

Interpretable Question Answering with Knowledge Graphs

Manasvi Srivastava

Center of Artificial Intelligence, Banasthali Vidyapith, Rajasthan ,India
manasvisrivastava26@gmail.com

Abstract. This paper presents a question-answering system that operates exclusively on a knowledge graph without relying on retrieval-augmented generation (RAG) with large language models (LLMs). Instead, lightweight paraphraser is used to paraphrase entity-relationship edges retrieved from the knowledge graph. The pipeline consists of two main stages. First, a document is pre-processed to generate question-answer (QA) pairs. Second, these QAs are converted into a knowledge graph, from which graph-based retrieval is performed using embeddings and fuzzy techniques. The graph is queried, re-ranked, and paraphrased to generate the final answer. Evaluation using LLM-as-judge on the CRAG benchmark achieved accuracies of 71.9% and 54.4% with LLAMA-3.2 and GPT-3.5-Turbo, respectively.

The principal contribution is a fully automatic knowledge-graph QA pipeline requiring no hand-crafted semantic parser or query-language generation (e.g., SPARQL or Cypher), combining node-level embedding retrieval, type-level generalization, and fuzzy entity matching to answer questions directly from an automatically constructed graph. We further quantify the reliability of the two LLM judges through inter-rater agreement (Cohen's $\kappa = 0.35$–0.40) and paired significance testing, and report a manual failure-case taxonomy. A direct comparison against a dense-retrieval RAG baseline remains a limitation.

Keywords: Knowledge Graph · Question Answering · Retrieval Augmented Generation

1 Introduction

Question Answering (QA) systems aim to provide precise and contextually relevant answers using structured or unstructured knowledge sources. Recently, Retrieval-Augmented Generation (RAG) [1] has gained attention, where user queries retrieve relevant text chunks passed to an LLM for answer generation. However, these systems rely heavily on unstructured documents, are prone to hallucination [2], [3], and offer limited transparency.

In contrast, Knowledge Graphs (KGs) represent information through entities (nodes) and relationships (edges) and can be constructed from structured or unstructured data. This structure supports interpretable reasoning and contextual association, making KGs suitable for knowledge-intensive QA. However,

© The Author(s) 2026
A. Shastri et al. (eds.), *Proceedings of the Fourth Workshop on Applications of Computational Intelligence and Interactive Systems (WACIIS 2026)*, Advances in Intelligent Systems Research 217,
https://doi.org/10.2991/978-94-6239-832-0_13

traditional KG-based QA systems often require complex semantic parsing and rule-based components, limiting scalability and adaptability.

Rather than relying on chunk-based retrieval as in standard RAG pipelines, the system described here builds a structured graph directly from generated QA pairs and queries this graph to answer new questions. This departs from dense-retrieval RAG frameworks that operate on raw text chunks [4]; instead, the document's content is abstracted into a graph that supports both semantic similarity search and entity-level lookup.

The overall approach is organised into two stages.

In the first stage, a prompt-driven language model generates QA pairs from the source document; these pairs are then processed by `LangChain's LLMGraphTransformer (backed by GPT-3.5-Turbo)` to extract entities and relations, which are assembled into a knowledge graph. In the second stage, incoming queries are matched against this graph through a combination of embedding similarity and fuzzy string matching to pull out the relevant subgraph. The extracted triples are then rendered into natural language by a lightweight paraphrasing model `(tuner007/pegasus_paraphrase)`. Answer quality is finally scored through an LLM-as-a-judge protocol.

Because every answer can be traced back to a specific retrieved triple before being paraphrased, this design offers a more auditable alternative to conventional RAG pipelines – a property that matters most in domains such as law or engineering, where factual consistency and traceability are non-negotiable. The contributions are threefold: (1) a fully automated pipeline that converts open-domain QA pairs into a queryable knowledge graph without manual schema design or logical query generation; (2) a three-stage retrieval strategy – node-level embedding similarity, type-level generalization, and fuzzy entity matching – that supports both instance- and category-level questions; and (3) an evaluation protocol that extends beyond a single LLM-as-judge accuracy score by reporting inter-judge reliability, paired significance testing, and a manual error taxonomy on a domain-specific PDF corpus and the CRAG benchmark.

Table 1. QA Graph-Based Answering Flow

Component	**Description**
Document Input	PDF containing contractual clauses
QA Pairs	Extracted using prompt-based QA generation
Graph Creation	GPT-3.5 identifies entities, relations
Embeddings	`text-embedding-3-large` on nodes and types
Retrieval	Cosine and fuzzy search on entities
Reranker	`BAAI/bge-reranker-large` on triples
Answer	`tuner007/pegasus_paraphrase` paraphrasing

2 Related Work

This section reviews related work on knowledge graph-based question answering without LLM-based RAG. Related work can be categorised into two approaches: (i) semantic parsing that converts questions into logical queries, e.g., Cypher for Neo4j, and executes them on the KG, and (ii) retrieval/embedding methods that find relevant subgraphs, entities, and paths without constructing full logical queries.

Gu et al. [5] reviewed question answering over knowledge bases for semantic parsing and proposed joint entity linking and parsing. Most systems target either SPARQL, a query language for data stored in Resource Description Framework (RDF) format, or Cypher for property graph databases. Du et al. [6] proposed a dual relation graph where relations are nodes connected when sharing head or tail entities. Their approach alternates between (1) reasoning on the primal entity graph, (2) propagating information on the dual relation graph, and (3) interaction between both graphs.

Zhao et al. [7] proposed linking relation phrases to relation paths rather than single relations. Their path ranking model aligns textual and structural information using gated attention and paraphrase dictionaries to handle vague phrases. Yang et al. [8] highlighted RAG limitations, showing that advanced LLMs with RAG achieve only 44–63% accuracy on dynamic, diverse questions. Although CRAG benchmarks RAG, our KG-only pipeline achieves competitive accuracy.

GraphRAG-style systems typically feed retrieved subgraphs into an LLM to produce the final answer [4], while other approaches apply embedding- or path-ranking techniques directly on pre-existing knowledge graphs [6]. The pipeline proposed here differs on both counts: the graph itself is built from automatically generated QA pairs rather than assumed to exist beforehand, and answer generation is handled by a small, non-LLM paraphraser instead of a full language model. This sacrifices some fluency in exchange for a fully traceable chain from a retrieved triple to the final answer text.

Overall, the pipeline ties together four components − graph construction from QA pairs, embedding-based retrieval, fuzzy entity matching, and reranking followed by paraphrasing − with the explicit goal of keeping the reasoning path visible at every step, in contrast to the opacity typical of black-box RAG systems, and of limiting the opportunities for hallucinated content to enter the final answer.

3 Research Methodology

3.1 QA Generation from Document

Two data sources were used for evaluation: a domain-specific PDF document and Tasks 1–2 of the CRAG [8] benchmark, which together provide 2,706 question-answer pairs. For the PDF-based experiments, the document was first broken into coherent segments − guided by paragraph boundaries, bullet points, clause

markers, and a minimum/maximum length rule − so that each segment could function as a self-contained unit of knowledge for downstream indexing. These segments serve as atomic knowledge blocks for indexing and retrieval. For each chunk, the Hugging Face model `iarfmoose/t5-base-question-generator` generates relevant QA pairs to extract atomic facts, stored in JSON format for further processing. The `t5-base-question-generator` and `LLMGraphTransformer` components were chosen over larger alternatives because they require no task-specific fine-tuning and reduce compute costs compared to full LLM-based RAG generation; Section 4.3 discusses the accuracy trade-offs.

3.2 Knowledge Graph Creation

This section describes the methodology for constructing a Knowledge Graph from question-answer (QA) pairs and the subsequent process of querying it for relevant information. Given an input question, the system identifies and retrieves the associated entities and relationships from the graph. These retrieved entities and relationships are then fed into a lightweight paraphrasing model, which synthesizes the final answer in natural language. Table 1 presents a high-level overview of the Knowledge Graph-based answer generation workflow.

Knowledge Graph Construction Sets of question-answer (QA) pairs are processed through the LLMGraphTransformer of LangChain, utilizing the GPT-3.5 Turbo language model to extract entities and their semantic relationships. The additional information to consider Clauses and Numerical Values as nodes was passed to the LLMGraphTransformer as a prompt shown in Fig. 1. For this research, sets of twenty QA pairs are passed to LLMGraphTransformer at a time.

```
prompt = "Clauses␣and␣Numerical␣Values,␣etc␣count␣as␣nodes"
llm_transformer = LLMGraphTransformer(llm=llm, additional_instructions=
    prompt)
```

Fig. 1. LLM Graph Transformer setup

These nodes and relationships form a Knowledge Graph stored in a Neo4j graph database. After processing the QA pairs, the graph represents the document's knowledge in structured form, with nodes as entities and edges as relationships. We created two knowledge graphs, one for each dataset.

We generated embeddings for each node and node type using transformer-based sentence embeddings with the `text-embedding-3-large` model. Neo4j computes and stores these embeddings internally. Node-type embeddings are also stored, as individual nodes represent specific instances (e.g., people) and may be less informative for general or class-level questions about groups or categories.

3.3 Retrieval Phase

The retrieval phase consists of two key steps:

Node-Level Semantic Matching: We compute the cosine similarity between the input question and candidate node embeddings in the Knowledge Graph. Nodes with the highest scores are selected along with their immediate relationships.

Type-Level Generalization Retrieval: We identify the node type most semantically similar to the input question using type embeddings. All nodes and relationships associated with the top-matching type are then selected. This enables generalized knowledge retrieval, especially for questions referring to categories (e.g., "scientists") rather than specific instances.

Fuzzy Entity Matching: Finally, we identify entities mentioned in the input question using the lightweight Hugging Face model `dslim/bert-base-NER`. Entity mentions are matched against graph nodes with Lucene's fuzzy operator set to an edit distance of two − a middle ground that, in practice, catches more true matches than exact string matching while producing fewer spurious matches than a looser edit-distance threshold would.

For node-level retrieval, the five highest-scoring nodes ($k = 5$) by cosine similarity are kept. A minimum-score cutoff is then applied during reranking, but at least four candidates are always carried forward even when their scores fall below this threshold, ensuring the paraphraser still has enough surrounding context to work with in low-similarity cases.

3.4 Paraphrase Phase

Once the relevant nodes and edges are retrieved, they are handed to a small paraphrasing model − `tuner007/pegasus_paraphrase` − which turns the raw triples into a single fluent answer. Preliminary experiments indicate that swapping in a more capable, better-trained paraphraser would likely push performance higher still.

3.5 Reranker Phase

Candidate paraphrased answers are then scored for semantic relevance to the original question using `BAAI/bge-reranker-large`, a HuggingFace reranking model. Running this reranker before rather than after paraphrasing was also tried, but it made no meaningful difference to accuracy. The five top-ranked answers are ultimately retained for evaluation.

4 Evaluation Methodology And Results

The remainder of this section reports results from an automated evaluation protocol in which an LLM serves as the judge of answer correctness.

4.1 LLM-as-a-Judge Evaluation

We employed Llama-3.2 and GPT-3.5 Turbo as automatic evaluators to assess answer correctness, as shown in Fig. 2. For each question used to construct the Knowledge Graph, the LLM checked whether the ground-truth answer appeared among the top five candidates. A lenient prompt was used because the lightweight paraphraser can produce broken English. The system achieved 89.6% accuracy with Llama-3.2 and 78.3% with gpt-3.5-turbo on the PDF dataset, indicating effective knowledge graph retrieval.

For CRAG, 308 of 2706 QA pairs were marked as "invalid question," leaving 2398 valid questions. Accuracy was 71.9% with Llama-3.2 and 54.4% with GPT-3.5-turbo, compared to the reported 63% [8]. Section 4.2 quantifies judge reliability and provides a manual taxonomy of failure cases.

```
prompt = f"""
You are a generous evaluator. Determine whether the **predicted answer
    reflects the intended meaning** of the expected answer, even when
    wording differs or the match is partial.

* Be lenient: accept paraphrases, synonyms, and generalizations.
* The predicted answer need not include every detail; focus on the key
    idea.
* If the main meaning is clearly or recognizably present, even in
    different words, consider it a match.
* Do not accept answers that are vague or only weakly related.
* Ignore unrelated or repetitive sentences.

Query:
{row[queryCol]}

Expected Answer (main idea):
{row['expected']}

Predicted Answer:
{row[predictedCol]}

Does the predicted answer contain the **main idea or intent** of the
    expected answer, directly, indirectly, partially, or through
    paraphrasing?

Respond only with "Yes" or "No".
"""
```

Fig. 2. Prompt used for LLM-as-a-judge

Table 2. Accuracy of system judged by different LLMs

Questions Asked	**Llama 3.2 Accuracy**	**GPT-3.5 Turbo Accuracy**
QAs generated at runtime using Hugging Face's model (set of 194)	89.6%	78.3%
QAs generated at runtime using Hugging Face's model and later perturbed (set of 194)	85.6%	72.6%
QAs from standard CRAG dataset, all pairs incl. invalid (n=2706)	68.6%	50.1%
QAs from standard CRAG dataset, valid subset only (n=2398)	71.9%	54.4%

4.2 Inter-Judge Reliability and Failure Analysis

To assess how much the reported accuracies depend on which judge is used, we computed raw agreement and Cohen's κ between Llama-3.2 and GPT-3.5-Turbo verdicts across all three evaluation conditions (Table 3, Fig. 3, Fig. 4).

Table 3. Accuracy and inter-judge reliability summary

Dataset	**n**	**Llama-3.2 acc.**	**GPT-3.5 acc.**	**Raw agreement**	κ
PDF	194	89.7%	78.4%	82.5%	0.36
PDF, perturbed	194	85.6%	72.7%	77.8%	0.35
CRAG, valid subset	2398	71.9%	54.4%	71.1%	0.40

Table 5 and Fig. 5 break this down further: on the PDF dataset the two judges disagree on 34/194 items (17.5%), almost entirely GPT-3.5 marking an answer wrong where Llama-3.2 marked it right; on CRAG, disagreement is more symmetric (692/2398, 28.9%).

To test whether these accuracy differences are statistically significant rather than judge-leniency noise, we computed 95% Wilson score confidence intervals for each accuracy figure and ran McNemar's test on the paired judge verdicts (Table 4). In all three conditions the difference between Llama-3.2 and GPT-3.5 accuracy is highly significant ($p < 0.001$), confirming Llama-3.2 is a measurably more lenient judge rather than the two being interchangeable.

Manual Failure Taxonomy We manually reviewed every case where both judges agreed the answer was incorrect: all 14 such cases on the PDF dataset, and a random sample of 50 of the 538 such cases on CRAG. Fig. 6 and Fig. 7

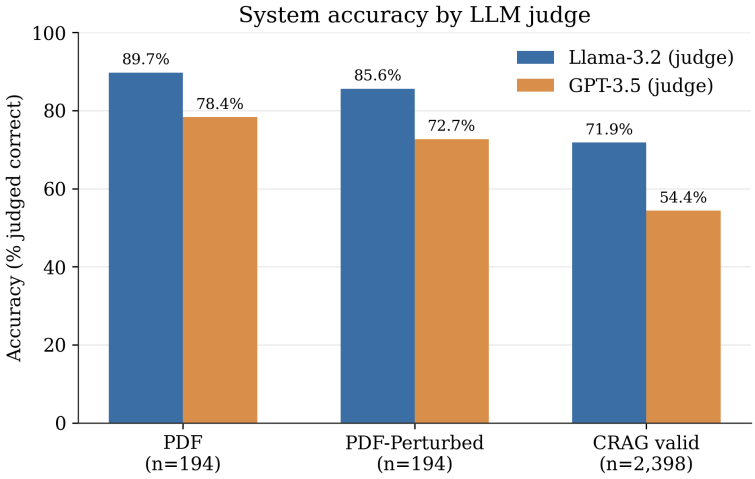

Fig. 3. System accuracy by LLM judge, across all three evaluation conditions.

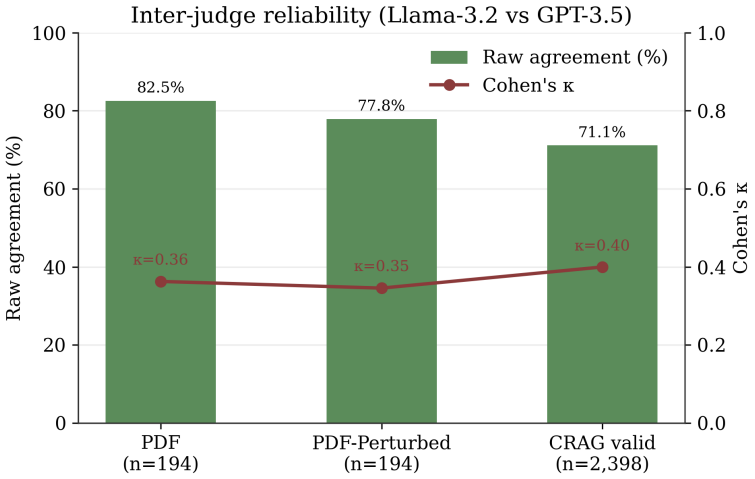

Fig. 4. Raw inter-judge agreement (bars) and Cohen's κ (line). Agreement is fair-to-moderate (κ = 0.35–0.40 per Landis and Koch), indicating the two judges broadly agree on correctness but diverge on a non-trivial share of borderline cases.

Table 4. Paired significance testing between judges (McNemar's test, 95% Wilson CIs)

Dataset	n	Llama acc. [95% CI]	GPT acc. [95% CI]	McNemar	p
PDF	194	89.7% [84.6, 93.2]	78.4% [72.0, 83.6]	12.97	3.2×10^{-4}
PDF, perturbed	194	85.6% [79.9, 89.8]	72.7% [66.0, 78.5]	13.40	2.5×10^{-4}
CRAG, valid	2398	71.9% [70.0, 73.6]	54.4% [52.4, 56.4]	251.28	1.4×10^{-56}

Table 5. Judge verdict confusion matrices

Dataset	Llama \ GPT	Yes	No
PDF (n=194)	Yes	146 (75.3%)	28 (14.4%)
	No	6 (3.1%)	14 (7.2%)
CRAG valid (n=2398)	Yes	1168 (48.7%)	555 (23.1%)
	No	137 (5.7%)	538 (22.4%)

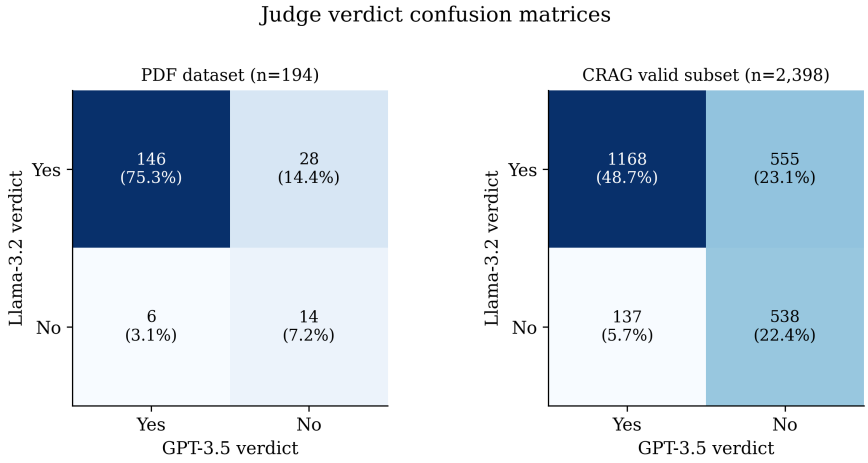

Fig. 5. Confusion matrices between the two judges' verdicts on the PDF dataset and the CRAG valid subset.

summarize the resulting categories (CRAG categories can overlap within a single failure).

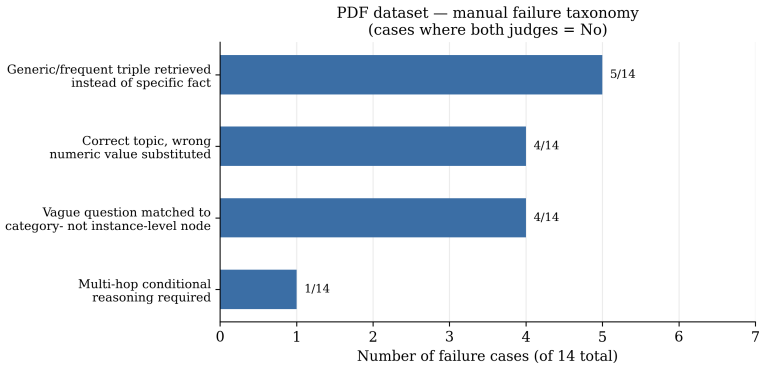

Fig. 6. PDF dataset manual failure taxonomy (all 14 both-judges-No cases reviewed).

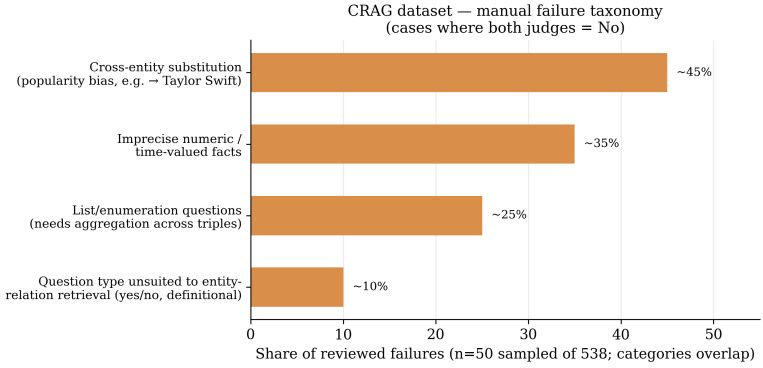

Fig. 7. CRAG dataset manual failure taxonomy (50 of 538 both-judges-No cases sampled and reviewed).

The dominant CRAG failure mode is cross-entity substitution driven by popularity bias in the embedding space: queries about lower-frequency entities (e.g., Toby Keith, Radiohead, The Weeknd) sometimes retrieve facts about a high-frequency node instead. Fig. 8 shows a representative case together with a PDF-dataset failure where the correct clause is retrieved but a numeric value from a neighboring clause is substituted.

```
# PDF dataset - both judges = No
query: How long does it take to carry out a snagging job?
expected: 18 months
predicted: The employer's response time is 20 working days...
llama_eval: No gpt_eval: No

# CRAG dataset - both judges = No
query: how many times has taylor swift won the grammy
          award for album of the year?
expected: four times, for albums "fearless", "1989",
          "folklore", and "midnights"
predicted: Taylor Swift won 14 awards. Is Taylor Swift
          a winner?...
llama_eval: No gpt_eval: No
```

Fig. 8. Representative prediction records, taken verbatim from the evaluation logs

4.3 Robustness Evaluation via Question Perturbation

To assess the robustness and generalisation capabilities of the system, the questions were perturbed while preserving their original intent using Llama 3.2. The modified questions were then re-evaluated to determine whether the system could still retrieve correct answers. The accuracy of the perturbed questions on the PDF dataset is 85.6% with Llama-3.2 and 72.6% with gpt-3.5-turbo. One of the examples is shown in Fig. 9:

```
Original Question: Which of the following is NOT an Employer's Risk under
     Clause 3.1?
Answer: Contractor's Negligence is not the employer's risk.

Perturbed Question: Which is not the responsibility of the Employer
    according to Clause 3.1?
Answer: Contractor's Negligence is not the employer's risk. (Correct)
```

Fig. 9. Perturbation Sample

4.4 Limitations

This evaluation does not directly compare against a dense-retrieval RAG baseline on the same PDF and CRAG subsets, as no such run exists in the current setup. [8] reports 44–63% accuracy for RAG systems on CRAG under a stricter protocol; our 71.9%/54.4% is favorable but not strictly comparable due to the

more lenient judging prompt. A second limitation is that both automatic judges are LLMs prompted for leniency, with Section 4.2 showing systematically higher scores from Llama-3.2 than GPT-3.5. Human evaluation on a subset would provide a stronger accuracy estimate. A controlled RAG baseline and partial human evaluation are the key additions for future work.

5 Conclusion

Because it returns ranked answers that are at least partially correct or contextually relevant, the system lends itself well to exploratory question answering, entity discovery, and knowledge validation tasks. Planned extensions include fine-tuning the QA-pair generator, improving relation extraction, running a controlled head-to-head comparison against dense-retrieval RAG under identical judging conditions, and adding partial human review to check the LLM judges' verdicts.

Code covering the experiments, methodology, and dataset used in this paper has been released at
`https://github.com/grimreapermanasvi/GraphRAG`.

References

1. P. Lewis, E. Perez, E. Piktus, E. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel et al.: Retrieval-augmented generation for knowledge-intensive NLP tasks. Advances in Neural Information Processing Systems, vol. 33, pp. 9459–9474, (2020).
2. Z. Xu, S. Jain, and M. Kankanhalli: Hallucination is inevitable: An innate limitation of large language models. arXiv preprint arXiv:2401.11817, (2024).
3. Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung: Survey of hallucination in natural language generation. ACM Computing Surveys, vol. 55, no. 12, pp. 1–38 (2023).
4. B. Peng, Y. Zhu, Y. Liu, X. Bo, H. Shi, C. Hong, Y. Zhang, and S. Tang: Graph retrieval-augmented generation: A survey. arXiv preprint arXiv:2408.08921 (2024).
5. Y. Gu, V. Pahuja, G. Cheng, and Y. Su: Knowledge base question answering: A semantic parsing perspective. arXiv preprint arXiv:2209.04994 (2022).
6. H. Du, Q. Huang, C. Li, C. Zhang, Y. Li, and D. Zhao: Relation-aware question answering for heterogeneous knowledge graphs. arXiv preprint arXiv:2312.11922 (2023).
7. Y. Zhao, J. Huang, W. Hu, Q. Chen, X. Qiu, C. Huo, and W. Ren: Implicit relation linking for question answering over knowledge graph. In Findings of the Association for Computational Linguistics: ACL 2022, pp. 3956–3968 (2022).
8. X. Yang, K. Sun, H. Xin, Y. Sun, N. Bhalla, X. Chen, S. Choudhary, R. Gui, Z. Jiang, Z. Jiang et al.: CRAG-comprehensive RAG benchmark. Advances in Neural Information Processing Systems, vol. 37, pp. 10470–10490 (2024).

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (http://creativecommons.org/licenses/by-nc/4.0/), which permits any noncommercial use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

