KR2: Knowledge-Enhanced Retrieval-Augmented Generation for Knowledge Graph Question Answering*

Duygu Sezen Islakoglu[1,*], Melisachew Wudage Chekol[1] and Yannis Velegrakis[1,2]

[1]Utrecht University, The Netherlands
[2]University of Trento, Italy

Abstract

Large language models (LLMs) are widely used for knowledge-intensive tasks such as question answering, but they often generate inaccurate or outdated responses due to their static training data and tendency to hallucinate. Retrieval-Augmented Generation (RAG) mitigates this issue by enriching prompts with external knowledge sources. Recently, knowledge graphs (KGs) have been adopted in RAG pipelines to support more structured and factually grounded retrieval. This integration is especially promising for knowledge graph question answering (KGQA), as LLMs offer strong language understanding and some reasoning capabilities. However, standard text encoder-based retrievers in these pipelines ignore the structural properties and type hierarchies inherent to KGs. In this work, we introduce KR2, a novel framework that improves retrieval in KGQA by leveraging the structure and entity types from the knowledge graph, specifically the instanceOf and subclassOf relations. It introduces a novel way to train a retriever by exploiting the structural nature of a knowledge graph and requires no annotated (question-answer) or (query-passage) pairs, unlike existing approaches. Our experiments across multiple datasets and pretrained language models demonstrate that KR2 achieves competitive performance compared to text encoder–based retrievers. In addition, type injection further improves retrieval accuracy and downstream LLM output quality.

Keywords

Knowledge Graph Question Answering, Retrieval-Augmented Generation, Large Language Models

1. Introduction

Large language models (LLMs) have become integral components of various pipelines and workflows, particularly for knowledge-intensive tasks across diverse domains. However, the answers generated by LLMs are not always reliable as they suffer from hallucinations [1, 2]. This issue arises due to several factors. Firstly, the pre-training data may lack domain-specific knowledge or sufficient context for long-tail entities. Second, knowledge evolves, and new information continuously emerges. LLMs do not have up-to-date knowledge as they are trained with data that only extends up to a certain point in time. Therefore, they cannot incorporate new and evolving knowledge without retraining.

In order to address these shortcomings and inject new knowledge into the LLM's parameters, various approaches have been proposed. One approach is model editing, which provides a way to incorporate new information; however, it is not scalable [3] and might have undesirable effects on LLM's general abilities [4]. Another approach involves fine-tuning the language models. Similar to model editing, this approach can be costly and may negatively impact the model's performance on other tasks.

An alternative solution is Retrieval-Augmented Generation (RAG) [5]. RAG connects LLMs to a knowledge source, such as a database or a collection of documents. When a user poses a question to an LLM, a retriever identifies relevant information from the given knowledge source and augments the LLM's prompt. This approach improves the model's performance by providing up-to-date and contextually relevant information. Moreover, RAG increases interpretability as the retrieved knowledge can be explicitly traced [3]. RAG approaches also ensure privacy, as the private or sensitive knowledge

RAGE-KG 2026: The Third International Workshop on Retrieval-Augmented Generation Enabled by Knowledge Graphs, co-located with ISWC 2026, October 25–29, 2026, Bari, Italy
*Corresponding author.
d.s.islakoglu@uu.nl (D. S. Islakoglu); m.w.chekol@uu.nl (M. W. Chekol); i.velegrakis@uu.nl (Y. Velegrakis)
0000-0002-9441-6458 (D. S. Islakoglu); 0000-0002-5286-8587 (M. W. Chekol); 0000-0001-6332-0296 (Y. Velegrakis)

© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

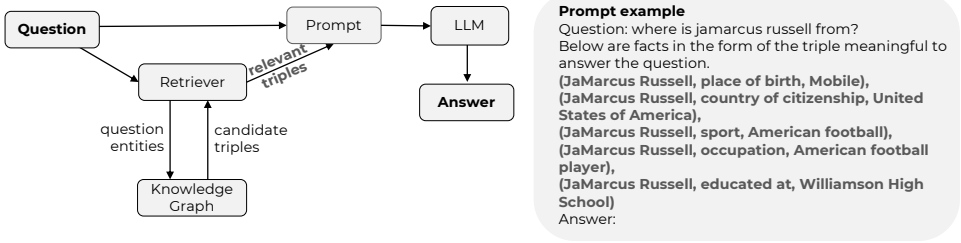

Figure 1: Left: An overview of retrieval-augmented generation based KGQA. Right: A knowledge-enhanced KGQA prompt template, adapted from [7].

will not be embedded into the model weights [6]. Finally, RAG approaches enable more efficient and cost-effective deployments as smaller language models can have strong performance with context-rich knowledge at inference time [7].

Recent studies utilize knowledge graphs (KGs) as knowledge sources for retrieval-augmented generation [8] to improve question-answering performance and enhance LLMs with high-quality knowledge. This integration, as illustrated in Figure 1, has shown promising results in tasks such as knowledge graph question answering (KGQA). To answer questions over a KG, it is possible to query the KG directly. However, interacting with knowledge graphs is challenging for non-experts and requires SPARQL knowledge. Some Text2SPARQL approaches translate the natural language query to SPARQL; however, these approaches are still error-prone. Therefore, a RAG-based KGQA framework offers the best of both worlds: leveraging KGs to enhance the factual accuracy and coverage of LLMs, while also using LLMs' language understanding capabilities to answer queries over a knowledge graph.

In RAG-based KGQA, incorporating all triples from a KG into the prompt is impractical due to the typically large size of KGs and the token length limitations of LLMs. Furthermore, incorporating irrelevant context can introduce noise and negatively impact the model's performance [9]. This highlights the need for a retriever that can effectively select only the most relevant triples from the KG. However, retrieving the most relevant information from a knowledge graph given a natural language question remains a challenging task. While text-based encoder models are commonly used for this purpose, they typically do not incorporate the entity types or the structural properties of the knowledge graph, thus, they do not fully exploit its rich semantics. In prior work, entity type information has been leveraged in various contexts and shown to enhance performance across multiple tasks. For instance, type embeddings are incorporated for representations of the entities in entity-oriented search [10] and for entity linking [11]. Another line of work performs question classification into target types to get a type constraint on the answer [12]. Another study investigates how entity type information can be utilized in entity retrieval and reports that type-based components improve the performance when combined with term-based retrieval [13] (e.g. [14]). More recently, domain and range constraints have been leveraged for link prediction [21]. Although type information in KGs is used in a wide range of applications, its integration into RAG systems remains underexplored.

In this work, we propose to integrate structural and hierarchical information from knowledge graphs into the retrieval process **to strengthen specifically smaller language models**, such as those with 3 billion (3B) parameters. We implement this in a knowledge-enhanced KGQA framework, referred to as KR2. Unlike prior approaches that rely on multiple LLM calls, KR2 requires a single LLM call. We make two important contributions that augment different components of RAG systems: (1) a novel method to fine-tune a base pre-trained language model as a retriever directly on the knowledge graph structure, without requiring any (question, answer) or (query, passage) pairs, and (2) a type enhancement mechanism that incorporates entity type information into both the question and the triples. To the best of our knowledge, this is the first study to use entity types from a KG for RAG-based KGQA. Experiments across multiple datasets and language models show that KR2 achieves competitive performance compared to text-based retrievers, and type injection further enhances retrieval accuracy and the quality of LLM-generated outputs.

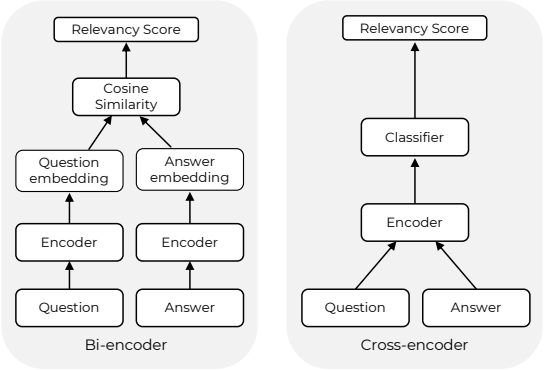

Figure 2: Bi-Encoder and Cross-Encoder Architectures. *Encoder* corresponds to a pre-trained encoder-only language model such as BERT [17].

2. Preliminaries

Knowledge Graphs. A knowledge graph (KG) is a directed graph $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{F})$ where $\mathcal{E}$ and $\mathcal{R}$ are sets of entities and relations, and $\mathcal{F}$ represents the set of facts (triples) in the format $\langle$subject s, relation r, object $o\rangle$ where $s, o \in \mathcal{E}$, $r \in \mathcal{R}$, and $\langle s, r, o\rangle \in \mathcal{F}$. For an entity $e \in \mathcal{E}$, the set of triples in which it appears as a subject is denoted by $\mathcal{N}_e$= {$\langle$e, r, $o\rangle$}.

Knowledge Graph Question-Answering (KGQA). Given a knowledge graph $\mathcal{G}$ and a question in a natural language q, knowledge graph question-answering aims to find the answer set that correctly answers q using the information in $\mathcal{G}$. In general, answers to KGQA questions may include entities, literals, or other values. In this work, we focus on entity-answer questions, where each answer e_a is an entity in $\mathcal{E}$. Note that a question may have multiple correct answers. Each question has one or multiple question entities $\{e_{q_i}\} \in \mathcal{E}$, and the KGQA datasets also provide them along with (question, answer) pairs.

RAG-based KGQA. The task of knowledge graph question-answering using LLMs through retrieval-augmented generation (RAG-based KGQA) is illustrated in Figure 1. Given a knowledge graph $\mathcal{G}$, a natural language question q and its corresponding question entities e_{q_i}, the set of candidate triples is $\bigcup_{i=1}^{n}\{\mathcal{N}_{e_{q_i}}\}$. From the set of candidate triples, a retriever Ret_k scores each triple to output k most relevant triples to the question. A prompt is constructed by combining the question q and the retrieved triples. Then, the prompt is fed into an LLM model to generate an answer text a. More compactly, a = LLM($prompt$) where $prompt = q + Ret_k(q, \bigcup_{i=1}^{n}\{\mathcal{N}_{e_{q_i}}\})$. To evaluate, we compare this answer a with each ground-truth answer in $\{e_{a_j}\}$.

In order to construct a prompt for a given question, the relevant triples need to be retrieved from the given KG. There are different approaches to retrieving k relevant triples from a KG $\mathcal{G}$ for a given question q. These include: (i) *subgraph-based retrieval*, where starting from the question entity e_q, the graph is traversed to retrieve k triples that are within h-hops of e_q [15]; (ii) *text encoder-based methods*, such as bi-encoder and cross-encoder architectures [16, 7], that gives a relevancy score between q and a KG triple based on their semantic similarity. Subgraph-based retrieval methods often struggle with incomplete or sparsely connected KGs. However, text encoder-based methods, which are the focus of this work, are capable of scoring any (question, triple) pairs, not only direct neighbors of e_q.

Bi-Encoder and Cross-Encoder Architectures as Triple Retrievers. Bi-encoder and cross-encoder are two different text encoder-based retriever architectures in information retrieval [18]. Although both take a question and some text (in our case, a triple) as input, and output a relevancy score, their architectures differ. Bi-encoder architecture embeds the question and a triple into a vector space independently, using the same encoder that is an encoder-only language model, such as BERT [17]. Then a similarity function, such as cosine similarity, is used to output a similarity score between these two embeddings. The bi-encoder approach allows for efficient retrieval because triple embeddings can be precomputed and indexed in advance. However, since the query embedding and triple embeddings

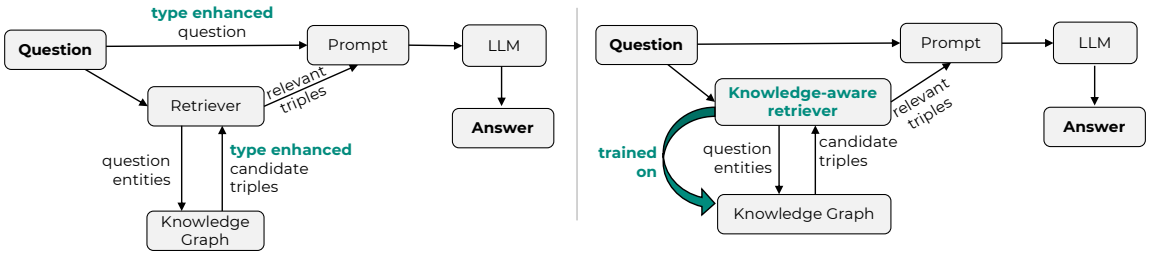

Figure 3: KR2 overview: two knowledge enhancement approaches for RAG-based KGQA.

are independent, their interaction cannot be properly captured. Cross-encoders, on the other hand, combine the question and the triple as a single text and get a single embedding by an encoder-only language model. The relevancy score between the question and the triple is derived by a linear layer (classifier) that is applied to this embedding. The cross-encoder architecture is also known as *re-ranker*. We illustrate both architectures in Figure 2. In this paper, we utilize both architectures to retrieve relevant triples, and we introduce a way to adapt the knowledge graph facts for their training.

Entity Types. Many entities in a KG belong to classes that group individuals with common characteristics [19]. For example, the entity *Barack Obama* is an "instance of" the class *Politician*, while the class *Politician* is a "subclass of" *Human*. Class memberships and classes are typically organized using relations such as *subclassOf* and *instanceOf* relations, which are very common in general knowledge graphs such as Wikidata [20]. These relations are important as they enable abstraction and inference in KGs, e.g., if an entity is an instance of a subclass, it is also an instance of the superclass. In this work, we use entity types to enhance the question and triples in a RAG-based KGQA system.

3. KR2 Framework

We present the KR2 framework, which introduces two independent approaches to enhance RAG-based KGQA systems by leveraging a knowledge graph. Within KR2, the knowledge graph serves not only as an external source for retrieval-augmented generation but also enhances the retrieval process itself. Two approaches are illustrated in Figure 3. The first approach, shown on the left, enhances prompts with entity types that are available in the KG (Section 3.1). The second approach, shown on the right, trains a retriever on a KG by systematically transforming triples into training examples (Section 3.2).

3.1. Approach 1: Prompt Enhancement with Entity Types

In this section, we describe how our framework enhances prompts by incorporating entity types that are available in a knowledge graph. This approach can be applied to both bi-encoder and cross-encoder retriever architectures.

There are several reasons why we include entity types in our design. First, we observe that some triples that are retrieved by text-based encoders e.g. SentenceBERT [16] can be irrelevant, as those models lack the concept of the types and only utilize the semantical similarity. Second, mentioning the entity types can be useful for LLMs in reasoning and entity disambiguation, thus has the potential to improve LLM's answers. Lastly, when we ask a question, we often mention types in the question, e.g. "What language do people in Iceland speak?" (from WebQSP dataset [21, 22]). Therefore, injecting types into triples can support the retriever to catch the triple that has the correct answer entity, and improve the retriever metrics.

In order to enhance a prompt, our framework makes use of *subclassOf* and *instanceOf* relations to get entity types from a KG. More formally, to obtain the type of an entity e, we check *subclassOf* or *instanceOf* relations in $\mathcal{N}_e$. For example, if the KG contains the triple <reggae, subclassOf, music of Jamaica>, we assign "music of Jamaica" as the type of the entity "reggae".

We design different settings to inject types: either we enhance the question with a type of question entity, or we enhance the triples by attaching an entity type to the subject and object. Based on the settings, we identify six different prompt templates: Inst[Q+K], Inst[Q], Inst[K], SubC[Q+K], SubC[Q], and SubC[K]; where Inst denotes the relation *instanceOf*, SubC corresponds to *subclassOf* relation, Q refers to "question", and K refers to "knowledge" (triples to retrieve). We enhance only the question for [Q], only the triples for [K], and both for [Q+K] templates.

- For [Q] templates, we take one type for each question entity. Then those are added to the question with the following format: "[e_q] is a [type]" for Inst and "[e_q] is a subclass of [type]" for SubC. For instance, for the question "who played on the jeffersons?", the sentence **"the jeffersons is a television series."** is added to the question to enhance the prompt.
- For [K] templates, we add entity types to the subject and the object of each triple prior to retrieval, so it can alter the retrieved results. To do so, we employ the following format for both Inst and SubC: "[subject] ([type]) [relation] [object] ([type])". For instance, the triple <Google, owner of, Google News> becomes **<Google (business), owner of, Google News (news aggregation website>.**

Since entities can be associated with multiple types, enriching prompts with irrelevant types may degrade performance. To mitigate this, we employ a pre-trained bi-encoder[1] to select the most relevant type to the given question. Specifically, we encode the question and each candidate type independently with this encoder and compute their cosine similarity in the shared embedding space. The candidate type with the highest cosine similarity score is then selected.

3.2. Approach 2: KG-Aware Retriever

In this section, we detail the methodology for training a cross-encoder and a bi-encoder retriever using data from a knowledge graph. Training those models for RAG-based KGQA would require a dataset of the form (question, triple, relevance score). Since such a dataset is not publicly available, we introduce an approach that creates training data directly from the KG by leveraging its structural properties. To the best of our knowledge, this is the first work to create training data for text-encoders (cross-encoder and bi-encoder) exclusively from KG structure, without relying on any question–answer pairs (e.g., from KGQA datasets). The trained models, referred to as KG-aware cross-encoders and KG-aware bi-encoders, are then employed as a retriever to score and rank candidate triples in the context of RAG-based KGQA systems. This integration allows the KG to not only provide factual content to the generation process but also directly enhance the retrieval mechanism.

For each triple $\langle s, r, o \rangle$ in the KG, we first construct a positive training instance by treating the subject–relation pair (s, r) as the question and the object o as the correct answer. This yields a positive example: $((s, r), o, 1)$. To generate corresponding negative examples, we sample an entity o' such that the triple $\langle s, r, o' \rangle$ is not present in the KG. This forms a negative instance: $((s, r), o', 0)$. This enables the models to learn a notion of triple plausibility, wherein they assign higher scores to triples that are plausible and lower scores to implausible triples. Effectively, they operate analogously to a link prediction model.

Cross-encoder training involves fine-tuning a pre-trained encoder-only language model for a classification task, as shown in Figure 2. The objective of the cross-encoder is to learn a scoring function that assigns higher scores to positive question-answer pairs and lower scores to negative ones. To achieve this, the model is trained as a binary classifier with positive and negative triples, using a classification loss function. We fine-tune a base model distilroberta-base[2] [23] for 10 epochs with binary cross-entropy loss. For bi-encoder models, we fine-tune the same base model "distilroberta-base". We use online contrastive loss with a margin of 0.5, and we train these models for one epoch[3].

[1] https://huggingface.co/sentence-transformers/all-mpnet-base-v2

[2] This model is chosen as it serves as the default model for this task in the SentenceTransformers library [16].

[3] Corresponding to the default configuration of the library.

4. Experiments

We evaluate our approach on two datasets, namely Mintaka [24] and WebQSP [21, 22]. Mintaka is a knowledge graph question-answering dataset in which the questions and answers are annotated with Wikidata [20] entities. It consists of eight different types of complex questions and comprises 4000 test samples. WebQSP (WebQuestions Semantic Parses Dataset) is a question-answering dataset that consists of real user questions. The entities of the question-answer pairs are originally based on Freebase [25]. However, we use the Wikidata version from [7], where the Freebase entities are mapped to Wikidata entities by [26]. With this mapping, we get 1,466 test samples for the WebQSP dataset. Note that each of these datasets is associated with a different knowledge graph.

For the data and source code, please refer to our *Supplemental Material Statement.* We conduct experiments with the following language models: T0-3B [27] , Flan-T5-3B [28] and Llama-3.2-3B-Instruct[4] (see Section A.2 for the larger models (7B, 8B, 9B)). While T0-3B and Flan-T5-3B have encoder-decoder architecture, Llama-3.2-3B-Instruct is a decoder-only model. Although our framework is LLM-agnostic, meaning that it can work with any language model, we use 3B models to demonstrate how smaller models with fewer parameters can benefit from external knowledge without retraining.

The maximum input sequence length is set to 1024. The maximum output token length is 128 for 3B models, and 256 for larger models. We follow [7] for retrieval settings, so we include 1-hop triples from the question entities, use the same prompt template (Figure 1), and set the number of retrieved triples, k, to 10. As cross-encoder input for 1-hop triples, we omit the subject entity from the triples (i.e., we input only the relation and object) since the question already contains the subject entity, and repeating it would not be compatible with the cross-encoder training data format. For KR2-cross-encoder and KR2-bi-encoder training, we obtain 1.8M triples for the knowledge graph associated with the Mintaka dataset [24] and approximately 2.2M triples for the WebQSP [21, 22] knowledge graph. For each triple, we create 5 negatives. For both architectures, we use 80% of the triples as the training set.

4.1. Evaluation

Generation Metrics. In order to assess the quality of the answers generated by the selected LLMs, we compare the ground-truth entity with the LLM's answer on the test data to evaluate question-answering performance. To this end, we use three metrics by following [7, 29]: accuracy, exact match (EM), and F1. As noted in [7], F1 and EM largely penalize the LLM's long answers. Nevertheless, given their common usage in evaluation, we have included these metrics in our experiments. First, we normalize both the LLM answers and the ground-truth entities by lowercasing them and removing whitespace, punctuation, and articles ('a', 'an', 'the'). The accuracy is 1 if the LLM's answer includes one of the ground truth entities. Exact match is 1 if the LLM's answer and a ground truth entity are exactly the same. The F1 score is the harmonic mean of precision and recall. Recall is defined as the ratio of the number of overlapping tokens between the LLM's answer and the ground-truth entity to the total number of tokens in the ground-truth entity. Precision is defined as the ratio of the number of overlapping tokens between the LLM's answer and the ground-truth entity to the total number of tokens in the LLM's answer. In cases with multiple ground truth entities, we report the highest accuracy, EM, and F1 among the scores obtained from each ground truth entity.

Retriever Metrics. To evaluate the effectiveness of our retriever, we employ two metrics: Accuracy at K (Acc@K) and Mean Reciprocal Rank at 10 (MRR@10). Acc@K measures the proportion of questions for which at least one triple with any answer entities appears within the top K retrieved triples. MRR@10 considers the position of the first triple with an answer entity among the top 10 retrieved triples. Formally, it is defined as: $\frac{1}{|Q|}\sum_{q\in Q}\frac{1}{\text{rank}_q}$ where rank_q is the position of the first triple that includes an answer entity for question q within the top 10 results. If an answer entity is not in any of the top 10 triples, the reciprocal rank is 0. In our experiments, we report these metrics as RAcc@1, RAcc@5, RAcc@10, and RMRR@10, where R indicates the retriever.

[4]https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct

Table 1
Type injection results in Mintaka and WebQSP datasets. The variant names with "Sem-" refer to type enhancement with the semantically most relevant type to the question, the type variants without "Sem-" indicate injection of a random type (the first type that appeared in 1-hop neighbor triples).

Dataset	Mintaka							WebQSP						
Metric	Acc	RAcc@1	RAcc@5	RAcc@10	RMRR@10	EM	F1	Acc	RAcc@1	RAcc@5	RAcc@10	RMRR@10	EM	F1
Model	T0-3B													
KAPING	21.49	8.60	**17.65**	22.02	12.31	18.47	27.89	53.88	32.40	**56.20**	**63.84**	42.50	**44.13**	54.31
KAPING-Inst[Q+K]	20.54	7.90	15.85	20.6	11.40	17.44	26.01	50.68	31.71	53.68	61.18	41.27	34.92	49.31
KAPING-Inst[Q]	20.71	7.15	15.6	20.92	10.86	17.80	27.04	48.77	25.17	50.75	59.20	35.84	38.47	49.23
KAPING-Inst[K]	21.74	8.70	16.87	20.85	12.15	18.01	27.44	52.66	**33.83**	54.84	61.80	**42.92**	38.06	51.44
KAPING-SubC[Q+K]	21.53	7.97	16.97	21.65	11.69	18.72	27.77	51.63	30.76	52.79	62.14	40.67	36.22	50.40
KAPING-SubC[Q]	21.78	8.15	17.20	21.6	11.83	18.69	28.29	53.88	30.69	54.77	63.30	40.93	43.65	54.41
KAPING-SubC[K]	21.78	8.52	17.47	**22.15**	12.24	19.04	27.65	53.47	32.53	55.04	63.57	42.65	37.51	51.59
KAPING-Sem-Inst[Q+K]	20.68	8.05	16.02	20.65	11.49	17.12	26.20	49.93	31.78	53.47	61.59	41.41	36.56	49.67
KAPING-Sem-Inst[Q]	21.21	7.27	15.75	20.7	10.95	17.98	27.25	49.24	25.71	50.61	59.34	36.30	38.67	49.95
KAPING-Sem-Inst[K]	21.25	**8.92**	17.07	21.52	**12.44**	17.44	26.79	53.13	32.94	54.91	62.34	42.60	36.35	51.25
KAPING-Sem-SubC[Q+K]	21.74	8.07	16.82	21.57	11.72	18.69	28.14	52.86	30.49	52.93	62.14	40.53	37.51	50.97
KAPING-Sem-SubC[Q]	21.28	8.15	17.15	21.57	11.82	18.44	**28.64**	**54.22**	30.55	55.25	63.50	40.97	43.86	**54.62**
KAPING-Sem-SubC[K]	**22.28**	8.6	17.32	22.07	12.27	**19.36**	28.55	53.34	32.19	55.38	63.36	42.44	38.60	51.29
Model	flan-t5-xl													
KAPING	22.74	8.60	**17.65**	22.02	12.31	18.44	27.89	59.14	32.40	**56.20**	**63.84**	42.50	34.03	48.16
KAPING-Inst[Q+K]	22.60	7.72	15.77	20.70	11.27	13.46	23.39	55.66	31.71	53.68	61.18	41.27	24.01	39.38
KAPING-Inst[Q]	23.20	7.12	15.55	20.90	10.83	14.99	25.37	54.43	25.17	50.75	59.20	35.84	25.10	42.50
KAPING-Inst[K]	23.45	8.72	16.90	20.95	12.20	17.69	27.46	58.39	**33.83**	54.84	61.80	**42.92**	30.76	45.87
KAPING-SubC[Q+K]	23.24	8.05	16.97	21.65	11.72	18.47	28.43	57.43	30.76	52.79	62.14	40.67	28.64	45.62
KAPING-SubC[Q]	22.70	8.15	17.20	21.57	11.83	17.94	27.88	58.79	30.69	54.77	63.30	40.93	33.42	47.97
KAPING-SubC[K]	23.48	8.60	17.50	**22.2**	12.29	**19.36**	**28.74**	59.00	32.53	55.04	63.57	42.65	32.12	47.82
KAPING-Sem-Inst[Q+K]	**23.66**	8.05	16.02	20.65	11.49	13.75	23.56	55.18	31.78	53.47	61.59	41.41	23.94	39.18
KAPING-Sem-Inst[Q]	23.24	7.27	15.75	20.70	10.95	15.31	25.37	54.16	25.71	50.61	59.34	36.30	25.37	42.75
KAPING-Sem-Inst[K]	21.85	**8.92**	17.07	21.52	**12.44**	17.16	26.44	58.25	32.94	54.91	62.34	42.60	30.08	44.59
KAPING-Sem-SubC[Q+K]	23.20	8.07	16.82	21.57	11.72	18.08	27.95	58.18	30.49	52.93	62.14	40.53	30.49	46.68
KAPING-Sem-SubC[Q]	22.88	8.15	17.15	21.57	11.82	18.08	27.91	**59.20**	30.55	55.25	63.50	40.97	**34.10**	**48.60**
KAPING-Sem-SubC[K]	23.59	8.6	17.32	22.07	12.27	18.94	28.53	57.84	32.19	55.38	63.36	42.44	30.35	47.01
Model	Llama-3.2-3B-Instruct													
KAPING	53.09	8.6	**17.65**	22.02	12.31	5.04	13.56	69.30	32.40	**56.20**	**63.84**	42.50	**17.05**	**31.03**
KAPING-Inst[Q+K]	54.65	7.77	15.72	20.57	11.26	2.62	11.21	70.05	32.12	53.81	61.32	41.62	6.34	21.63
KAPING-Inst[Q]	53.44	7.27	15.75	20.7	10.95	2.48	10.47	67.93	25.71	50.61	59.34	36.30	4.91	18.31
KAPING-Inst[K]	51.77	8.72	16.9	20.95	12.20	5.18	13.86	69.16	**33.83**	54.84	61.80	**42.92**	15.41	30.58
KAPING-SubC[Q+K]	54.72	8.05	16.95	21.65	11.72	4.54	12.57	69.09	30.83	53.00	62.41	40.82	13.43	27.42
KAPING-SubC[Q]	53.83	8.15	17.15	21.57	11.82	4.37	12.23	69.23	30.55	55.25	63.50	40.97	16.16	29.70
KAPING-SubC[K]	54.15	8.6	17.5	**22.2**	12.29	5.11	13.72	69.84	32.53	55.04	63.57	42.65	15.34	29.70
KAPING-Sem-Inst[Q+K]	**54.76**	8.05	16.02	20.65	11.49	3.02	12.00	**70.73**	31.78	53.47	61.59	41.41	5.79	21.66
KAPING-Sem-Inst[Q]	54.01	7.27	15.75	20.7	10.95	2.91	11.08	67.32	25.71	50.61	59.34	36.30	4.84	18.15
KAPING-Sem-Inst[K]	52.52	**8.92**	17.07	21.52	**12.44**	5.01	**14.43**	68.69	32.94	54.91	62.34	42.60	13.43	29.88
KAPING-Sem-SubC[Q+K]	53.62	8.07	16.82	21.57	11.72	4.51	12.88	70.12	30.49	52.93	62.14	40.53	13.84	28.49
KAPING-Sem-SubC[Q]	53.44	8.15	17.15	21.57	11.82	4.29	12.34	68.28	30.55	55.25	63.50	40.97	15.62	29.41
KAPING-Sem-SubC[K]	52.77	8.6	17.32	22.07	12.27	**5.68**	13.81	69.09	32.19	55.38	63.36	42.44	15.62	29.86

Table 2
The effect of adding Wikidata instanceOf type information to WebQSP. LLama2-7B refers to Llama-2-7b-chat-hf model.

Dataset	WebQSP								
Metric	RAcc@1	RAcc@5	RAcc@10	RMRR@10	T0-Acc	flan-t5-Acc	Llama3.2-3B-Acc	LLama2-7B-Acc	Mistral7B-Acc
KAPING	32.40	56.20	63.84	42.50	**53.88**	59.14	69.30	74.76	74.01
Wikidata-Inst[K]-NoHuman	**34.51**	**57.77**	**63.91**	**44.52**	53.27	**59.41**	**69.64**	**75.44**	**75.10**

4.2. Experiment 1: Enhancing Prompts with Entity Types

This experiment aims to evaluate the effectiveness of injecting entity types into the questions and the KG triples. The goal is to measure the extent to which entity types enhance KGQA systems' performance in retrieving the relevant triples and answering questions accurately. In Table 1, we compare the performances of KAPING [7] with and without entity types. KAPING attaches relevant facts from the knowledge graph into the language model prompt and performs zero-shot knowledge graph question answering. KAPING retrieves top-k triples from $\bigcup_{i=1}^{n}\{\mathcal{N}_{e_{q_i}}\}$ based on their semantic similarity to the question. For this purpose, they employ off-the-shelf bi-encoder retriever Sentence-BERT [16][5].

We chose KAPING as the base model to enhance a retriever as it offers a simple pipeline, uses SentenceBERT retriever without involving complex verbalization steps or requiring multiple LLM calls. This design makes KAPING suitable for evaluating our enhancements in a controlled and minimalistic RAG setting. There are RAG-based KGQA systems that either have multiple LLM calls (e.g. [30] uses a 13B-parameter KG-to-Text model based on Llama-2-chat [31]), or have a verbalization component

[5]https://huggingface.co/sentence-transformers/all-mpnet-base-v2

Table 3
Qualitative examples for type injection. The first example is on the Mintaka dataset with T0-3B model, and the second example is on the WebQSP dataset with flan-t5-xl model, both in KAPING-Sem-Inst[Q+K] setting. The third example is on the Mintaka dataset with Llama-3.2-3B-Instruct model in KAPING-Inst[Q+K] setting.

Ground-truth answer(s)	**Without types**	**With types**
Paranoid	question: what black sabbath album contains the song iron man? knowledge: ['Iron Man', 'performer', 'Black Sabbath'] ['Black Sabbath', 'instance of', 'heavy metal band'] ['Black Sabbath', 'discography', 'Black Sabbath discography'] ['Iron Man', 'genre', 'heavy metal'] ['Black Sabbath', 'genre', 'hard rock'] ['Black Sabbath', 'genre', 'heavy metal'] ['Iron Man', 'instance of', 'song'] ['Black Sabbath', 'genre', 'doom metal'] ['Black Sabbath', 'genre', 'traditional heavy metal'] ['Iron Man', 'lyrics by', 'Ozzy Osbourne'] no knowledge answer: Iron Man with knowledge answer: Iron Man	question: what black sabbath album contains the song iron man? Black Sabbath is a heavy metal band. Iron Man is a song. knowledge: ['Black Sabbath (heavy metal band)', 'discography', 'Black Sabbath discography'] ['Iron Man (song)', 'performer', 'Black Sabbath (heavy metal band)'] ['Iron Man (song)', 'part of', 'Paranoid (album)'] ['Iron Man (song)', 'genre', 'heavy metal (music genre)'] ['Black Sabbath (heavy metal band)', 'Commons category', 'text: Black Sabbath'] ['Black Sabbath (heavy metal band)', 'instance of', 'heavy metal band'] ['Black Sabbath (heavy metal band)', 'genre', 'heavy metal (music genre)'] ['Black Sabbath (heavy metal band)', 'genre', 'hard rock (music genre)'] ['Iron Man (song)', 'publication date', 'time: +1971-10-00'] ['Iron Man (song)', 'lyrics by', 'Ozzy Osbourne (human)'] no knowledge answer: Black Sabbath with knowledge answer: Paranoid
Juris Doctor Bachelor of Arts	question: what degrees did obama get in college? knowledge: ['Barack Obama', 'academic degree', 'Bachelor of Arts'] ['Barack Obama', 'educated at', 'Columbia University'] ['Barack Obama', 'educated at', 'Harvard University'] ['Barack Obama', 'academic degree', 'Juris Doctor'] ['Barack Obama', 'educated at', 'Harvard Law School'] ['Barack Obama', 'educated at', 'Occidental College'] ['Barack Obama', 'educated at', 'University of Chicago Law School'] ['Barack Obama', 'educated at', 'State Elementary School Menteng 01'] ['Barack Obama', 'educated at', 'Centaurus High School'] ['Barack Obama', 'field of work', 'politics'] no knowledge answer: Political Science with knowledge answer: Bachelor of Arts, Juris Doctor, Harvard Law School, Occidental College, University of Chicago Law School	question: what degrees did obama get in college? knowledge: ['Barack Obama', 'academic degree', "Bachelor of Arts (bachelor's degree)"] ['Barack Obama', 'academic degree', 'Juris Doctor (law degree)'] ['Barack Obama', 'educated at', 'Columbia University'] ['Barack Obama', 'educated at', 'Harvard University'] ['Barack Obama', 'educated at', 'Harvard Law School'] ['Barack Obama', 'educated at', 'Occidental College'] ['Barack Obama', 'educated at', 'University of Chicago Law School'] ['Barack Obama', 'educated at', 'State Elementary School Menteng 01'] ['Barack Obama', 'educated at', 'Centaurus High School'] ['Barack Obama', 'field of work', 'politics (activity)'] no knowledge answer: Political Science with knowledge answer: Bachelor of Arts (bachelor's degree)), Juris Doctor (law degree)
The Piper at the Gates of Dawn	question: what was the first pink floyd album titled? no knowledge answer: Pink Floyd	question: what was the first pink floyd album titled? Pink Floyd is a rock group. no knowledge answer: The first Pink Floyd album was titled "The Piper at the Gates of Dawn". It was released in 1967.

[32], or include post-retrieval steps. These extra steps make it harder to isolate the effect of type enhancements. Additional method results are provided in Section A.1; for instance, Wu et al. [30] achieve 74.02 with the T0-3B model and the Freebase knowledge graph.

As can be seen in Table 1, for the Mintaka dataset, the entity type variants achieved the highest values in all metrics for all models, except for retriever accuracy @5. For the WebQSP dataset, the type variants outperform KAPING in the accuracy metric. For both datasets, Inst[K] (both random and semantic) improves retriever acc @1 and Sem-Inst[K] improves RMRR@10. **These results demonstrate the effectiveness of the Inst[K] setting for retrieval**. Therefore, enhancing the knowledge triples with the classes can support the retriever to find correct answers more effectively. To further test this, we conduct two additional experiments. First, we extract instanceOf relations from Wikidata for the WebQSP dataset using SPARQL. We remove the *Human* type because it is highly frequent and uninformative. Table 2 shows that this type enhancement improves retrieval performance slightly across all retriever metrics and also increases the accuracy of all models, including larger models, with the exception of T0. Secondly, we experiment on the temporal KGQA dataset TimeQuestions [33]. We follow the same experimental setting as [34] and perform retrieval on a vector database (therefore removing the need for question entity IDs). Consistent with our prior findings, we observe an improvement from 40.90 to 41.05 in RAcc@1 and from 62.95 to 63.23 in RAcc@5.

In terms of generation, SubC[K] variants improve the LLM's accuracy in Mintaka, and Sem-SubC[Q] results in improved generation performance on the WebQSP dataset for both the T0-3B and flan-t5-xl models. These results suggest that injecting superclasses is particularly beneficial for the language model responses, rather than directly contributing to the retrieval.

The results in Table 1 are consistent with prior literature [10, 11, 12, 13, 14, 35] that show incorporating types improves question answering and link prediction performance. To further demonstrate the value

Table 4
The results for different retrievers on Mintaka and WebQSP datasets.

Dataset	Mintaka							WebQSP						
	Acc	RAcc@1	RAcc@5	RAcc@10	RMRR@10	EM	F1	Acc	RAcc@1	RAcc@5	RAcc@10	RMRR@10	EM	F1
Model	T0-3B													
No Knowledge	16.52	-	-	-	-	11.05	20.48	37.44	-	-	-	-	11.45	27.22
Pretrained bi-encoder	21.49	8.6	17.65	22.02	12.31	18.47	27.89	53.88	32.40	56.20	63.84	42.50	44.13	54.31
Pretrained cross-encoder	19.82	7.15	15.9	20.25	10.73	17.23	26.39	46.04	13.71	34.51	47.47	22.30	36.22	47.22
KR2-cross-encoder	20.43	6.42	15.07	19.82	10.16	17.76	27.16	42.63	8.18	27.83	42.22	16.85	33.08	44.56
KR2-bi-encoder	20.00	7.12	15.90	20.22	10.84	16.84	25.90	40.99	16.37	32.46	40.72	23.53	31.44	42.12
Model	flan-t5-xl													
No knowledge	21.39	-	-	-	-	19.58	27.53	40.65	-	-	-	-	37.24	47.11
Pretrained bi-encoder	22.74	8.6	17.65	22.02	12.31	18.44	27.89	59.14	32.40	56.20	63.84	42.50	34.03	48.16
Pretrained cross-encoder	23.16	7.2	15.9	20.25	10.76	17.76	27.78	50.27	13.71	34.51	47.47	22.30	28.58	42.25
KR2-cross-encoder	22.45	6.42	15.07	19.82	10.16	17.73	27.29	45.49	8.18	27.83	42.22	16.85	26.94	40.73
KR2-bi-encoder	22.13	7.12	15.90	20.22	10.84	18.83	27.04	45.70	16.37	32.46	40.72	23.53	28.44	40.20
Model	Llama-3.2-3B-Instruct													
No knowledge	58.91	-	-	-	-	1.81	15.05	69.98	-	-	-	-	1.97	17.18
Pretrained bi-encoder	53.09	8.6	17.65	22.02	12.31	5.04	13.56	69.30	32.40	56.20	63.84	42.50	17.05	31.03
Pretrained cross-encoder	53.62	7.17	15.90	20.25	10.75	4.61	12.56	63.91	13.71	34.51	47.47	22.30	14.25	28.06
KR2-cross-encoder	52.66	6.35	15.05	19.82	10.11	4.79	12.62	61.25	8.18	27.83	42.22	16.85	12.55	25.63
KR2-bi-encoder	51.77	7.12	15.90	20.22	10.84	3.23	10.28	62.82	16.37	32.46	40.72	23.53	10.36	22.12

of types for the KGQA task, we conducted a qualitative analysis, the results of which are presented in Table 3. In the first example, we observe how the types help the retrieval. The fact (Iron Man, part of, Paranoid) was not retrieved without the type injection. The second example illustrates how types help model to filter its answers. In this case, the university and college names are excluded from the answer, and only the degree names are included. In the last example, we see that the type injection into the question helps the model even without any knowledge retrieval.

Overall, we observe several challenges related to type injection that affect its effectiveness. First, we observe that not all entities in Wikidata are associated with a type and this leads to sparse enhancement. Second, some types are overly generic (e.g., "Human"), thus they provide limited discriminative value. Lastly, in some cases, the entity label already indicates the type (e.g., the entity "English Language" has type "Language"), and this results in redundant or less informative type injection.

4.3. Experiment 2: KG-Aware Retrievers

To evaluate the performance of our KR2 retrievers (KR2-cross-encoder and KR2-bi-encoder), we compare them against a *No knowledge* baseline, a pre-trained bi-encoder baseline and a pre-trained cross-encoder baseline. The last two are text-based encoders, and are capable of scoring general (question, triple) pairs like KR2. Other approaches either use the same pre-trained bi-encoder as a retriever or use a subgraph retriever [15, 30], are therefore not applicable to a general (question, triple) scoring. Although they are not directly comparable, we include their results in Section A.1. Additionally, we report the results of an experiment combining Approach 1 (Experiment 1) and Approach 2 (Experiment 2) in Section A.3.

In the *No knowledge* setting, the LLM answers the question without any additional context from the graph. As the pre-trained bi-encoder baseline, we chose the commonly used bi-encoder "all-mpnet-base-v2", which is used in KAPING [7]. It is a fine-tuned version of the pre-trained MPNet-base [36] model, trained on 1.1 billion sentence pairs obtained from 32 different datasets. As the pre-trained cross-encoder baseline, we chose the most used cross-encoder model in the HuggingFace library: "ms-marco-MiniLM-L6-v2"[6]. It is fine-tuned on more than 530k (query, passage) samples from the MS MARCO dataset[7] sourced from the Bing search engine [37].

We report the results in Table 4. On the Mintaka dataset, KR2-cross-encoder outperforms the pretrained cross-encoder in all generation metrics with T0-3B, and achieves comparable performance on retriever metrics relative to the baselines. KR2-bi-encoder model achieves higher retriever results than KR2-cross-encoder, however, it obtains lower results in generation accuracy. On the WebQSP dataset, KR2-bi-encoder outperforms the pretrained cross-encoder in RAcc@1 and RMRR@10. The pre-trained bi-encoder's strong performance may be attributed to its large-scale training data with 1.1B samples. Despite low results in retriever metrics, KR2-cross-encoder achieves comparable performance

[6]https://huggingface.co/cross-encoder/ms-marco-MiniLM-L6-v2
[7]https://microsoft.github.io/msmarco/Datasets.html

Table 5
2-hop retrieval results on WebQSP.

Dataset	WebQSP						
	Acc	RAcc@1	RAcc@5	RAcc@10	RMRR@10	EM	F1
Model	T0-3B						
KR2-cross-encoder-2-hop	30.55	12.27	25.98	34.58	18.12	23.46	33.97
KR2-bi-encoder-2-hop	34.92	16.16	30.76	37.03	22.28	27.08	36.90
Model	flan-t5-xl						
KR2-cross-encoder-2-hop	34.37	12.27	25.98	34.58	18.12	20.60	32.72
KR2-bi-encoder-2-hop	37.38	16.16	30.76	37.03	22.28	27.48	37.85
Model	Llama-3.2-3B-Instruct						
KR2-cross-encoder-2-hop	50.27	12.27	25.98	34.58	18.12	6.61	17.20
KR2-bi-encoder-2-hop	56.41	16.16	30.76	37.03	22.28	6.75	17.22

Table 6
Transferability results on Mintaka and WebQSP datasets.

Dataset	Mintaka							WebQSP						
	Acc	RAcc@1	RAcc@5	RAcc@10	RMRR@10	EM	F1	Acc	RAcc@1	RAcc@5	RAcc@10	RMRR@10	EM	F1
Model	T0-3B													
Across-KG cross-encoder	19.33	5.15	13.77	18.92	9.02	16.52	26.18	42.01	7.70	25.30	40.99	15.76	32.94	43.89
Across-KG bi-encoder	18.30	5.50	13.20	17.87	8.81	15.67	24.84	46.45	19.64	38.88	47.33	27.94	35.67	46.65
Model	flan-t5-xl													
Across-KG cross-encoder	22.13	5.45	13.77	18.92	9.19	17.66	27.09	44.61	7.70	25.30	40.99	15.76	26.39	39.90
Across-KG bi-encoder	20.68	5.50	13.20	17.87	8.81	16.59	25.73	50.20	19.64	38.88	47.33	27.94	31.17	43.89
Model	Llama-3.2-3B-Instruct													
Across-KG cross-encoder	50.24	5.10	13.77	18.92	8.99	5.33	13.67	58.73	7.70	25.30	40.99	15.76	15.68	28.89
Across-KG bi-encoder	51.24	5.50	13.20	17.87	8.81	4.83	12.65	65.62	19.64	38.88	47.33	27.94	14.46	27.81

in generation metrics. This indicates that even if the KG-aware cross-encoder retriever fails to catch the triple with the correct answer entity, it still creates a useful knowledge context for the language model. Finally, we expect KR2's performance to improve on questions and datasets that require domain-specific knowledge and when it is trained with harder negative samples.

2-hop Retrieval and Retriever Transferability KR2 is generalizable to a multi-hop setting. To obtain 2-hop retrieval results, we extend the search space beyond the immediate neighborhood of the question entities. Specifically, given a question entity e_{q_i}, we first retrieve its 1-hop neighbors $\mathcal{N}_{e_{q_i}}$. Then, for each of the neighbor entity $e_n \in \mathcal{N}_{e_{q_i}}$, we get its neighbors $\mathcal{N}_{e_n}$ to form a 2-hop subgraph centered around the question entity e_{q_i}. However, expanding the search depth from one to two hops significantly increases the computational cost. Table 5 presents the results of 2-hop retrieval on the WebQSP dataset. Consistent with findings from [7], introducing 2-hop retrieval reduces performance across all models and evaluation metrics with the exception of RAcc@1 and RMRR@10 for the KR2-cross-encoder-2-hop variant.

Moreover, a trained KR2 retriever can score arbitrary (question, triple) pairs regardless of the KG. Therefore, a trained retriever can be applied directly to a different KG without retraining or additional adaptation. This enables zero-shot transfer across KGs. To evaluate this capability, we apply the retriever trained on Mintaka to WebQSP questions, and vice versa. As shown in Table 6, performance generally decreases as expected (except for the WebQSP bi-encoder) but remains competitive, which indicates generalizability across datasets. We observe that this also may stem from high overlap between the two KGs: approximately 58% of the Mintaka training (query, positive) samples also appear in WebQSP.

5. Related Work

Recent advances in RAG [5, 38, 39] have mitigated several limitations of LLMs in downstream tasks such as question answering, information extraction, and fact verification. Despite their effectiveness, traditional RAG approaches typically ignore the inherent structure of the retrieved data. Incorporating structured knowledge [40, 41], such as knowledge graphs, introduces additional challenges.

KG-RAG for Question Answering. Given a question and a knowledge graph, KnowledgeNavigator [42] first predicts the number of hops required by the question using a fine-tuned pre-trained language model. The framework then iteratively predicts the relevant relations for each hop to reduce the search space. Finally, the triples with the most relevant relations are retrieved as the context. RoG [43] is

a framework that employs an LLM jointly trained for two instruction-tuning tasks: planning and reasoning. Given a question, the framework first generates relation paths (or "plans") grounded in the KG. Based on these plans, it then retrieves the corresponding KG triples, which are provided to the LLM to generate the final answer. MindMap [44] combines LLMs' internal knowledge with external KG facts. The framework first matches the question with the corresponding KG entities by using BERT [17]. Then it extracts two different subgraphs: one that connects these entities and another one with the neighbors of these entities. Next, they are aggregated by the LLM, e.g., by finding new connections between input entities. Therefore, the framework does not fully rely on KG. Finally, the aggregated knowledge is used as contextual input for the LLM to generate an answer to the given question. Moreover, the LLM builds a mind map based on the supporting evidence from KG to explain the reasoning behind the answer and enhance explainability. All of these approaches have multiple LLM calls, which increases the cost. StructGPT [45] is a framework that unifies information retrieval from different types of structured data (tables, KGs, and DBs) for their corresponding tasks (TableQA, KGQA, and Text-to-SQL). It defines specialized functions for accessing each type of data and the accessed data is sent to an LLM to obtain an answer for a given question. For KGQA, the framework prompts the LLM to choose the relevant relations connecting the question entity, then extracts triples in the format (question entity, chosen relation, object). For multi-hop reasoning, the LLM decides whether the current information is sufficient or additional triples from the next hop are needed.

KGQA. Traditional knowledge graph question answering (KGQA) methods are classified into two approaches: Semantic parsing (SP) and Information Retrieval (IR) approaches. SP models map the question to a logical form, which is then executed in the knowledge graph to produce an answer. Although logical forms provide interpretable reasoning steps, these approaches have several drawbacks [46]: (i) it is challenging to map complex queries into logical forms, (ii) these systems require and heavily depend on ground-truth logical forms, which most datasets do not have and are expensive to annotate. Regarding scalability, the search space for logical forms expands significantly with the number of entities and relations. Moreover, IR models first retrieve a subgraph for a question and then perform reasoning on that subgraph to answer the question [46]. Although this approach offers end-to-end training, its black-box nature brings a disadvantage. Additionally, the training requires reasoning paths, which are expensive to obtain. Both approaches face a common challenge: language understanding.

KG-RAG for KGQA. Retrieval-augmented generation offers a new approach for KGQA [47, 48, 49, 50] and addresses the language understanding challenge [51, 52]. Baek et al. [7] provide a framework called KAPING that injects relevant facts from the knowledge graph into the language model prompt and performs zero-shot knowledge graph QA. KAPING first finds the triples that include question entities and chooses top-K triples based on their semantic similarity to the question. For this purpose, they employ off-the-shelf retriever Sentence-BERT [16] as a retriever. Sen et al. [15] focus on complex question-answering by a KGQA model retriever and a language model reasoner. First, it predicts a distribution over relations to follow in every hop. The second step combines the distribution with question entities to output weighted triples, which the language model will reason over. Wu et al. [30] provides a KGQA framework focusing on improving the verbalization of the information from the knowledge graph. It consists of three modules: (i) retrieve, (ii) rewrite, and (iii) answer. The retrieve part first predicts the hop number of the question by a classifier. The relation path prediction module uses this predicted number to return a ranked list of relation paths for this hop number. Then the relation classifier takes the question and a relation path (initially empty) and predicts top K relations as candidates. In the case of a multi-hop setting, the candidate relation(s) from the previous steps form a relation path, and the classifier takes the question and this relation path to output relation candidates for the current hop. Unlike KR2, this model requires ground-truth relation paths and hop numbers to train these classifiers. Moreover, EFSum [32] uses an LLM as a fact summarizer to verbalize the KG facts (therefore also makes multiple LLM calls). First, the fact summarizer is trained with the reference summaries obtained through LLM prompting. The summarizer then fine-tuned with direct preference optimization (DPO) [53] on the preferred and dispreferred summary pairs. The objective is two-fold: a summary provides enough evidence to facilitate the LLM to answer correctly, and it does not include

anything inconsistent with the initial set of facts.

KG-RAG with Cross-Encoders (Rerankers) KC-GenRe [54] finetunes a decoder-only large language model to rerank the candidate entities for the knowledge graph completion task. [55] is used to rerank triples and [56] is used to rerank the paths in knowledge graph, however, those approaches are using off-the-shelf pre-trained rerankers. KG-Rank [57] uses MedCPT [58] to rerank the retrieved triples, where MedCPT is a cross-encoder model initialized by PubMedBERT [59]. This reranker is trained using 18.3M (query, article) pairs from PubMed search logs, therefore it does not use a KG and needs query-article pairs. [60] trains a re-ranker with a dataset which is manually annotated. [61] designs a contextual re-ranker for KGQA. The index is entities and the retriever returns top-K entities. The reranker takes (question, candidate triple from an entity, context from the same entity), where the context is all the other triples of the same entity. To train such a reranker, the study introduces triple-level labeling strategy: it gives a positive label to triples that contain the gold answer and a topic (question) entity. If no such triple exists, it removes the topic entity constraint and checks only for the gold answer. Unlike KR2, it requires a training dataset with query-answer pairs. In addition, since the *context* includes all triples associated with an entity, there is a risk of high computational cost when dealing with hub entities, which have a high number of neighbors.

6. Conclusion

We present KR2, a knowledge-enhanced retrieval framework for KGQA within a RAG setting. KR2 introduces a novel way to train a cross-encoder and a bi-encoder retriever with a KG. Unlike previous approaches, it does not require any annotated question-answer pairs for training. KR2 is also the first attempt at using types and class hierarchies for RAG-based KGQA. By leveraging the entity types with *instanceOf* and *subclassOf* relations, KR2 improves the quality of retrieved triples and enhances the accuracy of LLM-generated answers. Building on this, future work will explore ontology axioms beyond these relations.

Declaration on Generative AI

The author(s) used ChatGPT in order to: Grammar and spelling check, Paraphrase and reword, Improve writing style. After using this tool, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

Supplemental Material Statement

We provide the following links for reproducibility: The source code, the trained models and training datasets are available at:
https://github.com/duyguislakoglu/kg-rag

A. Appendix

A.1. Other KG-RAG for KGQA approaches

For the sake of completeness, we include the results of existing KG-RAG methods in Table 7. Some of these methods (such as [30]) use a verbalization module that enables triple to natural language sentence translation. Their findings show that adding a verbalization module enhances performance, this could be due to the fact that the verbalized format aligns more closely with the models' pre-training corpora. Additionally, both [15] and [30] assign weights to the KG relations for a given question. This could be an alternative for triple relevance to the given question, which we employ for our method KR2. As noted in Section 4.3, these methods are not our baselines; nevertheless, they achieve promising results on KGQA and outperform KR2.

A.2. Evaluation with Larger Language Models

To analyze the effect of model size, we experiment with larger models: Mistral-7B-Instruct-v0.2 [62][8], Meta-Llama-3.1-8B-Instruct[9], Gemma-2-9B-it[10]. The results in Table 8 show that these models perform worse than the no knowledge setting. This might suggest that larger models are more affected by irrelevant information in the knowledge context. This could be due to the fact that these models highly depend on the context as they are instruction-tuned and therefore do not utilize the internal knowledge. In future work, we plan to investigate potential reasons, including experimenting with different prompt templates.

A.3. Combining KG-Aware Cross-Encoder with Type Injection

In this experiment, we combine both approaches, so we integrate the KG-aware cross-encoder retriever and entity types. We report the results in Table 9. We observe that this integration does not lead to performance improvements. One possible explanation is that the KG-aware retriever is already type-aware, as its training data includes triples containing *subclassOf* and *instanceOf* relations. Another potential reason is that type enhancement alters the textual format of the input, which may be incompatible with the KG-aware retriever's learned representations.

Table 7
The accuracy values of different models on Mintaka, WebQSP and WebQ [21] datasets. WebQ and WebQSP have the same set of questions. "Freebase" indicates that the underlying knowledge graph is based on Freebase.

	T0-3B	flan-t5-xl
Mintaka		
Sen [15]	29.28	33.50
WebQSP		
KAPING$_{Freebase}$	52.28	-
Wu$_{Freebase}$ [30]	74.02	-
WebQ		
Sen [15]	53.33	55.58

[8]https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2
[9]https://huggingface.co/meta-llama/Meta-Llama-3.1-8B-Instruct
[10]https://huggingface.co/google/gemma-2-9b-it

Table 8
KGQA results for larger models.

	Acc	RAcc@1	RAcc@5	RAcc@10	RMRR@10	EM	F1
Mintaka							
Model	Mistral-7B-Instruct-v0.2						
No Knowledge	72.21	-	-	-	-	0	6.99
KAPING	52.55	8.6	17.65	22.02	12.31	0	6.93
Model	Llama-3.1-8B-Instruct						
No Knowledge	71.53	-	-	-	-	0.74	17.07
KAPING	66.34	8.6	17.65	22.02	12.31	0.14	5.17
Model	gemma-2-9b-it						
No Knowledge	71.60	-	-	-	-	1.52	20.84
KAPING	39.87	8.6	17.65	22.02	12.31	3.44	9.85
WebQSP							
Model	Mistral-7B-Instruct-v0.2						
No Knowledge	76.46	-	-	-	-	0	6.76
KAPING	74.01	32.40	56.20	63.84	42.50	0	13.36
Model	Llama-3.1-8B-Instruct						
No knowledge	78.78	-	-	-	-	1.56	17.46
KAPING	75.64	32.40	56.20	63.84	42.51	2.04	12.03
Model	gemma-2-9b-it						
No knowledge	74.62	-	-	-	-	0.47	21.37
KAPING	67.32	32.40	56.20	63.84	42.51	9.20	21.76

Table 9
Integrating KG-aware cross-encoder and entity types.

	Acc	RAcc@1	RAcc@5	RAcc@10	RMRR@10	EM	F1
Mintaka							
Model	T0-3B						
KR2-cross-encoder	20.43	6.42	15.07	19.82	10.16	17.76	27.16
KR2-cross-encoder-Sem-Inst[Q+K]	19.15	6.12	13.52	18.42	9.42	15.95	24.69
KR2-cross-encoder-Sem-Inst[Q]	19.68	6.0	13.6	18.55	9.36	17.19	26.48
KR2-cross-encoder-Sem-Inst[K]	20.07	6.6	15.25	19.9	10.24	16.70	26.04
KR2-cross-encoder-Sem-SubC[Q+K]	19.50	5.87	14.4	19.77	9.70	16.80	26.11
KR2-cross-encoder-Sem-SubC[Q]	20.07	6.12	14.62	19.57	9.86	17.48	27.31
KR2-cross-encoder-Sem-SubC[K]	20.11	5.97	14.75	19.97	9.88	17.41	26.50
Model	flan-t5-xl						
KR2-cross-encoder	22.45	6.42	15.07	19.82	10.16	17.73	27.29
KR2-cross-encoder-Sem-Inst[Q+K]	21.89	6.02	13.57	18.42	9.37	12.68	22.95
KR2-cross-encoder-Sem-Inst[Q]	21.25	5.82	13.6	18.57	9.27	13.18	23.02
KR2-cross-encoder-Sem-Inst[K]	21.46	6.5	15.3	19.9	10.19	16.73	26.23
KR2-cross-encoder-Sem-SubC[Q+K]	21.89	5.8	14.35	19.77	9.67	16.98	26.55
KR2-cross-encoder-Sem-SubC[Q]	21.32	6.12	14.65	19.57	9.86	16.87	26.62
KR2-cross-encoder-Sem-SubC[K]	22.28	5.95	14.7	19.97	9.86	17.55	27.32
WebQSP							
Model	T0-3B						
KR2-cross-encoder	42.63	8.18	27.83	42.22	16.85	33.08	44.56
KR2-cross-encoder-Sem-Inst[Q+K]	34.37	5.72	19.71	30.01	11.65	23.26	36.02
KR2-cross-encoder-Sem-Inst[Q]	33.96	5.04	21.96	33.28	12.47	25.57	36.94
KR2-cross-encoder-Sem-Inst[K]	42.70	7.09	24.69	39.42	15.28	30.15	42.22
KR2-cross-encoder-Sem-SubC[Q+K]	40.10	6.75	24.42	38.13	14.56	28.99	42.32
KR2-cross-encoder-Sem-SubC[Q]	41.95	7.50	26.26	40.31	15.87	32.46	44.40
KR2-cross-encoder-Sem-SubC[K]	42.83	7.29	25.98	40.24	15.53	30.62	43.35
Model	flan-t5-xl						
KR2-cross-encoder	45.49	8.18	27.83	42.22	16.85	26.94	40.73
KR2-cross-encoder-Sem-Inst[Q+K]	35.06	5.72	19.71	30.01	11.65	14.66	26.33
KR2-cross-encoder-Sem-Inst[Q]	36.76	5.04	21.96	33.28	12.47	16.84	30.04
KR2-cross-encoder-Sem-Inst[K]	44.06	7.09	24.69	39.42	15.28	23.66	37.71
KR2-cross-encoder-Sem-SubC[Q+K]	45.08	6.75	24.42	38.13	14.56	22.91	38.73
KR2-cross-encoder-Sem-SubC[Q]	44.27	7.50	26.26	40.31	15.87	25.44	39.13
KR2-cross-encoder-Sem-SubC[K]	44.95	7.29	25.98	40.24	15.53	24.48	39.70

References

[1] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, et al., A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions, ACM Transactions on Information Systems 43 (2025) 1–55.

[2] C. Wang, X. Liu, Y. Yue, X. Tang, T. Zhang, C. Jiayang, Y. Yao, W. Gao, X. Hu, Z. Qi, et al., Survey on factuality in large language models: Knowledge, retrieval and domain-specificity, arXiv preprint arXiv:2310.07521 (2023).

[3] A. Asai, S. Min, Z. Zhong, D. Chen, Acl 2023 tutorial: Retrieval-based language models and applications, ACL 2023 (2023).

[4] C.-H. Hsueh, P. K.-M. Huang, T.-H. Lin, C. W. Liao, H.-C. Fang, C.-W. Huang, Y.-N. Chen, Editing the mind of giants: An in-depth exploration of pitfalls of knowledge editing in large language models, in: Y. Al-Onaizan, M. Bansal, Y.-N. Chen (Eds.), Findings of the Association for Computational Linguistics: EMNLP 2024, Association for Computational Linguistics, Miami, Florida, USA, 2024, pp. 9417–9429. URL: https://aclanthology.org/2024.findings-emnlp.550/. doi:10.18653/v1/2024.findings-emnlp.550.

[5] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, D. Kiela, Retrieval-augmented generation for knowledge-intensive nlp tasks, in: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems, volume 33, Curran Associates, Inc., 2020, pp. 9459–9474. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf.

[6] N. Carlini, F. Tramèr, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. B. Brown, D. X. Song, Ú. Erlingsson, A. Oprea, C. Raffel, Extracting training data from large language models, in: USENIX Security Symposium, 2020.

[7] J. Baek, A. F. Aji, A. Saffari, Knowledge-augmented language model prompting for zero-shot knowledge graph question answering, in: B. Dalvi Mishra, G. Durrett, P. Jansen, D. Neves Ribeiro, J. Wei (Eds.), Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations (NLRSE), Association for Computational Linguistics, Toronto, Canada, 2023, pp. 78–106. URL: https://aclanthology.org/2023.nlrse-1.7. doi:10.18653/v1/2023.nlrse-1.7.

[8] H. Han, Y. Wang, H. Shomer, K. Guo, J. Ding, Y. Lei, M. Halappanavar, R. A. Rossi, S. Mukherjee, X. Tang, Q. He, Z. Hua, B. Long, T. Zhao, N. Shah, A. Javari, Y. Xia, J. Tang, Retrieval-augmented generation with graphs (graphrag), 2025. URL: https://arxiv.org/abs/2501.00309. arXiv:2501.00309.

[9] F. Shi, X. Chen, K. Misra, N. Scales, D. Dohan, E. H. Chi, N. Schärli, D. Zhou, Large language models can be easily distracted by irrelevant context, in: International Conference on Machine Learning, PMLR, 2023, pp. 31210–31227.

[10] Z. Liu, C. Xiong, M. Sun, Z. Liu, Entity-duet neural ranking: Understanding the role of knowledge graph semantics in neural information retrieval, in: I. Gurevych, Y. Miyao (Eds.), Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Melbourne, Australia, 2018, pp. 2395–2405. URL: https://aclanthology.org/P18-1223/. doi:10.18653/v1/P18-1223.

[11] N. Gupta, S. Singh, D. Roth, Entity linking via joint encoding of types, descriptions, and context, in: M. Palmer, R. Hwa, S. Riedel (Eds.), Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Copenhagen, Denmark, 2017, pp. 2681–2690. URL: https://aclanthology.org/D17-1284/. doi:10.18653/v1/D17-1284.

[12] X. Li, D. Roth, Learning question classifiers, in: COLING 2002: The 19th International Conference on Computational Linguistics, 2002. URL: https://aclanthology.org/C02-1150/.

[13] D. Garigliotti, F. Hasibi, K. Balog, Identifying and exploiting target entity type information for ad hoc entity retrieval, Inf. Retr. 22 (2019) 285–323. URL: https://doi.org/10.1007/s10791-018-9346-x. doi:10.1007/s10791-018-9346-x.

[14] R. Kaptein, J. Kamps, Exploiting the category structure of wikipedia for entity ranking, Artificial Intelligence 194 (2013) 111–129. URL: https://www.sciencedirect.com/science/article/pii/S0004370212000732. doi:https://doi.org/10.1016/j.artint.2012.06.003, artificial Intelli-

gence, Wikipedia and Semi-Structured Resources.

[15] P. Sen, S. Mavadia, A. Saffari, Knowledge graph-augmented language models for complex question answering, in: B. Dalvi Mishra, G. Durrett, P. Jansen, D. Neves Ribeiro, J. Wei (Eds.), Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations (NLRSE), Association for Computational Linguistics, Toronto, Canada, 2023, pp. 1–8. URL: https://aclanthology.org/2023.nlrse-1.1/. doi:10.18653/v1/2023.nlrse-1.1.

[16] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2019. URL: https://arxiv.org/abs/1908.10084.

[17] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, in: North American Chapter of the Association for Computational Linguistics, 2019.

[18] J. Lin, R. Nogueira, A. Yates, Pretrained transformers for text ranking: Bert and beyond, Springer Nature, 2022.

[19] G. Weikum, L. Dong, S. Razniewski, F. M. Suchanek, Machine knowledge: Creation and curation of comprehensive knowledge bases, Found. Trends Databases 10 (2020) 108–490. URL: https://api.semanticscholar.org/CorpusID:221879078.

[20] D. Vrandečić, M. Krötzsch, Wikidata: a free collaborative knowledgebase, Communications of the ACM 57 (2014) 78–85.

[21] J. Berant, A. Chou, R. Frostig, P. Liang, Semantic parsing on Freebase from question-answer pairs, in: D. Yarowsky, T. Baldwin, A. Korhonen, K. Livescu, S. Bethard (Eds.), Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Seattle, Washington, USA, 2013, pp. 1533–1544. URL: https://aclanthology.org/D13-1160/.

[22] W.-t. Yih, M. Richardson, C. Meek, M.-W. Chang, J. Suh, The value of semantic parse labeling for knowledge base question answering, in: K. Erk, N. A. Smith (Eds.), Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Association for Computational Linguistics, Berlin, Germany, 2016, pp. 201–206. URL: https://aclanthology.org/P16-2033/. doi:10.18653/v1/P16-2033.

[23] V. Sanh, L. Debut, J. Chaumond, T. Wolf, Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, ArXiv abs/1910.01108 (2019).

[24] P. Sen, A. F. Aji, A. Saffari, Mintaka: A complex, natural, and multilingual dataset for end-to-end question answering, in: Proceedings of the 29th International Conference on Computational Linguistics, International Committee on Computational Linguistics, Gyeongju, Republic of Korea, 2022, pp. 1604–1619. URL: https://aclanthology.org/2022.coling-1.138.

[25] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, J. Taylor, Freebase: a collaboratively created graph database for structuring human knowledge, in: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, SIGMOD '08, Association for Computing Machinery, New York, NY, USA, 2008, p. 1247–1250. URL: https://doi.org/10.1145/1376616.1376746. doi:10.1145/1376616.1376746.

[26] D. Diefenbach, T. P. Tanon, K. Singh, P. Maret, Question Answering Benchmarks for Wikidata, in: ISWC 2017, Vienne, Austria, 2017. URL: https://hal.science/hal-01637141.

[27] V. Sanh, A. Webson, C. Raffel, S. Bach, L. Sutawika, Z. Alyafeai, A. Chaffin, A. Stiegler, A. Raja, M. Dey, M. S. Bari, C. Xu, U. Thakker, S. S. Sharma, E. Szczechla, T. Kim, G. Chhablani, N. Nayak, D. Datta, J. Chang, M. T.-J. Jiang, H. Wang, M. Manica, S. Shen, Z. X. Yong, H. Pandey, R. Bawden, T. Wang, T. Neeraj, J. Rozen, A. Sharma, A. Santilli, T. Fevry, J. A. Fries, R. Teehan, T. L. Scao, S. Biderman, L. Gao, T. Wolf, A. M. Rush, Multitask prompted training enables zero-shot task generalization, in: International Conference on Learning Representations, 2022. URL: https://openreview.net/forum?id=9Vrb9D0WI4.

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

[29] J. Baek, S. Jeong, M. Kang, J. Park, S. Hwang, Knowledge-augmented language model verification, in: H. Bouamor, J. Pino, K. Bali (Eds.), Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Singapore, 2023, pp. 1720–1736. URL: https://aclanthology.org/2023.emnlp-main.107/. doi:10.18653/v1/2023.emnlp-main.107.

[30] Y. Wu, N. Hu, S. Bi, G. Qi, J. Ren, A. Xie, W. Song, Retrieve-rewrite-answer: A kg-to-text enhanced llms framework for knowledge graph question answering, ArXiv abs/2309.11206 (2023). URL: https://api.semanticscholar.org/CorpusID:262054223.

[31] H. Touvron, L. Martin, K. R. Stone, P. Albert, A. Almahairi, Y. Babaei, N. lay Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. M. Bikel, L. Blecher, C. tian Cantón Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. S. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. M. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. H. M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, T. Scialom, Llama 2: Open foundation and fine-tuned chat models, ArXiv abs/2307.09288 (2023).

[32] S. Ko, H. Cho, H. Chae, J. Yeo, D. Lee, Evidence-focused fact summarization for knowledge-augmented zero-shot question answering, 2024. URL: https://arxiv.org/abs/2403.02966. arXiv:2403.02966.

[33] Z. Jia, S. Pramanik, R. S. Roy, G. Weikum, Complex temporal question answering on knowledge graphs, Proceedings of the 30th ACM International Conference on Information & Knowledge Management (2021).

[34] X. Qian, Y. Zhang, Y. Zhao, B. Zhou, X. Sui, L. Zhang, K. Song, TimeR^4 : Time-aware retrieval-augmented large language models for temporal knowledge graph question answering, in: Y. Al-Onaizan, M. Bansal, Y.-N. Chen (Eds.), Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Miami, Florida, USA, 2024, pp. 6942–6952. URL: https://aclanthology.org/2024.emnlp-main.394. doi:10.18653/v1/2024.emnlp-main.394.

[35] W. Guo, X. Wang, J. Chen, Z. Li, Z. Chen, Ontology-enhanced knowledge graph completion using large language models, ArXiv abs/2507.20643 (2025). URL: https://api.semanticscholar.org/CorpusID:280323848.

[36] K. Song, X. Tan, T. Qin, J. Lu, T.-Y. Liu, Mpnet: masked and permuted pre-training for language understanding, in: Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS '20, Curran Associates Inc., Red Hook, NY, USA, 2020.

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

[38] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, Q. Guo, M. Wang, H. Wang, Retrieval-augmented generation for large language models: A survey, ArXiv abs/2312.10997 (2023). URL: https://api.semanticscholar.org/CorpusID:266359151.

[39] W. Fan, Y. Ding, L. bo Ning, S. Wang, H. Li, D. Yin, T.-S. Chua, Q. Li, A survey on rag meeting llms: Towards retrieval-augmented large language models, in: Knowledge Discovery and Data Mining, 2024. URL: https://api.semanticscholar.org/CorpusID:269740933.

[40] B. Peng, Y. Zhu, Y. Liu, X. Bo, H. Shi, C. Hong, Y. Zhang, S. Tang, Graph retrieval-augmented generation: A survey, arXiv preprint arXiv:2408.08921 (2024).

[41] Y. Chen, D. Röder, J.-J. Erker, L. Hennig, P. Thomas, S. Möller, R. Roller, Retrieval-augmented knowledge integration into language models: A survey, in: S. Li, M. Li, M. J. Zhang, E. Choi, M. Geva, P. Hase, H. Ji (Eds.), Proceedings of the 1st Workshop on Towards Knowledgeable Language Models (KnowLLM 2024), Association for Computational Linguistics, Bangkok, Thailand, 2024, pp. 45–63. URL: https://aclanthology.org/2024.knowllm-1.5/. doi:10.18653/v1/2024.knowllm-1.5.

[42] T. Guo, Q. Yang, C. Wang, Y. Liu, P. Li, J. Tang, D. Li, Y. Wen, Knowledgenavigator: Leveraging

large language models for enhanced reasoning over knowledge graph, Complex & Intelligent Systems 10 (2024) 7063–7076.

[43] L. Luo, Y.-F. Li, G. Haffari, S. Pan, Reasoning on graphs: Faithful and interpretable large language model reasoning, in: International Conference on Learning Representations, 2024.

[44] Y. Wen, Z. Wang, J. Sun, Mindmap: Knowledge graph prompting sparks graph of thoughts in large language models, in: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics, 2024.

[45] J. Jiang, K. Zhou, Z. Dong, K. Ye, X. Zhao, J.-R. Wen, StructGPT: A general framework for large language model to reason over structured data, in: H. Bouamor, J. Pino, K. Bali (Eds.), Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Singapore, 2023, pp. 9237–9251. URL: https://aclanthology.org/2023.emnlp-main.574. doi:10.18653/v1/2023.emnlp-main.574.

[46] Y. Lan, G. He, J. Jiang, J. Jiang, W. X. Zhao, J.-R. Wen, A survey on complex knowledge base question answering: Methods, challenges and solutions, in: Z.-H. Zhou (Ed.), Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21, International Joint Conferences on Artificial Intelligence Organization, 2021, pp. 4483–4491. URL: https://doi.org/10.24963/ijcai.2021/611. doi:10.24963/ijcai.2021/611, survey Track.

[47] M. Bockling, H. Paulheim, A. Iana, Walk&retrieve: Simple yet effective zero-shot retrieval-augmented generation via knowledge graph walks, ArXiv abs/2505.16849 (2025).

[48] Z. Gao, Y. Cao, H. Wang, A. Ke, Y. Feng, X. Xie, S. K. Zhou, Frag: A flexible modular framework for retrieval-augmented generation based on knowledge graphs, in: Annual Meeting of the Association for Computational Linguistics, 2025.

[49] Y. Cui, Z. Sun, W. Hu, Z. Fu, Kgfr: A foundation retriever for generalized knowledge graph question answering, ArXiv abs/2511.04093 (2025). URL: https://api.semanticscholar.org/CorpusID:282812675.

[50] Z. Liu, H. Sack, G. A. Gesese, Hyp-kgrag: Hypothetical path-based knowledge graph retrieval augmented generation with deepseek, in: Proceedings of the Second International Workshop on Retrieval-Augmented Generation Enabled by Knowledge Graphs (RAGE-KG 2025) co-located with the 24th International Semantic Web Conference (ISWC 2025), November 2–6, 2025, Nara, Japan, CEUR Workshop Proceedings, CEUR-WS.org, 2025.

[51] M. Li, S. Miao, P. Li, Simple is effective: The roles of graphs and large language models in knowledge-graph-based retrieval-augmented generation, in: International Conference on Learning Representations, 2025.

[52] Y. Wu, Y. Huang, N. Hu, Y. Hua, G. Qi, J. Chen, J. Z. Pan, CoTKR: Chain-of-thought enhanced knowledge rewriting for complex knowledge graph question answering, in: Y. Al-Onaizan, M. Bansal, Y.-N. Chen (Eds.), Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Miami, Florida, USA, 2024, pp. 3501–3520. URL: https://aclanthology.org/2024.emnlp-main.205/. doi:10.18653/v1/2024.emnlp-main.205.

[53] R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, C. Finn, Direct preference optimization: Your language model is secretly a reward model, ArXiv abs/2305.18290 (2023). URL: https://api.semanticscholar.org/CorpusID:258959321.

[54] Y. Wang, M. Hu, Z. Huang, D. Li, D. Yang, X. Lu, KC-GenRe: A knowledge-constrained generative re-ranking method based on large language models for knowledge graph completion, in: N. Calzolari, M.-Y. Kan, V. Hoste, A. Lenci, S. Sakti, N. Xue (Eds.), Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024), ELRA and ICCL, Torino, Italia, 2024, pp. 9668–9680. URL: https://aclanthology.org/2024.lrec-main.845/.

[55] S. Li, Y. Gao, H. Jiang, Q. Yin, Z. Li, X. Yan, C. Zhang, B. Yin, Graph reasoning for question answering with triplet retrieval, in: A. Rogers, J. Boyd-Graber, N. Okazaki (Eds.), Findings of the Association for Computational Linguistics: ACL 2023, Association for Computational Linguistics, Toronto, Canada, 2023, pp. 3366–3375. URL: https://aclanthology.org/2023.findings-acl.208/. doi:10.18653/v1/2023.findings-acl.208.

[56] X. Jiang, R. Zhang, Y. Xu, R. Qiu, Y. Fang, Z. Wang, J. Tang, H. Ding, X. Chu, J. Zhao, et al., Hykge: A hypothesis knowledge graph enhanced framework for accurate and reliable medical llms responses, arXiv preprint arXiv:2312.15883 (2023).

[57] R. Yang, H. Liu, E. Marrese-Taylor, Q. Zeng, Y. H. Ke, W. Li, L. Cheng, Q. Chen, J. Caverlee, Y. Matsuo, et al., Kg-rank: Enhancing large language models for medical qa with knowledge graphs and ranking techniques, arXiv preprint arXiv:2403.05881 (2024).

[58] Q. Jin, W. Kim, Q. Chen, D. C. Comeau, L. Yeganova, W. J. Wilbur, Z. Lu, Medcpt: Contrastive pre-trained transformers with large-scale pubmed search logs for zero-shot biomedical information retrieval, Bioinformatics 39 (2023) btad651.

[59] Y. Gu, R. Tinn, H. Cheng, M. Lucas, N. Usuyama, X. Liu, T. Naumann, J. Gao, H. Poon, Domain-specific language model pretraining for biomedical natural language processing, ACM Transactions on Computing for Healthcare (HEALTH) 3 (2021) 1–23.

[60] X. Dai, Y. Hua, T. Wu, Y. Sheng, Q. Ji, G. Qi, Large language models can better understand knowledge graphs than we thought, Knowledge-Based Systems 312 (2025) 113060. URL: https://www.sciencedirect.com/science/article/pii/S0950705125001078. doi:https://doi.org/10.1016/j.knosys.2025.113060.

[61] K. Sun, N. P. Jedema, K. Sharma, R. Janssen, J. Pujara, P. Szekely, A. Moschitti, Efficient and accurate contextual re-ranking for knowledge graph question answering, in: N. Calzolari, M.-Y. Kan, V. Hoste, A. Lenci, S. Sakti, N. Xue (Eds.), Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024), ELRA and ICCL, Torino, Italia, 2024, pp. 5585–5595. URL: https://aclanthology.org/2024.lrec-main.496.

[62] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, W. E. Sayed, Mistral 7b, 2023. arXiv:2310.06825.