iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** X **Month of publication:** October 2026

DOI: https://doi.org/10.22214/ijraset.2026.85389

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue X Oct 2026- Available at www.ijraset.com

A Comprehensive Review on Synapse: A GraphRAG-Based Context-Aware Research Assistant

Prof. Priti Tijare[1], Prof. Aparna Khairkar[2], Bhushan Dhok[3], Krushna Chinchamalatpure[4], Abhas Gawali[5], Devang Dhote[6]

[1, 2]*ASST. Prof, Department of Information Technology, PRMCEAM, Amravati, Maharashtra, India*
[3, 4, 5, 6]*UG Student, Department of Information Technology, PRMCEAM, Amravati, Maharashtra, India*

Abstract: ***Research assistants built on Retrieval-Augmented Generation (RAG) usually depend on vector similarity, which struggles with multi-document synthesis, multi-hop questions, and fragmented context. This review examines how Graph Retrieval-Augmented Generation (GraphRAG) closes these gaps by pairing embedding-based search with knowledge graph traversal, and proposes Synapse, a context-aware research assistant built on that idea. Synapse parses scholarly documents into structure-preserving chunks, indexes them both as vectors and as an entity–relation graph, groups the graph into thematic communities, and produces answers with inline, clickable citations to source passages. We survey recent work on RAG, knowledge graph construction, and trustworthy generation, describe the architecture and its key algorithms, and discuss practical challenges such as indexing cost and entity duplication. The review indicates that hybrid retrieval is a promising route to traceable and reliable research assistance.***

Keywords: ***Knowledge graph, retrieval-augmented generation, GraphRAG, hybrid retrieval, multi-hop reasoning, large language models, document parsing***

I. INTRODUCTION

Scholarly output is growing faster than any researcher can read it, and large language models (LLMs) have made it natural to put questions to a document collection in plain language. Left to their own parametric memory, however, LLMs can produce fluent statements that no source supports. Retrieval-Augmented Generation (RAG) reduces this risk by fetching relevant passages at query time and asking the model to answer from them [1]. Most RAG systems embed fixed-size text chunks and return the nearest neighbors of the query. This suits narrow factual lookups but breaks down on research material in three ways. Chunk boundaries can separate a claim from its evidence; isolated chunks carry no explicit links between entities, so questions needing several reasoning steps go unanswered; and corpus-wide questions about themes or trends cannot be answered from a handful of passages [2], [3]. Work on retrieval and structuring adds that unorganized passages give the generator little to reason over [4], and broader RAG surveys document the same pattern as the field diversifies [5], [6]. GraphRAG responds by extracting entities and relations from the corpus into a knowledge graph and retrieving from the graph and the text together. Reliability matters as much as recall, since researchers need answers they can verify; trustworthiness, covering factuality, robustness, and transparency, has become a central theme [7]. Evaluation of RAG is still maturing [8], and agentic RAG, where the model plans its own retrieval, is an emerging direction [9]. We make three contributions: (i) a concise review of RAG, knowledge graph, and GraphRAG literature relevant to research assistants; (ii) the design of Synapse, our layered GraphRAG architecture with citation-level traceability; and (iii) a discussion of implementation challenges, planned evaluation, and limitations. Section II gives background, Section III reviews the literature, Section IV presents the architecture, Section V covers implementation, Section VI discusses challenges, Section VII covers applications, evaluation, and limitations, and Section VIII concludes.

II. BACKGROUND

A. Document Question Answering and Vector Retrieval

Document-based question answering confines the model to a bounded corpus and proceeds in four stages: collecting files, processing them into chunks, retrieving the chunks relevant to a query, and generating an answer from them [1], [6]. Because the model is told to answer from the retrieved text, it relies less on whatever it memorized during training, and the answer can in principle be checked against the corpus.

©IJRASET: All Rights are Reserved

In the common dense-retrieval setup, an embedding model turns every chunk into a list of numbers so that passages with similar meaning end up close together. The query is converted the same way, and the system returns the few chunks that lie closest to it. This handles paraphrase well, since a question about a "puppy" can find a passage about a "dog". What it cannot do is tell how the people, methods, and datasets named in different passages are related, because closeness in meaning says nothing about explicit connections [2]. The quality of the final answer therefore depends heavily on whether the retrieval step surfaces the right, and logically connected, evidence.

B. Knowledge Graphs and Multi-Hop Reasoning

A knowledge graph stores facts as nodes and labeled links. Nodes stand for entities such as authors, methods, datasets, or institutions, and each link states a relation between two of them, for example that a method was evaluated on a dataset; every fact is thus a simple subject–relation–object triple [10]. Such graphs can be queried declaratively [11], extracted automatically from text [12], [13], and kept current, refined, and audited for quality [14], [15], [16].

This structure suits multi-hop questions [17], where the answer lies at the end of a chain of relations. For example, "Which optimizer did the author of Paper A use for Task Z?" requires moving from the paper to its author, from the author to a method, and from the method to the task. Embedding search may retrieve passages that mention "Paper A" or "optimizer" yet miss the chain when its links sit in different documents, whereas graph traversal follows those links explicitly [18]. Combining LLMs with knowledge graphs for question answering is an active area [19].

C. Why Combine Vectors and Graphs

The two representations complement each other. Vector search keeps the original wording of a passage and is forgiving about phrasing, but it sees each chunk in isolation. A graph captures structure and relationships across documents, but it compresses text into short facts and can lose the nuance of the original sentences. GraphRAG places the graph directly in the retrieval path, so that the system can use text for detail and the graph for connection [2], [3]. Synapse builds on this pairing.

III. LITERATURE REVIEW

A. Retrieval-Augmented Generation and Its Limits

Successive surveys describe a progression from basic retrieve-then-read pipelines toward richer, modular systems with steps such as query rewriting, re-ranking, and repeated retrieval [1], [5], [6]. They agree that the basic form degrades on long, structured documents, because fixed-size chunking can cut arguments apart and because ranking passages independently ignores how they relate. Evaluating RAG remains an open problem, since retrieval quality and generation faithfulness need separate measures [8]. Agentic RAG [9] goes a step further by letting autonomous components decide when and how to retrieve.

B. Knowledge Graphs and LLM-Based Construction

Classical graph construction relied on hand-built ontologies and extraction rules, which made large graphs costly to derive from unstructured text [12]. LLMs lower that cost by extracting entities and relations from prompts instead of task-specific trained extractors [13]. Quality remains the main concern: extracted graphs contain duplicates, missing links, and wrong facts, which motivates work on refinement and quality management [15], [16]. On the reasoning side, neural-symbolic methods combine learned representations with logical query operations over graphs [18], and LLM–knowledge-graph hybrids for question answering have recently been surveyed [19].

C. GraphRAG

The GraphRAG surveys in [2] and [3] propose a three-part taxonomy, which we adopt here: graph-based indexing, graph-guided retrieval, and graph-enhanced generation. Indexing converts a corpus into entities and relations, often with community structure; retrieval selects nodes, paths, or subgraphs alongside text passages; and generation conditions the LLM on both. The reviewed studies report advantages on multi-hop and corpus-level questions, at the price of a heavier indexing phase [2], [3]. Treating retrieval and structuring together [4] also suggests that the structure should be matched to the question type rather than applied uniformly. Table I summarizes the approaches discussed [1]–[3].

TABLE I
COMPARISON OF RETRIEVAL APPROACHES

Approach	Retrieval Method	Relation Awareness	Main Limitation
Keyword search (BM25)	Lexical matching	None	Misses synonyms and paraphrases
Vector search	Embedding similarity	Low	Relations not represented; weak on corpus-wide questions
Naive RAG	Vector retrieval + LLM	Limited	Context fragmentation; unsupported links possible
Knowledge graph QA	Graph traversal	High	Costly construction; local text nuance lost
GraphRAG	Vector + graph + LLM	High	Higher indexing cost

D. Trustworthiness, Evaluation, and Document Parsing

Trustworthiness surveys name factuality, robustness, transparency, and privacy as dimensions on which RAG systems must be judged [7], so graph-based designs should be assessed on these as well, not on accuracy alone. Evaluation surveys likewise separate the quality of retrieved evidence from the quality of the generated answer [8]. A further practical dependency is document parsing: research papers contain multi-column layouts, tables, and figures, and errors made at this first stage propagate to every later one [20], [21]. Another survey frames trustworthiness along comparable dimensions [22]. Table II lists the groups of surveys reviewed here and what each contributes to the Synapse design.

TABLE II
SCOPE OF THE SURVEYS REVIEWED

Topic	Refs.	Contribution to This Review
RAG foundations	[1], [5], [6]	Paradigms from naive to modular RAG; components and enhancements
GraphRAG	[2]–[4]	Graph-based indexing, retrieval, and generation; structuring retrieved knowledge
Trust and evaluation	[7], [8], [22]	Dimensions of trustworthiness; how to measure RAG quality
Agentic RAG	[9]	Models that plan and adapt retrieval steps
Knowledge graphs	[10]–[16]	Fundamentals, querying, LLM-based construction, refinement, quality
Graph reasoning	[17]–[19]	Multi-hop questions; neural-symbolic and LLM-based graph QA
Document parsing	[20], [21]	Layout analysis, OCR, and structured extraction from documents

E. Open Issues

Across these surveys a few issues recur: the cost of building graphs, the quality of automatically extracted graphs, the lack of settled evaluation practice, and the difficulty of tracing a generated claim back to its source [2], [3], [8], [16]. The design we describe next addresses each of these four issues.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

IV. PROPOSED SYSTEM ARCHITECTURE

We organize Synapse in five layers around a dual-pathway design: every document is turned into both dense vectors and a relational graph, and both are consulted at query time. Fig. 1 gives an overview.

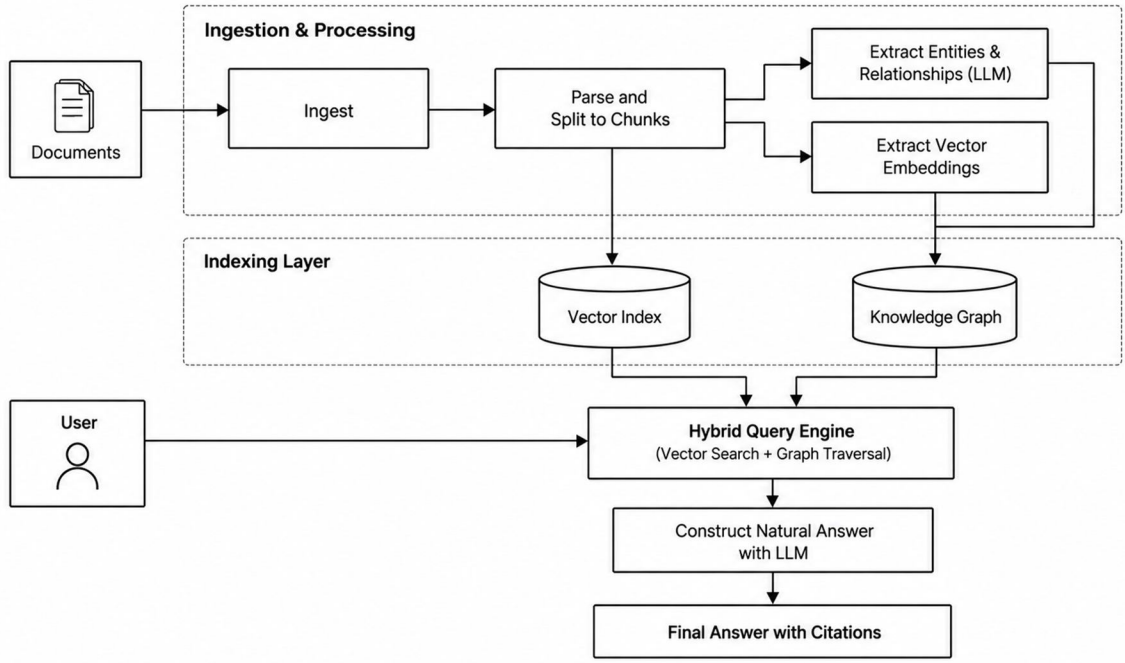


Fig. 1 Overall architecture of Synapse: documents are indexed as vectors and as a knowledge graph, and a hybrid retriever feeds a grounded LLM

A. *User and Application Layer*

The front end handles authentication, sessions, and document upload. Beyond a chat window, it renders retrieved subgraphs and shows each citation as a clickable link that opens the supporting passage, so users can check claims against the source.

B. *Backend Orchestration and Ingestion*

The backend coordinates the APIs and pipelines. Uploaded PDF, DOCX, and PPTX files are converted to Markdown by a parsing library such as MarkItDown, which removes layout noise while keeping headings, lists, and tables [20], [21]. The text is then split at structural boundaries (headers, paragraphs, list items) rather than at fixed token counts, and each chunk keeps its document ID, section title, and page number. This metadata is what later makes exact citation possible.

C. *Dual Indexing Pathways*

1) *Vector pathway:* Each chunk is embedded and stored in a vector database such as Qdrant, FAISS, or Chroma, enabling fast similarity search over the original wording.
2) *Graph pathway:* The same chunks are given to an LLM prompted for information extraction. It identifies entities (methods, datasets, authors, organizations) and the relations between them, and the resulting triples are stored in a graph database such as Neo4j. Duplicate entities are merged and related entities are grouped into communities whose summaries are stored for retrieval, following the community-summary approach introduced by Edge et al. [23] and detailed in Section V.

D. *Hybrid Retrieval Engine*

1) *Semantic recall:* The query is embedded and the most similar chunks are retrieved.
2) *Graph expansion:* Entities found in the query and in the retrieved chunks become entry nodes, from which one-hop and two-hop neighbors are collected to surface less obvious connections.
3) *Re-ranking:* Chunks, triples, and community summaries are merged, de-duplicated, and re-ranked into a single context window.

E. Grounded Generation and Attribution

The LLM is instructed to answer only from the supplied context and to tag every claim with the identifiers of the chunks that support it. The front end resolves these tags to the original passages. This design lowers the chance of unsupported answers and makes the remaining ones easy to spot, although it cannot remove errors entirely [7], [22].

F. Security and Workspace Isolation

Researchers often upload unpublished drafts or confidential reports. Synapse therefore keeps each user's documents, vectors, and graph in a separate workspace that is reachable only through an authenticated session, so that content from one user never appears in another user's answers.

G. Example Query Flow

Consider two questions. A specific one, such as "Which dataset was used to evaluate the method in Paper A?", is answered mainly from retrieved chunks, with the graph confirming which entities are involved. A broad one, such as "What are the main research directions in this collection?", is answered mainly from the stored community summaries, because no handful of passages could cover it.

In both cases the answer returns with links to the chunks it relied on. A router that judges the intent of each question decides how much weight to give each pathway.

V. IMPLEMENTATION METHODOLOGY

A. Parsing and Chunking

Academic documents are heterogeneous, with multi-column layouts, embedded figures, and varied citation styles. Normalizing everything to Markdown gives the later stages a single, clean input format regardless of the original file type [20], [21]. Chunks follow the document's own structure, so a methodology paragraph is not separated from the result it supports, and every chunk is stored with its identifier so that the pipeline can always point back to it.

B. Entity Extraction and Normalization

The same concept often appears under several names across papers, such as an acronym in one and its full form in another. Before a newly extracted entity is added to the graph, it is compared with existing entities; when the two look like the same thing, an LLM confirms the match and the new relations are attached to the existing node. This keeps aliases from fragmenting the graph and is consistent with the quality concerns raised in the literature [15], [16].

C. Community Grouping and Summaries

For corpus-level questions, Synapse groups densely connected entities into communities that correspond to research themes, such as a family of methods or a shared application area. An LLM then writes a short summary of each community, and the summaries are stored for retrieval.

This follows the global GraphRAG pipeline of Edge et al. [23], in which each community summary contributes a partial answer and the partial answers are then combined into one response. Thematic questions are thus answered from the summaries rather than from raw text across many documents, which would be slow and often impossible within the model's context limit [2], [3]. To keep themes meaningful, the strength of a link takes into account both how often two entities appear together and how similar their meanings are.

D. Technology Stack

An asynchronous Python backend (FastAPI) handles concurrent ingestion and parallel retrieval. LangChain and LlamaIndex orchestrate extraction and generation with hosted LLMs such as Gemini or OpenAI models. Qdrant or FAISS serves the vector index, Neo4j with Cypher serves the graph, and a JavaScript front end provides interactive visualization of the retrieved graph and citations.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

VI. CHALLENGES AND MITIGATIONS

Table III lists the main obstacles anticipated in building a GraphRAG research assistant and the mitigation adopted in Synapse.

TABLE III
CHALLENGES AND MITIGATIONS IN BUILDING SYNAPSE

Challenge	Mitigation	Refs.
Heterogeneous file formats	Convert all inputs to Markdown before processing, preserving headings and tables	[20], [21]
Context lost in chunking	Structure-aware chunking with section and page metadata	[1], [6]
Duplicate entities	Similarity check plus LLM confirmation before a node is committed	[15], [16]
Merging vector and graph results	Re-rank chunks, triples, and community summaries into one context	[2], [3]
Unfaithful or untraceable answers	Grounded prompts; mandatory chunk-ID citations rendered as links	[7], [22]
Indexing cost and latency	Batch and asynchronous extraction, selective LLM calls, cached community summaries	[2], [13], [23]
Local versus global queries	Intent-based query routing between chunk and community retrieval	[3], [9]

Two trade-offs deserve emphasis. First, indexing is the expensive step: passing a whole corpus through an LLM costs far more than computing embeddings, so Synapse limits LLM calls to where they add value and caches what it computes [2], [13]. Second, running vector search and graph expansion in parallel keeps response time acceptable, but the combined context must be trimmed carefully so that the generator is not swamped by marginally relevant material.

VII. APPLICATIONS, EVALUATION, AND LIMITATIONS

A. *Intended Applications*

The primary use is literature review: a student or researcher uploads a set of papers and asks what methods exist, how they relate, and which results support a claim. The same design suits thesis preparation, onboarding to an unfamiliar field, and technical or organizational document collections where answers must be traceable. Fields that demand verifiable answers, such as medicine and engineering, benefit most from citation-level grounding [7].

B. *Planned Evaluation*

Because Synapse is a proposed design, systematic evaluation is our next step. Following the separation of retrieval and generation quality in the evaluation literature [8], we will compare Synapse against a vector-only baseline on three question types: specific facts, multi-hop questions, and corpus-wide thematic questions. We will measure answer correctness, citation accuracy (whether each cited chunk really supports the claim), completeness, response time, and indexing cost. Based on the surveyed work, we expect the largest gains on multi-hop and thematic questions [2], [3].

C. *Limitations*

Several limitations should be stated plainly. Extraction quality depends on the LLM, so some relations will be missed or wrong, and graph errors can mislead retrieval [15], [16]. Indexing cost grows with corpus size [2]. Domain-specific vocabulary may need tuned prompts. Finally, grounding and citations reduce but do not eliminate unsupported statements, so users should still treat the cited passages as the final authority [7], [22].

VIII. CONCLUSION AND FUTURE SCOPE

This review examined how pairing vector retrieval with knowledge graph traversal addresses the limits of standard RAG for scholarly question answering, and described Synapse, our layered design that preserves document structure, indexes content as both vectors and a graph, summarizes communities for corpus-level questions, and ties each claim to a source passage. The case for the design rests on the surveyed literature. We have not yet evaluated Synapse, so its benefits remain expected rather than measured.

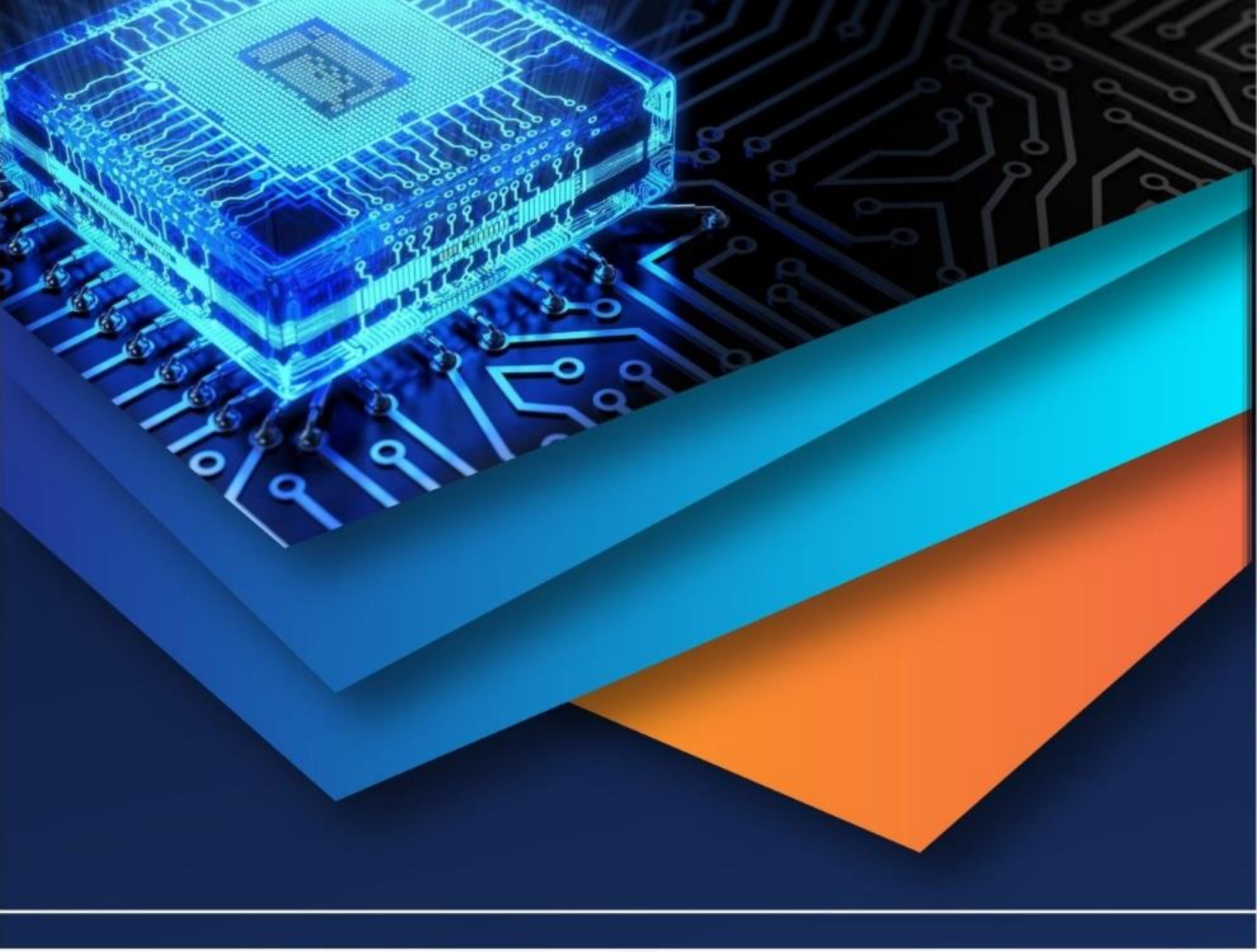
©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

Future work includes benchmarking against vector-only baselines with established RAG evaluation protocols [8], incremental graph updates that avoid full rebuilds when documents are added, multimodal parsing of figures and tables [20], and agentic retrieval routing that lets the system plan its own search steps [9].

REFERENCES

[1] Y. Gao *et al.*, "Retrieval-augmented generation for large language models: A survey," *arXiv* preprint arXiv:2312.10997, 2023, doi: 10.48550/arXiv.2312.10997. [Online]. Available: https://arxiv.org/abs/2312.10997.
[2] B. Peng et al., "Graph retrieval-augmented generation: A survey," ACM Trans. Inf. Syst., vol. 44, no. 2, pp. 1–52, 2026, doi: 10.1145/3777378.
[3] Q. Zhang et al., "A survey of graph retrieval-augmented generation for customized large language models," arXiv preprint arXiv:2501.13958, 2025, doi: 10.48550/arXiv.2501.13958. [Online]. Available: https://arxiv.org/abs/2501.13958.
[4] P. Jiang, S. Ouyang, Y. Jiao, M. Zhong, R. Tian, and J. Han, "A survey on retrieval and structuring augmented generation with large language models," in Proc. 31st ACM SIGKDD Conf. Knowledge Discovery and Data Mining, 2025, doi: 10.1145/3711896.3736557.
[5] Y. Huang and J. X. Huang, "A survey on retrieval-augmented text generation for large language models," ACM Comput. Surv., vol. 58, no. 12, Art. no. 300, pp. 1–38, 2026, doi: 10.1145/3805774.
[6] C. Sharma, "Retrieval-augmented generation: A comprehensive survey of architectures, enhancements, and robustness frontiers," arXiv preprint arXiv:2506.00054, 2025, doi: 10.48550/arXiv.2506.00054. [Online]. Available: https://arxiv.org/abs/2506.00054.
[7] B. Ni et al., "Towards trustworthy retrieval augmented generation for large language models: A survey," arXiv preprint arXiv:2502.06872, 2025, doi: 10.48550/arXiv.2502.06872. [Online]. Available: https://arxiv.org/abs/2502.06872.
[8] A. Gan et al., "Retrieval augmented generation evaluation in the era of large language models: A comprehensive survey," arXiv preprint arXiv:2504.14891, 2025, doi: 10.48550/arXiv.2504.14891. [Online]. Available: https://arxiv.org/abs/2504.14891.
[9] A. Singh et al., "Agentic retrieval-augmented generation: A survey on agentic RAG," arXiv preprint arXiv:2501.09136, 2025, doi: 10.48550/arXiv.2501.09136. [Online]. Available: https://arxiv.org/abs/2501.09136.
[10] A. Hogan et al., "Knowledge graphs," ACM Comput. Surv., vol. 54, no. 4, Art. no. 71, pp. 1–37, 2021, doi: 10.1145/3447772.
[11] A. Khan, "Knowledge graphs querying," ACM SIGMOD Rec., vol. 52, no. 1, pp. 6–17, 2023, doi: 10.1145/3604437.3604439.
[12] L. Zhong, J. Wu, Q. Li, H. Peng, and X. Wu, "A comprehensive survey on automatic knowledge graph construction," ACM Comput. Surv., vol. 56, no. 4, Art. no. 94, pp. 1–62, 2024, doi: 10.1145/3618295.
[13] H. Bian, "LLM-empowered knowledge graph construction: A survey," arXiv preprint arXiv:2510.20345, 2025, doi: 10.48550/arXiv.2510.20345. [Online]. Available: https://arxiv.org/abs/2510.20345.
[14] X. Jiang et al., "On the evolution of knowledge graphs: A survey and perspective," arXiv preprint arXiv:2310.04835, 2023, doi: 10.48550/arXiv.2310.04835. [Online]. Available: https://arxiv.org/abs/2310.04835.
[15] B. Subagdja, D. Shanthoshigaa, Z. Wang, and A.-H. Tan, "Machine learning for refining knowledge graphs: A survey," ACM Comput. Surv., vol. 56, no. 6, Art. no. 156, pp. 1–38, 2024, doi: 10.1145/3640313.
[16] B. Xue and L. Zou, "Knowledge graph quality management: A comprehensive survey," IEEE Trans. Knowl. Data Eng., vol. 35, no. 5, pp. 4969–4988, May 2023, doi: 10.1109/TKDE.2022.3150080.
[17] V. Mavi, A. Jangra, and A. Jatowt, "Multi-hop question answering," Found. Trends Inf. Retr., vol. 17, no. 5, pp. 457–586, 2024, doi: 10.1561/1500000102.
[18] L. Liu, Z. Wang, and H. Tong, "Neural-symbolic reasoning over knowledge graphs: A survey from a query perspective," arXiv preprint arXiv:2412.10390, 2024, doi: 10.48550/arXiv.2412.10390. [Online]. Available: https://arxiv.org/abs/2412.10390.
[19] C. Ma, Y. Chen, T. Wu, A. Khan, and H. Wang, "Large language models meet knowledge graphs for question answering: Synthesis and opportunities," in Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP), Nov. 2025, pp. 24578–24597, doi: 10.18653/v1/2025.emnlp-main.1249.
[20] Q. Zhang, B. Wang, V. S.-J. Huang, J. Zhang, Z. Wang, H. Liang, C. He, and W. Zhang, "Document parsing unveiled: Techniques, challenges, and prospects for structured information extraction," arXiv preprint arXiv:2410.21169, 2024, doi: 10.48550/arXiv.2410.21169. [Online]. Available: https://arxiv.org/abs/2410.21169.
[21] N. Subramani et al., "A survey of deep learning approaches for OCR and document understanding," arXiv preprint arXiv:2011.13534, 2020, doi: 10.48550/arXiv.2011.13534. [Online]. Available: https://arxiv.org/abs/2011.13534.
[22] Y. Zhou et al., "Trustworthiness in retrieval-augmented generation systems: A survey," arXiv preprint arXiv:2409.10102, 2024, doi: 10.48550/arXiv.2409.10102. [Online]. Available: https://arxiv.org/abs/2409.10102.
[23] D. Edge et al., "From local to global: A graph RAG approach to query-focused summarization," arXiv preprint arXiv:2404.16130, 2024, doi: 10.48550/arXiv.2404.16130. [Online]. Available: https://arxiv.org/abs/2404.16130.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)