DO-RAG: A Domain-Specific QA Framework Using Knowledge Graph-Enhanced Retrieval-Augmented Generation

David Osei Opoku*, Ming Sheng†, Yong Zhang‡
*Department of Computer Science and Technology, Tsinghua University, Beijing, China
†Beijing National Research Center for Information Science and Technology - Tsinghua University, Beijing, China

***Abstract*—Domain-specific QA systems require not just generative fluency but high factual accuracy grounded in structured expert knowledge. While recent Retrieval-Augmented Generation (RAG) frameworks improve context recall, they struggle with integrating heterogeneous data and maintaining reasoning consistency. To address these challenges, we propose DO-RAG, a scalable and customizable hybrid QA framework that integrates multi-level knowledge graph construction with semantic vector retrieval. Our system employs a novel agentic chain-of-thought architecture to extract structured relationships from unstructured, multimodal documents, constructing dynamic knowledge graphs that enhance retrieval precision. At query time, DO-RAG fuses graph and vector retrieval results to generate context-aware responses, followed by hallucination mitigation via grounded refinement. Experimental evaluations in the database and electrical domains show near-perfect recall and over 94% answer relevancy, with DO-RAG outperforming baseline frameworks by up to 33.38%. By combining traceability, adaptability, and performance efficiency, DO-RAG offers a reliable foundation for multi-domain, high-precision QA at scale.**

***Index Terms*—Vertical Domain QA, Large Language Model, Knowledge Graph, Retrieval-Augmented Generation, Technical Knowledge Retrieval**

I. Introduction

Question-answering (QA) systems enable users to retrieve information from large corpora by posing natural-language queries. They fall into two broad categories: open-domain QA, which draws on general knowledge to answer questions, and closed-domain QA, which relies on specialized, often technical, resources to address queries. Closed-domain scenarios expose the limitations of generic models, which must be augmented with structured expert knowledge to deliver precise, reliable answers.

Recent advances in large language models (LLMs) such as DeepSeek-R1 [1], Grok 3 [2], QwQ-32B [3], and OpenAI's O3 [4] have dramatically improved fluency and contextual understanding. Yet these models depend predominantly on parametric knowledge and can hallucinate or provide imprecise responses when confronted with domain-specific terminology or complex multi-step reasoning [5], [6].

To address these limitations, Retrieval-Augmented Generation (RAG) has emerged as a popular approach for improving factual consistency by retrieving relevant text snippets before generation [7]–[9]. In parallel, knowledge graphs (KGs) offer a complementary solution by encoding entities and their relations in a structured form, facilitating multi-hop reasoning and precise context assembly [10], [11].

However, existing RAG frameworks struggle to capture the intricate relationships among entities in technical manuals and multimodal resources, leading to fragmented retrieval outputs and persistent hallucinations [12]–[14]. When retrieval and generation components remain loosely coupled, there is no guarantee that the final answer faithfully reflects the retrieved evidence. Although knowledge graphs promise more structured context, manually constructing and maintaining high-quality, domain-specific graphs is labor-intensive, and marrying them with vector search and LLM prompting adds substantial engineering overhead. Consequently, current KG-RAG hybrids face scalability bottlenecks and require extensive manual tuning to remain robust as knowledge evolves.

In this paper, we introduce DO-RAG, a **Do**main-specific **R**etrieval-**A**ugmented **G**eneration framework that end-to-end automates the transformation of unstructured, multimodal documents into a dynamic, multi-level knowledge graph via an agentic chain-of-thought extraction pipeline; tightly fuses the resulting graph traversal with semantic vector search to assemble a context-rich prompt; and then grounds generation in retrieved evidence through a refinement step that detects and corrects hallucinations.

Our contributions are as follows:

- **Agentic, multi-stage KG construction.** We design and implement a hierarchical, agent-based extraction pipeline that processes text, tables, code snippets, and images to automatically build and update a knowledge graph capturing entities, relations, and attributes.
- **Hybrid retrieval fusion.** We develop a unified mechanism that merges graph-based traversal with semantic search at query time, ensuring that all relevant, structurally grounded information informs the LLMs prompt.
- **Grounded hallucination mitigation.** We introduce a post-generation refinement step that cross-verifies initial LLM outputs against the knowledge graph and iteratively corrects inconsistencies, dramatically reducing factual errors.

* Yong Zhang (zhangyong05@tsinghua.edu.cn) is the corresponding author.

- **Plug-and-play modularity.** Our framework accommodates diverse LLMs and retrieval modules, enabling seamless component swapping and straightforward extension to new domains without retraining.

Evaluated on expert-curated benchmarks in the database and electrical engineering domains, DO-RAG achieves near-perfect contextual recall and over 94% answer relevancy, outperforming existing RAG platforms such as FastGPT [15], TiDB.AI [16] and Dify.AI [17] by up to 33.38%. By unifying structured and generative approaches, DO-RAG provides a scalable, high-precision solution for domain-specific QA.

The rest of this paper is organized as follows: Section II reviews existing work on domain-specific QA and RAG frameworks. Section III explains the design and approach of DO-RAG. Section IV presents the experimental setup, datasets, metrics, and results, comparing DO-RAG to baselines, and ends with a discussion of limitations and future work. Section V summarizes the main findings and contributions.

II. Preliminary and Related Work

A. Preliminary

In domain-specific question-answering, a system retrieves answers from a document corpus $\mathcal{D}$ and a knowledge graph $\mathcal{G} = (V, E, W)$, where V is a set of entities, E denotes relationships, and $W : E \to [0, 1]$ assigns confidence weights. A query q is mapped to a vector:

$$Q = E(q), \quad Q \in \mathbb{R}^d, \tag{1}$$

where E is an embedding function, and d is the embedding dimension. Document chunks $c_i \in \mathcal{D}$ are similarly embedded:

$$C_i = E(c_i), \quad C_i \in \mathbb{R}^d. \tag{2}$$

The retrieval score S combines semantic and graph-based relevance:

$$S = \alpha \cdot \max_i \text{sim}(Q, C_i) + (1 - \alpha) \cdot R(\mathcal{G}_Q), \tag{3}$$

where $\text{sim}(Q, C_i)$ is cosine similarity, $R(\mathcal{G}_Q)$ retrieves relevant entities from $\mathcal{G}$, and $\alpha \in [0, 1]$ balances contributions. The answer is generated as:

$$A = G(Q, S), \tag{4}$$

where G is the generation function. The goal is to minimize factual errors:

$$\min_\theta \mathbb{E}_{(q,a^*)\sim\mathcal{D}_{\text{eval}}} \mathcal{L}(G(Q, S; \theta), a^*), \tag{5}$$

with θ as model parameters, $\mathcal{D}_{\text{eval}}$ as the evaluation dataset with ground-truth answers a^*, and $\mathcal{L}$ as the loss function.

B. Related Work

Domain-specific QA systems have evolved from rule-based approaches to advanced methods integrating structured knowledge and generative models. Early systems relied on static rules and limited knowledge bases, restricting their adaptability to complex queries [18]. Knowledge graphs address this by encoding entities and relationships, enhancing reasoning, but their manual construction is resource-intensive [19].

Large language models excel in natural language tasks but struggle in specialized domains due to hallucinations and insufficient domain knowledge [20]. Retrieval-augmented Generation mitigates this by grounding responses in retrieved documents [21], yet faces challenges in capturing intricate relationships in technical contexts [7], [22].

Current frameworks like KIMedQA [13] and Panda [23] are domain-specific but lack scalability and flexibility. KIMedQA achieves high medical QA accuracy but is computationally intensive, while Panda, focused on database debugging, does not support multi-domain applications. KG-RAG [24] integrates biomedical KGs with RAG but struggles with scalability.

To overcome these limitations, we propose DO-RAG, a framework that automates dynamic KG construction, fuses vector retrieval with grounded generation, and enhances precision and adaptability in domain-specific QA.

III. Approach

A. System Overview

As illustrated in Figure 1, DO-RAG comprises four key stages: (1) multimodal document ingestion and chunking, (2) multi-level entity-relation extraction for knowledge graph (KG) construction, (3) hybrid retrieval combining graph traversal and dense vector search, and (4) a multi-stage generation pipeline for grounded, user-aligned answers.

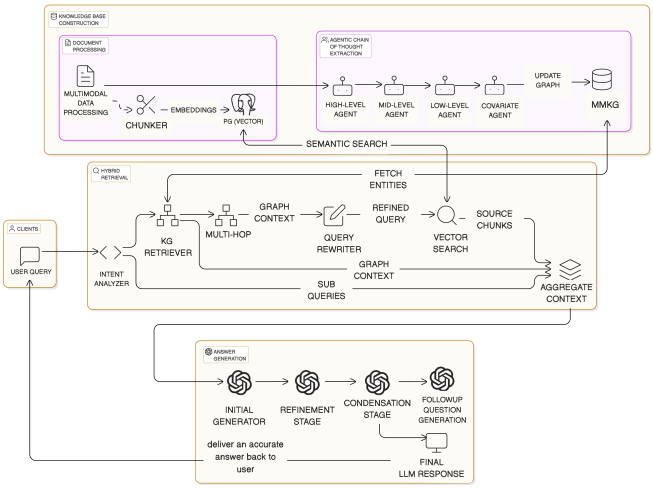


Fig. 1: High-level overview of DO-RAG methodology.

It begins by parsing heterogeneous domain data—such as logs, technical manuals, diagrams, and specifications—into meaningful chunk units. These chunks are stored alongside their vector embeddings in a pgvector-enabled PostgreSQL instance. Simultaneously, an agentic chain-of-thought entity extraction pipeline transforms document content into a structured multimodal knowledge graph (MMKG), capturing multi-granular relationships such as system parameters, behaviors, and dependencies.

At the point of user interaction, the user query undergoes decomposition through an intent recognition module, which segments it into one or more sub-queries representing discrete informational intents. The system first performs retrieval from the knowledge graph by embedding the initial query and matching it to relevant entities. This is followed by a multi-hop traversal to expand the retrieval scope, yielding structured, domain-specific context grounded in entity relationships and metadata.

Next, this graph-derived context is used to rewrite the original query into a more specific and disambiguated form using a graph-aware prompt template. The refined query is then encoded into a dense vector and used to retrieve top-k semantically similar chunks from the vector database.

At this point, the system consolidates all relevant information sources: the original user query, its refined version, the knowledge graph context, the retrieved text chunks, and prior user interaction history. These components are integrated into a unified prompt structure and passed into the generation pipeline.

The generation proceeds in three stages: initial answer generation, refinement for factual consistency and clarity, and condensation to ensure coherence and brevity. Finally, DO-RAG invokes a follow-up question generator, which formulates next-step queries based on the refined answer and overall conversation context, enhancing user engagement and supporting multi-turn interaction.

B. Knowledge Base Construction

Document processing begins with multimodal ingestion, where text, tables, and images are normalized and segmented into coherent chunks. Metadata—including source file structure, section hierarchy, and layout tagsis retained for traceability.

In parallel, we extract structured knowledge using a multi-agent, multi-level pipeline. As shown in Figure 2, the pipeline includes four specialized agents operating at different abstraction levels:

- **High-Level Agent**: Identifies structural elements (e.g., chapters, sections, paragraphs).
- **Mid-Level Agent**: Extracts domain-specific entities such as system components, APIs, and parameters.
- **Low-Level Agent**: Captures fine-grained operational relationships like thread behavior or error propagation.
- **Covariate Agent**: Attaches attributes (e.g., default values, performance impact) to existing nodes.

The output is a dynamic KG $\mathcal{G} = (V, E, W)$, where nodes V represent entities, edges E represent relationships, and weights W encode confidence scores. To avoid redundancy, deduplication is enforced using cosine similarity between new and existing entity embeddings. Additionally, synopsis nodes are synthesized to group similar entities and reduce graph complexity.

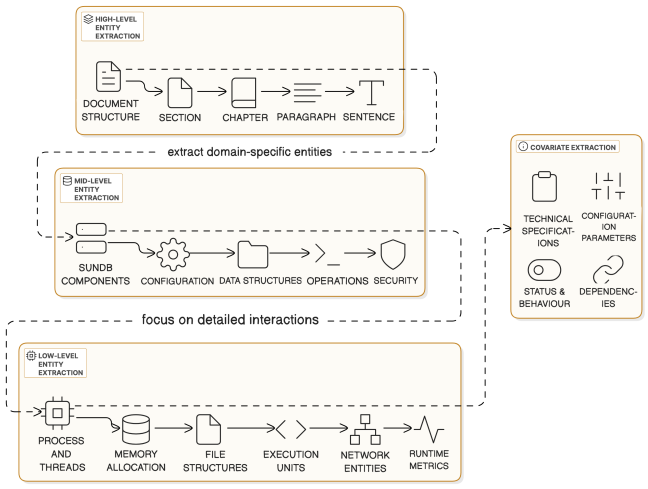


Fig. 2: Multi-level entity-relation extraction.

C. Hybrid Retrieval and Query Decomposition

As illustrated in Figure 3,when a user submits a question, DO-RAG performs a structured decomposition using an LLM-based intent analyzer. This yields sub-queries that are used to guide retrieval from both the KG and the vector store. It first retrieves relevant nodes from the KG using semantic similarity, then performs multi-hop traversal to assemble a context-rich subgraph. This graph-derived evidence is used to rewrite and disambiguate the query via a graph-aware prompt. The refined query is then vectorized and used to retrieve semantically similar chunks from the vector database. Finally, DO-RAG aggregates all sources—original and refined queries, graph context, vector-retrieved chunks, and user conversation history—into a unified prompt structure.

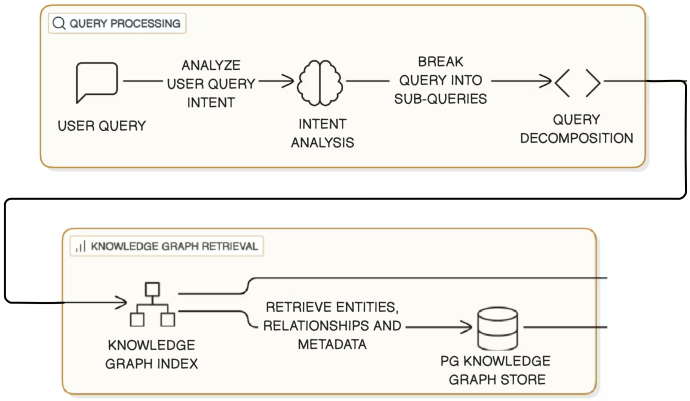
D. Grounded Answer Generation and Delivery

As shown in Figure 4, the final answer is generated using a staged prompt strategy. The initial naive prompt instructs the LLM to answer based only on the retrieved evidence while explicitly avoiding unsupported content. The output is passed through a refinement prompt that restructures and validates the answer, followed by a condensation stage that aligns tone, language, and style with the original query.

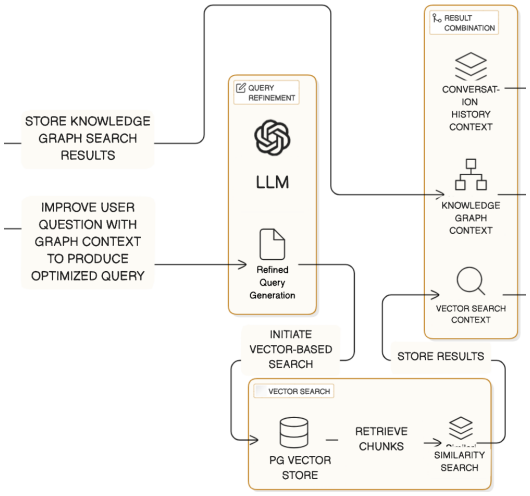
To enhance user engagement and simulate expert-like guidance, DO-RAG also generates follow-up questions based on the refined answer. The final output includes (1) a polished, verifiable answer, (2) citation footnotes tracing the answer to the source, and (3) a set of targeted follow-up questions. If the system cannot locate sufficient evidence, the model is required to return I do not know, preserving reliability and preventing hallucination.

IV. Experiments

To evaluate the DO-RAG framework, we selected SunDB, a distributed relational database management system developed by Client Service International, Inc. (CSII), as the specialized

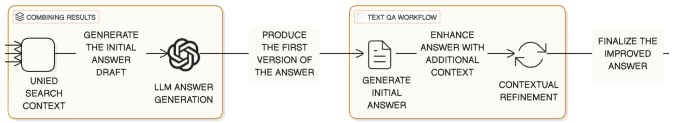


(a) Query processing and knowledge graph retrieval.

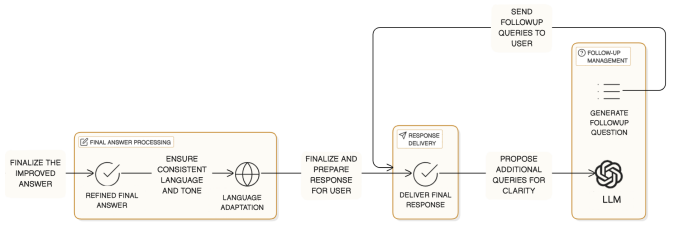


(b) Query refinement, vector Search, and result combination.

Fig. 3: Overview of the retrieval process.



(a) Combining results and Text QA workflow.



(b) Final answer processing, response delivery, and follow-up management.

Fig. 4: Overall response pipeline.

domain. SunDB is widely used in critical industries such as banking, telecommunications, and government, offering a complex, heterogeneous dataset of technical manuals, system logs, and specifications. This domain is ideal for testing DO-RAGs multimodal ingestion, multi-level entity-relationship extraction, hybrid retrieval, and grounded answer generation.

A. Experimental Setup

Hardware and Software. The evaluation was conducted on a high-performance Ubuntu workstation equipped with 64GB RAM, an NVIDIA A100 80GB GPU, and a 1TB hard disk. The experiment was implemented using locally deployed software to prioritize data security and efficiency. LangFuse (v3.29.0) [25] enabled tracing and logging of retrieval-augmented generation operations, Redis (v7.2.5) [26] managed caching, MinIO (v2025-04-22T22-12-26Z) [27] handled document storage, and ClickHouse (v25.4.3.22-stable) [28] supported high-performance analytics. PostgreSQL (v16.4) [29] served as the relational backend storage with pgvector.

Datasets. Two datasets were employed to assess cross-domain adaptability: the primary SunDB dataset, comprising technical manuals, logs, and specifications with embedded tables and code, and a secondary Electrical dataset of manuals and schematics with diagrams. Each dataset included 245 expert-curated questions with ground-truth answers, annotated with source locations (chapter, section, page) for precise validation.

Metrics and Tools. The evaluation focused on four core metrics, each with a success threshold of 0.7 (score range: 0-1), to assess retrieval and generation quality:

- **Answer Relevancy (AR)**: Measures alignment between the query and the generated answer, computed as:

$$\text{AR} = \frac{1}{N}\sum_{i=1}^{N}\cos(E_{g_i}, E_o) = \frac{1}{N}\sum_{i=1}^{N}\frac{E_{g_i}\cdot E_o}{\|E_{g_i}\|\|E_o\|},$$

  where $N = 3$, E_{g_i} is the embedding of the i-th generated answer, and E_o is the query embedding.
- **Contextual Recall (CR)**: Evaluates the retrieval of all relevant information:

$$\text{CR} = \frac{\text{Number of Attributable Statements}}{\text{Total Number of Statements}}.$$

- **Contextual Precision (CP)**: Assesses the accuracy of retrieved context, excluding irrelevant data:

$$\text{CP@K} = \frac{\sum_{k=1}^{K}(\text{Precision@k} \times v_k)}{\text{Total number of relevant items in top } K \text{ results}},$$

  where $\text{Precision@k} = \frac{\text{true positives@k}}{\text{true positives@k}+\text{false positives@k}}$, and $v_k \in \{0, 1\}$ indicates relevance.
- **Faithfulness (F)**: Measures how accurately the answer reflects the retrieved context:

$$\text{F} = \frac{\text{Number of Truthful Claims}}{\text{Total Number of Claims}}.$$

These metrics were computed using RAGAS [30] for granular insights, DeepEval [25] for end-to-end analysis, and LangFuse [25] for detailed trace analysis, logging operations from query decomposition to answer generation.

Baseline. Two baseline comparisons were established. Externally, DO-RAG, implemented as SunDB.AI, was compared

against FastGPT, a versatile open-source RAG framework with LLM-driven knowledge base construction; TiDB.AI, a graph-enhanced RAG framework tailored to database domain knowledge; and Dify.AI, an industrial-grade open-source RAG platform with generalized knowledge graph integration. Internally, DO-RAG with knowledge graph integration was contrasted against a variant relying solely on vector-based retrieval, isolating the knowledge graphs impact.

Evaluation Methodology. The evaluation followed a structured methodology. Identical knowledge bases were constructed for all frameworks using the SunDB and Electrical datasets. The same 245 questions were tested across frameworks and domains. Three language models—DeepSeek-R1, DeepSeek-V3 [31], and GPT-4o-mini [32]—were selected. Metrics were computed for each model, and a composite score, averaging the four core metrics, provided a balanced performance measure.

B. Results

External Baseline Comparison. Table I presents the composite scores for the external baseline comparison, evaluating SunDB.AI against FastGPT, TiDB.AI, and Dify.AI across the tested language models. SunDB.AI consistently outperformed all three baselines.

Figure 5 visualizes these results, illustrating SunDB.AIs consistent superiority.

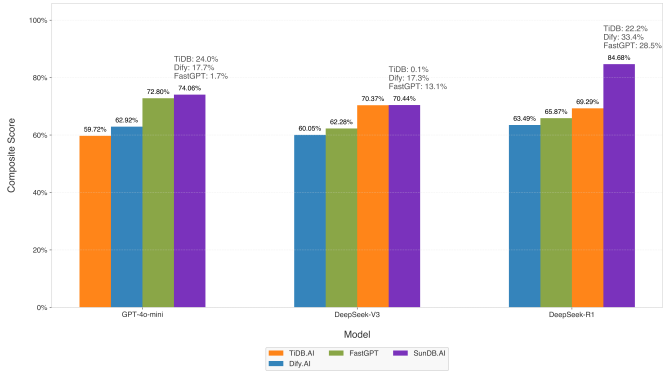


Fig. 5: Composite score comparison across frameworks and language models.

Internal Baseline Comparison. Table II evaluates the impact of knowledge graph integration using DeepSeek-R1 and DeepSeek-V3. With the knowledge graph, both models achieved perfect Contextual Recall (1.000), with DeepSeek-V3 improving Answer Relevancy by 5.7% and Contextual Precision by 2.6%. Without the knowledge graph, recall dropped to 0.9640.977, and Faithfulness was lower due to reliance on unstructured vector search. DeepSeek-R1 exhibited a slight Faithfulness decline (-5.6%) with the knowledge graph, likely due to creative deviations.

Domain-Specific Performance. Tables III and IV display the performance metrics for each language model in the SunDB and Electrical domains, respectively. In both domains, Contextual Recall values are at or near 1.0. Variations in Answer Relevancy, Contextual Precision, and Faithfulness reveal model-specific strengths.

C. Discussion

Limitations. The frameworks reliance on language models presents challenges, as creative models like DeepSeek-R1 occasionally introduced hallucinations despite knowledge graph grounding. The dataset, limited to 245 questions per domain, may not capture rare or edge-case queries, potentially limiting generalizability. The computational overhead of multi-agent extraction and hybrid retrieval, while optimized, remains significant for real-time updates in large-scale deployments.

Future Work. Future efforts will focus on enhancing hallucination mitigation through stricter prompt engineering to prioritize factual adherence. Expanding datasets to include diverse, edge-case queries will improve robustness. Distributed processing and adaptive caching mechanisms will be explored to enhance scalability and reduce latency.

V. Conclusion

This paper introduces DO-RAG, a domain-specific retrieval-augmented generation framework that addresses limitations of existing RAG systems in closed-domain QA. DO-RAG transforms unstructured, multimodal domain data into dynamic, multi-level knowledge graphs using an agentic chain-of-thought extraction pipeline. It integrates graph traversal with semantic vector search to retrieve context-rich, structurally grounded information. A post-generation refinement step cross-verifies outputs against the knowledge graph, iteratively correcting hallucinations to enhance factual accuracy. Empirical results in the database and electrical domains demonstrate near-perfect recall and answer relevancy exceeding 94%, with DO-RAG outperforming baseline frameworks by up to 33.38%. These findings demonstrate DO-RAGs effectiveness in delivering robust, high-accuracy QA across specialized domains, unifying structured knowledge representation with generative reasoning for scalable and adaptable information systems.

References

[1] DeepSeek-AI, "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," 2025.

[2] xAI, "Grok 3 beta the age of reasoning agents," 2025.

[3] Q. Team, "Qwq: Reflect deeply on the boundaries of the unknown," 2024.

[4] OpenAI, "Introducing o3 and o4 mini," 2024.

[5] Z. Yang *et al.*, "CuriousLLM: Elevating multi-document question answering with llm-enhanced knowledge graph reasoning," in *NAACL*, 2025.

[6] Q. Zhou *et al.*, "An llm-based multi-stage approach for automated test case generation from user stories," in *SEKE*, 2024.

[7] S. Siriwardhana *et al.*, "Improving the domain adaptation of retrieval augmented generation (RAG) models for open domain question answering," *Trans. Assoc. Comput. Linguistics*, vol. 11, pp. 1–17, 2023.

[8] Z. Huang *et al.*, "Recent trends in deep learning based open-domain textual question answering systems," *IEEE Access*, vol. 8, pp. 94 341–94 356, 2020.

[9] K. Bhushan *et al.*, "Systematic knowledge injection into large language models via diverse augmentation for domain-specific rag," in *NAACL*, 2025.

TABLE I: Comparative performance of graph-enhanced RAG frameworks (External Baseline).

Model	Composite Score				Improvement over SunDB.AI		
	SunDB.AI	FastGPT	TiDB.AI	Dify.AI	vs FastGPT	vs TiDB.AI	vs Dify.AI
GPT-4o-mini	**0.741**	0.728	0.597	0.629	↑ 1.70%	↑ 24.02%	↑ 17.72%
DeepSeek-V3	**0.704**	0.623	0.704	0.601	↑ 13.10%	↑ 0.10%	↑ 17.31%
DeepSeek-R1	**0.847**	0.659	0.693	0.635	↑ 28.50%	↑ 22.21%	↑ 33.38%

TABLE II: Impact of knowledge graph integration on DeepSeek models.

Metric	DeepSeek-R1		DeepSeek-V3		KG Impact (R1)	KG Impact (V3)
	Baseline	+KG	Baseline	+KG	(△)	(△)
Answer Relevancy	0.851	0.820	0.871	**0.921**	↓ 3.6%	↑ 5.7%
Contextual Recall	0.964	**1.000**	0.977	**1.000**	↑ 3.7%	↑ 2.4%
Contextual Precision	0.907	**0.918**	0.919	**0.943**	↑ 1.2%	↑ 2.6%
Faithfulness	0.718	0.678	0.711	**0.714**	↓ 5.6%	↑ 0.4%

TABLE III: Performance comparison of LLMs in the database domain.

Model	Answer Relevancy	Contextual Recall	Contextual Precision (ragas)	Faithfulness
DeepSeek-R1	0.820407	**1.000000**	0.918463	0.677958
DeepSeek-V3	0.920967	**1.000000**	0.942694	0.714305
GPT-4o	**0.944203**	0.979167	0.950279	0.770946
GPT-4o mini	0.939535	0.977273	0.921744	**0.842160**
Grok 3	0.898365	**1.000000**	**0.953021**	0.836704

TABLE IV: Performance comparison of LLMs in the electrical domain.

Model	Answer Relevancy	Contextual Recall	Contextual Precision (ragas)	Faithfulness
DeepSeek-V3	0.965342	**1.000000**	0.919427	0.912089
GPT-4o	**0.975000**	**1.000000**	**0.930203**	0.870054
GPT-4o mini	0.958386	**1.000000**	0.913326	**0.943798**
O3 mini	0.949360	**1.000000**	0.888545	0.880241

[10] Z. Zhang *et al.*, "A domain question answering algorithm based on the contrastive language-image pretraining mechanism," *J. Comput. Commun.*, vol. 11, pp. 1–15, 01 2023.

[11] A. Tu *et al.*, "Lightprof: A lightweight reasoning framework for large language model on knowledge graph," in *AAAI*, 2025.

[12] Y. Wang *et al.*, "Knowledge graph prompting for multi-document question answering," *AAAI*, vol. 38, no. 17, pp. 19 206–19 214, 03 2024.

[13] A. Zafar *et al.*, "Kimedqa: towards building knowledge-enhanced medical qa models," *J. Intell. Inf. Syst.*, 01 2024.

[14] A. Dagnino, "Importance of good data quality in ai business information systems for iiot enterprises: A data-centric ai approach," in *SEKE*, 2024.

[15] Labring, "Fastgpt: Empower ai with your expertise," 2025.

[16] P. Inc., "Tidb.ai: An open-source graphrag knowledge base tool," 2024.

[17] D. Contributors, "Dify.ai: Open-source llm app development platform," 2023.

[18] A. M. N. Allam and M. H. Haggag, "Question answering in restricted domains: An overview," *Computational Linguistics*, vol. 33, no. 1, pp. 41–66, 2012.

[19] E. Dimitrakis *et al.*, "A survey on question answering systems over linked data and documents," *J. Intell. Inf. Syst.*, vol. 55, no. 1, pp. 37–63, 2020.

[20] K. Singhal *et al.*, "Large language models encode clinical knowledge," *Nature*, vol. 620, no. 7972, pp. 172–180, 2023.

[21] P. Lewis *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," in *NIPS*, vol. 33. Curran Associates, Inc., 2020, pp. 9459–9474.

[22] X. Zhu *et al.*, "Knowledge graph-guided retrieval augmented generation," in *NAACL*, 2025, pp. 8912–8924.

[23] V. Y. Singh *et al.*, "Panda: Performance debugging for databases using LLM agents," in *CIDR*, 2024.

[24] K. Soman *et al.*, "Biomedical knowledge graph-enhanced prompt generation for large language models," *CoRR*, vol. abs/2311.17330, 2023.

[25] L. Team, "Langfuse: Observability for llm applications," 2025, version 3.29.0.

[26] R. Ltd., "Redis," 2025, version 7.2.5.

[27] I. MinIO, "Minio: High performance object storage," 2025, version 2025-04-22T22-12-26Z.

[28] I. ClickHouse, "Clickhouse: Open source column-oriented dbms," 2025, version 25.4.3.22-stable.

[29] "Postgresql," 2025, version 16.4.

[30] R. Developers, "Ragas documentation," 2023.

[31] DeepSeek-AI *et al.*, "Deepseek-v3 technical report," *CoRR*, vol. abs/2412.19437, 2024.

[32] OpenAI, "Gpt-4o mini: advancing cost-efficient intelligence," 2024.