrieve relevant content to the question.
You have access to the following functions:
Use the function 'open_search_entity_by_name' to 'Search for entities by name in the Open domain.': {"type": "function", "function": {"name": "open_search_entity_by_name", "description": "Search for entities by name in the Open domain.", "parameters": {"type": "object", "properties": {"query": {"type": "string", "description": "The name of the entity interested."}}, "required": ["query"]}}}
Use the function 'open_get_entity' to 'Retrieve detailed information about an entity in the Open domain.': {"type": "function", "function": {"name": "open_get_entity", "description": "Retrieve detailed information about an entity in the Open domain.", "parameters": {"type": "object", "properties": {"entity": {"type": "string", "description": "The name of the entity interested."}}, "required": ["entity"]}}}
Use the function 'movie_get_person_info' to 'Get information about a person related to movies.': {"type": "function", "function": {"name": "movie_get_person_info", "description": "Get information about a person related to movies.", "parameters": {"type": "object", "properties": {"person_name": {"type": "string", "description": "The name of the person interested."}}, "required": ["person_name"]}}}
Use the function 'movie_get_movie_info' to 'Get information about a movie.': {"type": "function", "function": {"name": "movie_get_movie_info", "description": "Get information about a movie.", "parameters": {"type": "object", "properties": {"movie_name": {"type": "string", "description": "The name of the movie interested."}}, "required": ["movie_name"]}}}
Use the function 'movie_get_year_info' to 'Get information about movies released in a specific year.': {"type": "function", "function": {"name": "movie_get_year_info", "description": "Get information about movies released in a specific year.", "parameters": {"type": "object", "properties": {"year": {"type": "string", "description": "The year interested."}}, "required": ["year"]}}}
Use the function 'movie_get_person_info_by_id' to 'Get person information by their unique ID.': {"type": "function", "function": {"name": "movie_get_person_info_by_id", "description": "Get person information by their unique ID.", "parameters": {"type": "object", "properties": {"person_id": {"type": "integer", "description": "The person id interested."}}, "required": ["person_id"]}}}
Use the function 'movie_get_movie_info_by_id' to 'Get movie information by its unique ID.': {"type": "function", "function": {"name": "movie_get_movie_info_by_id", "description": "Get movie information by its unique ID.", "parameters": {"type": "object", "properties": {"movie_id": {"type": "integer", "description": "The movie id interested."}}, "required": ["movie_id"]}}}
Use the function 'finance_get_company_name' to 'Search for company names in the finance domain.': {"type": "function", "function": {"name": "finance_get_company_name",

"description": "Search for company names in the finance domain.", "parameters": {"type": "object", "properties": {"query": {"type": "string", "description": "The query interested."}}, "required": ["query"]}}}
Use the function 'finance_get_ticker_by_name' to 'Retrieve the ticker symbol for a given company name.': {"type": "function", "function": {"name": "finance_get_ticker_by_name", "description": "Retrieve the ticker symbol for a given company name.", "parameters": {"type": "object", "properties": {"query": {"type": "string", "description": "The query interested."}}, "required": ["query"]}}}
Use the function 'finance_get_price_history' to 'Return daily Open price, Close price, High price, Low price and trading Volume.': {"type": "function", "function": {"name": "finance_get_price_history", "description": "Return daily Open price, Close price, High price, Low price and trading Volume.", "parameters": {"type": "object", "properties": {"ticker_name": {"type": "string", "description": "The ticker name of the stock interested."}}, "required": ["ticker_name"]}}}
Use the function 'finance_get_detailed_price_history' to 'Return minute-level Open price, Close price, High price, Low price and trading Volume.': {"type": "function", "function": {"name": "finance_get_detailed_price_history", "description": "Return minute-level Open price, Close price, High price, Low price and trading Volume.", "parameters": {"type": "object", "properties": {"ticker_name": {"type": "string", "description": "The ticker name of the stock interested."}}, "required": ["ticker_name"]}}}
Use the function 'finance_get_dividends_history' to 'Return dividend history of a ticker.': {"type": "function", "function": {"name": "finance_get_dividends_history", "description": "Return dividend history of a ticker.", "parameters": {"type": "object", "properties": {"ticker_name": {"type": "string", "description": "The ticker name of the stock interested."}}, "required": ["ticker_name"]}}}
Use the function 'finance_get_market_capitalization' to 'Return the market capitalization of a ticker.': {"type": "function", "function": {"name": "finance_get_market_capitalization", "description": "Return the market capitalization of a ticker.", "parameters": {"type": "object", "properties": {"ticker_name": {"type": "string", "description": "The ticker name of the stock interested."}}, "required": ["ticker_name"]}}}
Use the function 'finance_get_eps' to 'Return earnings per share of a ticker.': {"type": "function", "function": {"name": "finance_get_eps", "description": "Return earnings per share of a ticker.", "parameters": {"type": "object", "properties": {"ticker_name": {"type": "string", "description": "The ticker name of the stock interested."}}, "required": ["ticker_name"]}}}
Use the function 'finance_get_pe_ratio' to 'Return price-to-earnings ratio of a ticker.': {"type": "function", "function": {"name": "finance_get_pe_ratio", "description": "Return price-to-earnings ratio of a ticker.", "parameters": {"type": "object", "properties": {"ticker_name": {"type": "string", "description": "The ticker name of the stock interested."}}, "required": ["ticker_name"]}}}
Use the function 'finance_get_info' to 'Return rough meta data of a ticker.': {"type": "function", "function": {"name": "finance_get_info", "description": "Return rough meta data of a ticker.", "parameters": {"type": "object", "properties": {"ticker_name": {"type": "string", "description": "The ticker name of the stock interested."}}, "required": ["ticker_name"]}}}
Use the function 'music_search_artist_entity_by_name' to 'Search for music artists by name.': {"type": "function", "function": {"name": "music_search_artist_entity_by_name", "description": "Search for music artists by name.", "parameters": {"type": "object", "properties": {"artist_name": {"type": "string", "description": "The name of the artist interested."}}, "required": ["artist_name"]}}}
Use the function 'music_search_song_entity_by_name' to 'Search for songs by name.': {"type": "function", "function": {"name": "music_search_song_entity_by_name", "description": "Search for songs by name.", "parameters": {"type": "object", "properties": {"song_name": {"type": "string", "description": "The name of the song interested."}}, "required": ["song_name"]}}}
Use the function 'music_get_billboard_attributes' to 'Get attributes of a song from Billboard rankings.': {"type": "function", "function": {"name": "music_get_billboard_attributes", "description": "Get attributes of a song from Billboard rankings.", "parameters": {"type": "object", "properties": {"date": {"type": "string", "description": "The date you are interested."}, "attribute": {"type": "string", "description": "The attribute you are interested."}, "song_name": {"type": "string", "description": "The song you are interested."}}, "required": ["date", "attribute", "song_name"]}}}
Use the function 'music_get_billboard_rank_date' to 'Get Billboard ranking for a specific rank and date.': {"type": "function", "function": {"name": "music_get_billboard_rank_date", "description": "Get Billboard ranking for a specific rank and date.", "parameters": {"type": "object", "properties": {"rank": {"type": "integer", "description": "The rank you are interested."}, "date": {"type": "string", "description": "The date you are interested."}}, "required": ["rank"]}}}
Use the function 'music_grammy_get_best_album_by_year' to 'Get the Grammy Album of the Year for a specific year.': {"type": "function", "function": {"name": "music_grammy_get_best_album_by_year", "description": "Get the Grammy Album of the Year for a specific year.", "parameters": {"type": "object", "properties": {"year": {"type": "integer", "description": "The year interested."}}, "required": ["year"]}}}
Use the function 'music_grammy_get_all_awarded_artists' to 'Get all artists awarded the Grammy Best New Artist.': {"type": "function", "function": {"name": "music_grammy_get_all_awarded_artists", "description": "Get all artists awarded the Grammy Best New Artist.", "parameters": {"type": "object", "properties": {}, "required": []}}}
Use the function 'music_grammy_get_award_count_by_song' to 'Get the total Grammy awards won by a song.': {"type": "function", "function": {"name": "music_grammy_get_award_count_by_song", "description": "Get the total Grammy awards won by a song.", "parameters": {"type": "object", "properties": {"song_name": {"type": "string", "description": "The song interested."}}, "required": ["song_name"]}}}
Use the function 'music_grammy_get_best_song_by_year' to 'Get the Grammy Song of the Year for a specific year.': {"type": "function", "function": {"name": "music_grammy_get_best_song_by_year", "description": "Get the Grammy Song of the Year for a specific year.", "parameters": {"type": "object", "properties": {"year": {"type": "integer", "description": "The year interested."}}, "required": ["year"]}}}
Use the function 'music_grammy_get_award_date_by_artist' to 'Get

the years an artist won a Grammy award.': {"type": "function", "function": {"name": "music_grammy_get_award_date_by_artist", "description": "Get the years an artist won a Grammy award.", "parameters": {"type": "object", "properties": {"artist_name": {"type": "string", "description": "The artist interested."}}, "required": ["artist_name"]}}}
Use the function 'music_grammy_get_best_album_by_year' to 'Get the Grammy Album of the Year for a specific year.': {"type": "function", "function": {"name": "music_grammy_get_best_album_by_year", "description": "Get the Grammy Album of the Year for a specific year.", "parameters": {"type": "object", "properties": {"year": {"type": "integer", "description": "The year interested."}}, "required": ["year"]}}}
Use the function 'music_grammy_get_all_awarded_artists' to 'Get all artists awarded the Grammy Best New Artist.': {"type": "function", "function": {"name": "music_grammy_get_all_awarded_artists", "description": "Get all artists awarded the Grammy Best New Artist.", "parameters": {"type": "object", "properties": {}, "required": []}}}
Use the function 'music_get_artist_birth_place' to 'Return the birth place country code (2-digit) for the input artist.': {"type": "function", "function": {"name": "music_get_artist_birth_place", "description": "Return the birth place country code (2-digit) for the input artist.", "parameters": {"type": "object", "properties": {"artist_name": {"type": "string", "description": "The name of the artist interested."}}, "required": ["artist_name"]}}}
Use the function 'music_get_artist_birth_date' to 'Return the birth date of the artist.': {"type": "function", "function": {"name": "music_get_artist_birth_date", "description": "Return the birth date of the artist.", "parameters": {"type": "object", "properties": {"artist_name": {"type": "string", "description": "The name of the artist interested."}}, "required": ["artist_name"]}}}
Use the function 'music_get_members' to 'Return the member list of a band / person.': {"type": "function", "function": {"name": "music_get_members", "description": "Return the member list of a band / person.", "parameters": {"type": "object", "properties": {"band_name": {"type": "string", "description": "The name of the band / person interested."}}, "required": ["band_name"]}}}
Use the function 'music_get_lifespan' to 'Return the lifespan of the artist.': {"type": "function", "function": {"name": "music_get_lifespan", "description": "Return the lifespan of the artist.", "parameters": {"type": "object", "properties": {"artist_name": {"type": "string", "description": "The name of the artist interested."}}, "required": ["artist_name"]}}}
Use the function 'music_get_song_author' to 'Get the author of a song.': {"type": "function", "function": {"name": "music_get_song_author", "description": "Get the author of a song.", "parameters": {"type": "object", "properties": {"song_name": {"type": "string", "description": "The name of the song interested."}}, "required": ["song_name"]}}}
Use the function 'music_get_song_release_country' to 'Get the release country of a song.': {"type": "function", "function": {"name": "music_get_song_release_country", "description": "Get the release country of a song.", "parameters": {"type": "object", "properties": {"song_name": {"type": "string", "description": "The name of the song interested."}}, "required": ["song_name"]}}}
Use the function 'music_get_song_release_date' to 'Get the release date of a song.': {"type": "function", "function": {"name": "music_get_song_release_date", "description": "Get the release date of a song.", "parameters": {"type": "object", "properties": {"song_name": {"type": "string", "description": "The name of the song interested."}}, "required": ["song_name"]}}}
Use the function 'music_get_artist_all_works' to 'Return the list of all works of a certain artist.': {"type": "function", "function": {"name": "music_get_artist_all_works", "description": "Return the list of all works of a certain artist.", "parameters": {"type": "object", "properties": {"artist_name": {"type": "string", "description": "The name of the artist interested."}}, "required": ["artist_name"]}}}
Use the function 'sports_soccer_get_games_on_date' to 'Get soccer games on a specific date. Result includes game attributes such as date, time, GF: GF: Goals For - the number of goals scored by the team in question during the match, GA: Goals Against - the number of goals conceded by the team during the match, xG: Expected Goals - a statistical measure that estimates the number of goals a team should have scored based on the quality of chances they created, xGA: Expected Goals Against - a measure estimating the number of goals a team should have conceded based on the quality of chances allowed to the opponent, Poss: Possession - the percentage of the match time during which the team had possession of the ball.': {"type": "function", "function": {"name": "sports_soccer_get_games_on_date", "description": "Get soccer games on a specific date. Result includes game attributes such as date, time, GF: GF: Goals For - the number of goals scored by the team in question during the match, GA: Goals Against - the number of goals conceded by the team during the match, xG: Expected Goals - a statistical measure that estimates the number of goals a team should have scored based on the quality of chances they created, xGA: Expected Goals Against - a measure estimating the number of goals a team should have conceded based on the quality of chances allowed to the opponent, Poss: Possession - the percentage of the match time during which the team had possession of the ball.", "parameters": {"type": "object", "properties": {"date": {"type": "string", "description": "The date interested."}, "team_name": {"type": "string", "description": "The team interested."}}, "required": ["date", "team_name"]}}}
Use the function 'sports_nba_get_games_on_date' to 'Get NBA games on a specific date. Result includes game attributes such as team_name_home: The full name of the home team, wl_home: The outcome of the game for the home team, pts_home: The total number of points scored by the home team.': {"type": "function", "function": {"name": "sports_nba_get_games_on_date", "description": "Get NBA games on a specific date. Result includes game attributes such as team_name_home: The full name of the home team, wl_home: The outcome of the game for the home team, pts_home: The total number of points scored by the home team.", "parameters": {"type": "object", "properties": {"date": {"type": "string", "description": "The date interested."}, "team_name": {"type": "string", "description": "The team interested."}}, "required": ["date", "team_name"]}}}
Use the function 'sports_nba_get_play_by_play_data_by_game_ids' to 'Get NBA play by play data for a set of game ids. Result includes play-by-play event time, description, player etc.': {"type": "function", "function": {"name": "sports_nba_get_play_by_play_data_by_game_ids", "description": "Get NBA play by play data for a set of game ids. Result includes play-by-play event time, description, player etc.", "parameters": {"type": "object", "properties": {"game_ids": {"type": "list", "description": "List of game ids."}}, "required": ["game_ids"]}}}

If you choose to call a function ONLY reply in the following format with no prefix or suffix, if you think you need to chain multiple function calls, reply the function that you want to call at the current round and a special token <CONTINUE>:
Question: What is the price of ABCD currently? Your answer: <function=finance_get_detailed_price_history>{"ticker_name": "ABCD"}</function>
Question: Can you tell me who directed the movie ABCD? Your answer: <function=movie_get_movie_info>{"movie_name": "ABCD"}</function>
Question: Who are the release date of the last song of band ABCD? Your answer: <function=music_get_artist_all_works>{"band_name": "ABCD"}</function>; <CONTINUE>
Question: On 2022-11-11, how many points did Pacers put up in their game? Your answer: <function=sports_nba_get_games_on_date>{"date": "2022-11-11", "team_name": "Indiana Pacers"}</function>
Question: What is the population of China? Your answer: <function=open_get_entity>{"entity": "China"}</function>
Question: What is the movie that has id 123456? Your answer: <function=movie_get_movie_info>{"movie_id": 123456}</function>
Reminder:
- Function calls MUST follow the specified format, start with <function= and end with </function>
- Put the entire function call reply on one line
- If there is no function call available, answer I don't know.'
- Reply <CONTINUE> after the function call you choose, seperate them with ;.
The current query time is: <*Query Time*>
<*Question*>

Tool-use Prompts of Head2Tail

You need to write SPARQL query to access DBPedia to get the answer to the QUESTION. Here is the instruction: 1. Please select the main entity from the QUESTION; 2. Select the proper predicate from the given PREDICATE LIST; 3. Based on the main entity, predicate selected, write a SPARQL query to answer the QUESTION. Enclose the main entity selected with <>; Enclose the predicate selected with []. Only respond the main entity and predicate!!!"
QUESTION: <*Question*>
PREDICATE LIST: <*List of predicate*>

I.5 Auto-evaluation Prompts

Auto-evaluation Prompts

Auto-evaluation of CRAG

Assume you are a human expert in grading predictions given by a model. You are given a question and a model prediction. Judge if the prediction matches the ground truth answer by following these steps: 1: Take it as granted that the Ground Truth is always correct.
2: If the Prediction indicates it is not sure about the answer, "score" should be "0"; otherwise, go the next step.
3: If the Prediction exactly matches the Ground Truth, "score" is 1.
4: If the Prediction does not exactly match the Ground Truth, go through the following steps and likely give a score as 0.
5: If the Ground Truth is a number, "score" is 1 if and only if the Prediction gives a number that almost exactly matches the ground truth.
6: If the Prediction is self-contradictory, "score" must be 0.
7: If the prediction is not answering the question, "score" must be 0.
8: If the prediction is a concise and correct summary of the ground truth, "score" is 1.
9: If ground truth contains a set of items, prediction must contain exactly same items for the score to be 1.
10: Otherwise, "score" is 0.
Output a JSON blob with an "explanation" field explaining your answer as short as possible and an "score" field with value 1 or 0. You should make the judgment based on provided examples. Examples: Question: "which company has higher eps, btu or cma?" Ground Truth: "cma" Prediction: "it is not possible to determine which company has a higher eps." Output: {"score": 0, "explanation": "The prediction is not sure about the answer."}
Question: "who is taller, a or b?" Ground Truth: "a" Prediction: "The answer is a. a is of height 1.75 m and b is of height 1.82 m. So b is taller." Output: {"score": 0, "explanation": "The prediction is self-contradictory."}
Question: "who among patricia place and lana clarkson is younger?" Ground Truth: "lana clarkson" Prediction: "lana clarkson (born april 5, 1962) is younger than patricia clarkson (born december 29, 1959)." Output: {"score": 1, "explanation": "The prediction gives the correct answer."}
Question: "did lakers win more games than warriors in 2024-4?" Ground Truth: "yes" Prediction: "yes, lakers won more games than warriors in 2023-2024 season." Output: {"score": 0, "explanation": "The prediction is not answering the question."}
Question: "what is the height of the building where citigroup is headquartered?" Ground Truth: "151 m" Prediction: "the height is 915 feet (279 m)." Output: {"score": 0, "explanation": "The prediction, 151 m, does not match the ground truth, 279 m."}
Question: "who were the last 3 prime ministers of uk?" Ground Truth: "rishi sunak, liz truss, and boris johnson" Prediction: "1. liz truss, 2. boris johnson, 3.theresa may" Output: {"score": 0, "explanation": "The prediction does not contain item, rishi sunak, that is in the grount truth."}
Question: "who authored the taming of the shrew?" Ground Truth: "william shakespeare" Prediction: "william shakespeare" Output: {"score": 1, "explanation": "The prediction exactly matches the ground truth."}
Question: "who authored The Taming of the Shrew?" Ground Truth: "william shakespeare" Prediction: "w shakespeare" Output: {"score": 1, "explanation": "The prediction matches the ground truth as w is the abbreviation of william."}
Question: "what is the state bird of california?" Ground Truth: "california quail" Prediction: "california valley quail" Output: {"score": 1, "explanation": "The prediction matches the ground truth as california quail is also known as california valley quail."}
Question: "what country is maharashtra metro rail corporation limited located in?" Ground truth: "india" Prediction: "indianapolis" Output: {"score": 0, "explanation": "The prediction doesn't match the ground truth answers."}
Question: "what was wolves's record in their last game in eng-premier league?" Ground Truth: "w" Prediction: "wolves's record in their last game in the eng-premier League was a 2-1 win over fulham." Output: {"score": 1, "explanation": "The prediction correctly answers the result is a win."}

Question: "how many seconds is 3 minutes 15 seconds?" Ground Truth: "195 seconds" Prediction: "3 minutes 15 seconds is 195 seconds." Output: {"score": 1, "explanation": "The prediction matches ground truth as it correctly states the number of seconds asked in the question."}
Question: "on which days did am distribute dividends in the last year?" Ground Truth: "2023-01-24, 2023-04-25, 2023-07-25, 2023-10-24" Prediction: "the company distributed dividends on october 24, 2023 and january 24, 2023." Output: {"score": 0, "explanation": "The prediction partially matches the ground truth, but it is incomplete."}
Question: "on which days did xxx distribute dividends in the last year?" Ground Truth: "2023-01-13, 2023-03-25, 2023-11-21" Prediction: "xxx distributed dividends on 1. 2023-01-13, 2. 2023-03-25, 3. 2023-10-21." Output: {"score": 0, "explanation": "The prediction contains an item, 2023-10-21, which is not in ground truth."}
Question: "who are current band members of maroon 5?" Ground Truth: "adam Levine, jesse carmichael, james valentine, matt flynn, pj morton, sam farrar" Prediction: "they are jesse carmichael, matt flynn, adam levine, pj morton, sam farrar, and james valentine." Output: {"score": 1, "explanation": "The prediction exactly matches the ground truth."}
Question: "which movies comprise the matrix franchise?" Ground Truth: "the matrix, the matrix reloaded, the matrix revolutions, the matrix resurrections" Prediction: "the matrix, the matrix reloaded, the matrix revolutions, the animatrix, and the matrix resurrections." Output: {"score": 0, "explanation": "The prediction covers more items than what are given by the ground truth."}
Question: "how deep is the deepest lake of new york?" Ground Truth: "618 ft" Prediction: "the deepest lake in new york is seneca lake, with a depth of 618.23 feet." Output: {"score": 1, "explanation": "The prediction exactly matches the number in ground truth after rounding."}
Question: "what is the closing price of meta yesterday?" Ground Truth: "$310.17" Prediction: "310.2" Output: {"score": 1, "explanation": "The prediction exactly matches the number in ground truth after rounding."}
Question: "what is the current market cap of appl?" Ground Truth: "2.81 trillion" Prediction: "2.667 trillion" Output: {"score": 0, "explanation": "The prediction does not match the number in ground truth."}
Question: "what is the current pe ratio of appl?" Ground Truth: "28.3" Prediction: "the current pe ratio of apple is 26.66" Output: {"score": 0, "explanation": "The prediction does not match the number in ground truth."}
Question: "how much is tesla's stock price down from its all-time high?" Ground Truth: "$221.83" Prediction: "209.52" Output: {"score": 0, "explanation": "The prediction does not match the number in ground truth."}
Question: "what is the length of amazon river?" Ground Truth: "over 4000 miles" Prediction: "the length of amazon river is 4,000 miles" Output: {"score": 0, "explanation": "The prediction does not say Amazon River is longer than 4000 miles."}
Question: "how many copies x were sold?" Ground Truth: "2 million." Prediction: "it is over 2 million" Output: {"score": 0, "explanation": "The prediction does not match the ground truth."}
Question: "what is the population of country x?" Ground Truth: "3,576,873" Prediction: "the population of country x is 3.3 million." Output: {"score": 0, "explanation": "The prediction, 3.3 M, does not match the number, 3.6 M, in ground truth."}
Question: "what is the current market value of stock x?" Ground Truth: "$2,237,578,268" Prediction: "$2.16 billion." Output: {"score": 0, "explanation": "The prediction, 2.16 B, does not match the number, 2.2 B, in ground truth."}
Question: "how old is michael jordan?" Ground Truth: "61" Prediction: "mj was born on February 17, 1963." Output: {"score": 0, "explanation": "The prediction does not explicitly say the age."}
Question: "how long did amy teach as a teacher?" Ground Truth: "24 years and 255 days" Prediction: "25 years." Output: {"score": 0, "explanation": "The prediction does not exactly match the ground truth."}
Question: "at what age did usher perform in the half time show of superbowl?" Ground Truth: "45 years old" Prediction: "usher has not performed at a super bowl halftime show." Output: {"score": 0, "explanation": "The prediction does not match the ground truth."}
Question: "what year did olivia rodrigo win the grammys best new artist award?" Ground Truth: "2022" Prediction: "sure! here's an answer in three words or fewer: 2022 grammy awards." Output: {"score": 1, "explanation": "The prediction answers the correct number of the year."}
Question: "how many times have zendaya and tom holland starred in a film together?" Ground Truth: "3" Prediction: "they have starred in 3 films together: spider-man: homecoming, spider-man: far from home, and spider-man: no way home." Output: {"score": 1, "explanation": "The prediction gives the right number and it is reasonable."}
Question: "who is younger, timothee chalamet or tom holland?" Ground Truth: "tom holland" Prediction: "timothée chalamet is younger than tom holland." Output: {"score": 0, "explanation": "The prediction does not match the ground truth."}
Question: "who had more number one hits on the us billboard, a or b?" Ground Truth: "a had more number one hits on the us billboard than b, with 20 number one hits compared to b's 15." Prediction: "a" Output: {"score": 1, "explanation": "The prediction is a concise and correct summary of the ground truth."}
Question: "what is xxx's birthdate?" Ground Truth: "1996-01-01." Prediction: "02/01/1996" Output: {"score": 0, "explanation": "The prediction does not match the ground truth."}
Question: "what was the worldwide box office haul for movie x?" Ground Truth: "101756123." Prediction: "102 million" Output: {"score": 1, "explanation": "The prediction exactly matches the number in ground truth after rounding."}
Question: "how much has spotify's user base increased by since 2020 in na?" Ground Truth: "spotify's user base increased by 34 million since 2020." Prediction: "spotify's north american user base increased from 36 million in 2020 to 85 million by 2021" Output: {"score": 0, "explanation": "The prediction is not answering the question as it only gives the increase from 2020 to 2021."}
Question: *<Question>* Ground truth: *<Ground Truth>* Prediction: *<Answer>*

Auto-evaluation of Head2Tail

The task is provided a QUESTION with GROUND_TRUTH answer, evaluate whether my ANSWER is correct, answer briefly with Yes/No. You will first see some [EXAMPLE]s on this task and then you will complete the [TASK].
Here are some examples:
[EXAMPLE]: QUESTION: Bangladesh Nationalist Party is the member of which international organization? GROUND_TRUTH: Asia Pacific Democrat Union or Cen-

trist Democrat International ANSWER: according to the wikipedia page, bangladesh nationalist party is a member of the centrist democrat international. Your answer: Yes
[EXAMPLE]: QUESTION: What/who influenced Charles Fourier? GROUND_TRUTH: Nicolas-Edme Rétif ANSWER: bob black. Your answer: No
[EXAMPLE]: QUESTION: What patrol aircraft is used by the South African Air Force? GROUND_TRUTH: C-47 Skytrain ANSWER: according to the wikipedia infobox, the patrol aircraft used by the south african air force is the c-47tp. Your answer: Yes
[EXAMPLE]: QUESTION: Which automobile team had the fastest driver during the 1960 Indianapolis 500? GROUND_TRUTH: A.J. Watson ANSWER: ken-paul Your answer: No
[EXAMPLE]: QUESTION: What party was split from Communist Refoundation Party? GROUND_TRUTH: Italian Communist Party ANSWER: according to the wikipedia infobox, the italian communist party (pci) was split from to form the communist refoundation party (prc) in 1991. Your answer: Yes
[EXAMPLE]: QUESTION: Which company owns TV Land? GROUND_TRUTH: Paramount Media Networks ANSWER: paramount global. Your answer: No
[EXAMPLE]: QUESTION: What is the stadium where BSG Chemie Leipzig (1950)'s home matches are held? GROUND_TRUTH: Alfred-Kunze-Sportpark or Georg-Schwarz-Sportpark ANSWER: alfred-kunze-sportpark (also known as georg-schwarz-sportpark) Your answer: Yes
[EXAMPLE]: QUESTION: What language is spoken in Evening (magazine)? GROUND_TRUTH: Japanese language ANSWER: english Your answer: No
[EXAMPLE]: QUESTION: What is the ending theme of My Papa Pi? GROUND_TRUTH: Pia Wurtzbach ANSWER: the ending theme of my papa pi is "my papa pi" by piolo pascual, pia wurtzbach, and pepe herrera. Your answer: Yes
[EXAMPLE]: QUESTION: What is the record label for Cogumelo Records? GROUND_TRUTH: Relapse Records ANSWER: cogumelo records. Your answer: No
[EXAMPLE]: QUESTION: What is the legislative body in Albanian Kingdom (1928–1939)? GROUND_TRUTH: Parliament of Albania ANSWER: according to the wikipedia infobox and summary, the legislative body in the albanian kingdom (1928–1939) is the constitutional assembly. Your answer: Yes
[EXAMPLE]: QUESTION: Jim Pearson was born in which place? GROUND_TRUTH: Falkirk ANSWER: chatham, ontario, canada. Your answer: No
[EXAMPLE]: QUESTION: The predecessor of Cirilo Almario is? GROUND_TRUTH: Manuel del Rosario ANSWER: manuel p. del rosario, d.d. Your answer: Yes
[EXAMPLE]: QUESTION: What is the format of The Wedge (Australian TV series)? GROUND_TRUTH: Stereophonic sound ANSWER: the format of the wedge (australian tv series) is a sketch show. Your answer: No
[EXAMPLE]: QUESTION: What is the mouth of Montreal River (Timiskaming District)? GROUND_TRUTH: Timiskaming, Unorganized, West Part, Ontario ANSWER: according to the wikipedia infobox and summary, the mouth of the montreal river (timiskaming district) is lake timiskaming on the ottawa river. Your answer: Yes
[EXAMPLE]: QUESTION: Who developed Flappy? GROUND_TRUTH: DB-SOFT ANSWER: flappy bird was developed by .gears, which is a game development company founded by dong nguyen. Your answer: No
[EXAMPLE]: QUESTION: What significant design was created by Joseph Berlin? GROUND_TRUTH: Cinema of Israel ANSWER: mograbi cinema, tel aviv. Your answer: Yes
[EXAMPLE]: QUESTION: What is Cinematic soul derived from? GROUND_TRUTH: Disco ANSWER: soul music, psychedelic soul, orchestral music, and film score. Your answer: No
[EXAMPLE]: QUESTION: What patrol aircraft is used by the VPB-127? GROUND_TRUTH: Lockheed Ventura or PV-1 ANSWER: pv-1 Your answer: Yes
[EXAMPLE]: QUESTION: Which automobile team had the fastest driver during the 1953 Curtis Trophy? GROUND_TRUTH: Cooper Car Company ANSWER: cooper-bristol. Your answer: No
[TASK]: QUESTION: <*Question*> GROUND_TRUTH: <*Ground Truth*> ANSWER: <*Answer*>
Your answer is?