iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** XI **Month of publication:** November 2026

DOI: https://doi.org/10.22214/ijraset.2026.85707

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 XI Nov 2026- Available at www.ijraset.com

From Vector Retrieval to Knowledge Graphs: A Comprehensive Survey of Retrieval-Augmented Generation Techniques for Multi-Hop Reasoning

Adarsha Sagar H V[1], Soham Hathi[2], Shrinanda Shivprasad Dinde[3], Sinchana S[4], Shubhanshu Raj[5]
B.M.S College of Engineering, India

I. INTRODUCTION

The rapid development of Large Language Models (LLMs) has significantly changed the way information is processed, retrieved and generated. Despite their ability to understand natural language and generate meaningful responses, LLMs still face limitations when dealing with knowledge-intensive tasks. Their responses may contain outdated information, factual inaccuracies or unsupported statements, particularly when questions require information that is not adequately represented in their training data. Retrieval-Augmented Generation (RAG) has emerged as an effective approach to address these limitations by connecting language models with external knowledge sources. Instead of relying entirely on information stored in model parameters, RAG retrieves relevant information from an external collection and provides it as context to the language model during response generation [1].

Traditional RAG systems generally follow a retrieval-and-generation pipeline. Documents are collected, divided into smaller chunks, converted into vector representations using embedding models and stored in a vector index. When a user submits a query, the system generates its embedding and retrieves the most relevant chunks based on similarity measures. Dense Passage Retrieval demonstrated the effectiveness of learned dense representations for retrieving relevant passages in open-domain question answering, establishing an important foundation for embedding-based retrieval systems [2]. Alongside dense retrieval, sparse retrieval methods such as TF-IDF and BM25 continue to be widely used because of their effectiveness in identifying documents based on lexical relevance. The combination of sparse and dense retrieval has further contributed to the development of hybrid retrieval approaches, which aim to improve information coverage by considering both lexical and semantic relevance.

Although vector-based retrieval has improved the ability of LLMs to access external information, retrieving relevant individual passages does not always ensure that the system can answer complex questions. Many real-world queries require information to be collected from multiple documents, connected through relationships and interpreted together before generating an answer. For example, a question asking how an organization influenced the development of a particular technology may require identifying the organization, retrieving its associated projects, connecting those projects to relevant technologies and combining evidence from different sources. Such questions involve multiple retrieval steps rather than a single similarity-based search. This requirement is commonly referred to as multi-hop reasoning, where the answer depends on establishing connections between multiple pieces of evidence.

The limitations of conventional retrieval systems in handling such questions have encouraged research into more advanced RAG architectures. Techniques such as query expansion, query rewriting, reranking, contextual retrieval and iterative retrieval attempt to improve the relevance and completeness of retrieved information. Other approaches, including hierarchical retrieval and retrieval over summarized document representations, aim to address the limitations of fixed-size chunk retrieval. These developments demonstrate that improving RAG requires attention not only to the quality of embeddings but also to how queries are interpreted, information is retrieved and evidence is combined during generation.

Knowledge graphs provide another approach to representing and retrieving information. Unlike conventional vector indexes, which primarily organize information through numerical representations of text, knowledge graphs explicitly represent entities and the relationships connecting them. Information is commonly represented through nodes and edges, where nodes correspond to entities or concepts and edges describe their relationships. This structure makes it possible to identify connections between entities and retrieve information through graph traversal, path exploration and subgraph selection. When integrated with LLMs, knowledge graphs provide a structured source of information that can support reasoning across multiple related entities and documents.

The integration of knowledge graphs with Retrieval-Augmented Generation has led to the development of Graph-based Retrieval-Augmented Generation (GraphRAG). One notable approach is presented by Edge et al., which introduces a graph-based indexing

©IJRASET: All Rights are Reserved

and retrieval framework that constructs an entity knowledge graph from source documents and generates community summaries to support question answering over large document collections [3]. Other graph-based approaches investigate different methods of entity extraction, relationship representation, graph construction and retrieval. These approaches extend conventional RAG by incorporating structural information that may not be sufficiently represented through independent text chunks.

However, graph-based retrieval also introduces additional challenges. Constructing and maintaining knowledge graphs can require significant computational resources, and the quality of retrieval depends on the accuracy of entity extraction and relationship identification. Graph traversal may retrieve irrelevant or excessive information when the search process is not sufficiently constrained. Furthermore, converting retrieved graph structures into a suitable context for language models requires careful selection and organization of relevant evidence. Therefore, the use of knowledge graphs does not automatically guarantee accurate multi-hop reasoning. The effectiveness of a GraphRAG system depends on how well it identifies relevant entities, discovers meaningful relationships, retrieves the required evidence and preserves the connections needed to answer a query.

The importance of evaluating these capabilities is reflected in the development of multi-hop question-answering datasets and benchmarks. HotpotQA introduced questions that require reasoning over multiple supporting documents and provides supporting facts to facilitate explainable question answering [4]. Similarly, 2WikiMultiHopQA incorporates structured and unstructured information and provides evidence paths to evaluate reasoning across multiple steps [5]. More recently, MultiHop-RAG introduced a benchmark specifically designed to examine the retrieval and generation capabilities of RAG systems when answering multi-hop queries [6]. These datasets provide useful foundations for studying how different retrieval architectures handle questions that require connecting information from multiple sources.

Although research in vector-based retrieval, advanced RAG and graph-based retrieval has progressed considerably, understanding their capabilities in the context of multi-hop reasoning requires examining these approaches together. Dense retrieval, sparse retrieval, hybrid search, iterative retrieval and graph-based retrieval address different aspects of the information retrieval problem. Their effectiveness may vary depending on query complexity, the availability of structured relationships, the number of reasoning steps and the quality of the underlying knowledge representation. A comparative understanding of these approaches is therefore important for identifying their strengths, limitations and suitability for different multi-hop question-answering scenarios.

In this paper, we present a comprehensive survey of the evolution of Retrieval-Augmented Generation, beginning with traditional sparse and dense retrieval techniques and extending to advanced retrieval architectures and knowledge graph-based approaches. Particular attention is given to the mechanisms through which these systems support multi-hop reasoning, including query decomposition, entity linking, relationship traversal, path-based retrieval, subgraph retrieval and iterative evidence collection. The survey also examines the embedding methods, vector indexing techniques, graph representations, retrieval frameworks, datasets and evaluation metrics used across existing approaches.

The main contributions of this survey are as follows:

1) Review of RAG evolution: We examine the development of retrieval-augmented generation from traditional sparse and dense retrieval to hybrid, advanced and graph-based retrieval architectures.
2) Classification of retrieval techniques: We organize different retrieval mechanisms, embedding approaches, vector indexing methods and knowledge graph-based techniques according to their roles in the RAG pipeline.
3) Analysis of multi-hop reasoning approaches: We examine how existing systems retrieve, connect and integrate evidence across multiple documents and entities, with particular attention to iterative retrieval, graph traversal, path exploration and subgraph-based reasoning.
4) Comparative evaluation: We review commonly used datasets, benchmarks and evaluation metrics to understand how different RAG architectures are assessed in terms of retrieval quality, answer correctness, reasoning performance and computational efficiency.
5) Identification of research challenges: We discuss limitations related to retrieval completeness, relationship extraction, reasoning-chain preservation, graph construction, scalability and the integration of structured and unstructured knowledge.

The remainder of this paper is organized as follows. Section 2 presents the background and evolution of Retrieval-Augmented Generation. Section 3 discusses traditional retrieval mechanisms, including sparse, dense and hybrid retrieval. Section 4 examines advanced RAG architectures and retrieval optimization techniques. Section 5 focuses on knowledge graph-based RAG approaches and their underlying mechanisms. Section 6 provides a detailed discussion of multi-hop reasoning strategies. Section 7 presents a comparative analysis of different retrieval architectures. Section 8 reviews datasets, benchmarks and evaluation metrics. Section 9

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

II. BACKGROUND AND EVOLUTION OF RETRIEVAL-AUGMENTED GENERATION

A. Language Models and the Need for External Knowledge

Large Language Models (LLMs) have demonstrated strong capabilities in natural language understanding, text generation, summarization, and question answering. These models acquire knowledge during pre-training from large text corpora and encode it within their parameters. However, relying entirely on parametric knowledge introduces several limitations. The information stored within a model may become outdated, domain-specific knowledge may be insufficiently represented, and the model may generate plausible but factually incorrect responses when the required information is unavailable. Furthermore, updating a model's internal knowledge generally requires additional training or fine-tuning, which can be computationally expensive. Retrieval-Augmented Generation (RAG) addresses these limitations by connecting language models to external knowledge sources. Instead of relying exclusively on information acquired during pre-training, a RAG system retrieves relevant information at inference time and incorporates it into the generation process. This enables the model to use information from external documents, databases, and other knowledge repositories while generating its responses [1]. The integration of retrieval and generation also introduces a degree of separation between knowledge storage and language generation. External knowledge can be updated independently of the language model, making the architecture suitable for applications involving frequently changing information and specialized document collections. However, the quality of generated responses depends on the relevance, completeness, and reliability of the retrieved information.

B. Foundations of Retrieval-Augmented Generation

The development of RAG is closely related to earlier research on incorporating external knowledge into language models. REALM (Retrieval-Augmented Language Model Pre-Training) introduced a latent knowledge retrieval mechanism into language model pre-training, allowing the model to retrieve and attend to relevant documents from a large external corpus [7]. This demonstrated the potential of combining parametric language representations with non-parametric knowledge storage. Lewis et al. subsequently introduced a general Retrieval-Augmented Generation framework that combines a pre-trained sequence-to-sequence generator with a dense vector-based document retriever [1]. The architecture retrieves relevant passages from an external knowledge source and uses them as additional context for generating responses. The original RAG formulation demonstrated improvements in knowledge-intensive tasks, including open-domain question answering and fact verification. A conventional RAG pipeline consists of three major components: document indexing, information retrieval, and response generation.

1) Document indexing: The source documents are collected, processed, and divided into smaller text segments. These segments are converted into numerical representations using embedding models and stored in an index for efficient retrieval.
2) Information retrieval: When a user submits a query, the system processes it and retrieves relevant documents or text segments from the indexed knowledge base. Retrieval may be based on lexical matching, semantic similarity, or a combination of different retrieval mechanisms.
3) Response generation: The retrieved information is provided to the language model as contextual input. The model uses this information to generate a response relevant to the original query.

The conventional retrieval-augmented generation process follows a retrieve-then-generate approach, in which retrieval generally occurs before response generation. Although this architecture provides an effective foundation for knowledge-grounded question answering, its effectiveness depends on whether the retrieved passages contain sufficient information to answer the query.

C. Evolution of RAG Architectures

As RAG applications expanded beyond basic question answering, researchers identified limitations in conventional retrieval pipelines. These limitations motivated the development of architectures that improve document preparation, query processing, retrieval quality, contextual organization, and generation.

Gao et al. organized the development of RAG architectures into three broad categories: Naive RAG, Advanced RAG, and Modular RAG [8]. These categories provide a useful framework for understanding architectural developments, although they are not mutually exclusive and should not be interpreted as strictly separated stages.

1) Naive RAG: Naive RAG represents the conventional retrieve-then-generate architecture. Documents are divided into chunks, converted into embeddings, and stored in a vector index. At query time, the system retrieves the most similar chunks and provides them to the language model. Its relatively simple architecture makes implementation straightforward. However, it can be affected by poor chunk boundaries, irrelevant retrieval results, incomplete context, and limitations in capturing relationships across multiple documents.

2) Advanced RAG: Advanced RAG introduces additional processing stages to improve the quality of retrieval and generation. Techniques such as query rewriting, query expansion, reranking, contextual retrieval, and hierarchical indexing are incorporated to address limitations in the conventional pipeline. These techniques attempt to improve the relevance of retrieved information and provide the language model with more useful context. However, improving the relevance of individual passages does not necessarily ensure that all evidence required for a complex question has been retrieved.
3) Modular RAG: Modular RAG provides a more flexible architecture in which retrieval, transformation, ranking, and generation are organized into independent or interchangeable components. Depending on the application, a system may incorporate multiple retrieval mechanisms, query transformation modules, iterative retrieval stages, and additional verification components. This architecture allows different retrieval strategies to be combined according to the requirements of a particular task.

Further developments introduced retrieval mechanisms that incorporate feedback and reasoning into the retrieval process. Self-RAG, proposed by Asai et al., uses reflection tokens to enable a language model to determine when retrieval is required and to evaluate retrieved passages and generated responses [9]. Corrective Retrieval-Augmented Generation (CRAG) introduces a retrieval evaluator to assess the quality of retrieved documents and trigger corrective actions when the retrieved information is insufficient or unreliable [10].

Another development is RAPTOR, which constructs a hierarchical tree by recursively clustering and summarizing text chunks. Instead of retrieving only individual text segments, RAPTOR allows retrieval at different levels of abstraction, providing access to both detailed information and broader document context [11].

These developments illustrate that RAG is no longer limited to a single retrieval operation followed by generation. Modern architectures increasingly incorporate additional stages to improve retrieval quality, contextual understanding, and response reliability. Nevertheless, the effectiveness of these approaches for multi-hop reasoning depends on their ability to retrieve and connect evidence distributed across multiple sources.

D. From Unstructured Text Retrieval to Structured Knowledge Retrieval

Most conventional RAG systems operate on unstructured text collections. Information is divided into chunks, and retrieval is performed by identifying passages that are relevant to a given query. Although this approach is effective for retrieving directly related information, it does not explicitly represent the relationships between entities mentioned across different documents.

For example, consider a question that requires identifying the technologies used by an organization through its associated research projects. The required information may be distributed across several documents. One document may identify the organization and its projects, while another describes the technologies associated with a particular project. Retrieving passages based only on their similarity to the original query may not provide all the intermediate information required to establish the relationship.

Knowledge Graphs (KGs) offer an alternative representation by organizing information into entities and relationships. Entities are represented as nodes, while relationships between entities are represented as edges. This structure provides an explicit representation of connections between different pieces of information. Graph-based Retrieval-Augmented Generation extends conventional RAG by incorporating graph representations into the indexing and retrieval process. Instead of retrieving only isolated text segments, a graph-based system can identify related entities, explore their connections, and retrieve relevant subgraphs or associated textual information. Edge et al. introduced GraphRAG as an approach that constructs an entity knowledge graph from source documents and generates community summaries over groups of related entities. These summaries support query-focused summarization, particularly for questions requiring an understanding of broader relationships and themes across a document collection [3].

However, structured retrieval introduces additional challenges. Knowledge graph construction requires entity and relationship extraction, entity resolution, and graph organization. Errors during these stages can affect the quality of subsequent retrieval. Graph traversal can also introduce irrelevant information when the retrieved subgraph expands beyond the relationships necessary to answer a query. Therefore, graph-based retrieval should not be considered a universal replacement for vector-based retrieval. The two approaches represent different ways of organizing and accessing information, and their effectiveness depends on the structure of the knowledge base and the requirements of the query.

E. Multi-Hop Question Answering and Retrieval Requirements

Question answering systems traditionally focus on retrieving information that directly corresponds to a given question. Such questions can often be answered using a single relevant passage and are commonly described as single-hop questions. However, many real-world questions require information from multiple sources and involve establishing relationships between different entities or facts. These are generally referred to as multi-hop questions.

Multi-hop question answering requires a system to identify multiple pieces of evidence and combine them through a sequence of reasoning steps. The information required for the final answer may not be available in a single document. Instead, one retrieved fact may provide the information necessary to identify another relevant entity or source, which must then be retrieved to complete the reasoning process.

For instance, consider the question: "Which programming language was used in the project developed by the organization founded by a particular researcher?" Answering this question may require identifying the researcher, determining the associated organization, locating its projects, and retrieving the programming language used in the relevant project. Each intermediate relationship contributes to the final answer.

This requirement introduces several challenges for conventional RAG systems. Retrieving passages that are individually relevant may still result in incomplete evidence when the relationships between those passages are not identified. Similarly, retrieving a large number of documents does not guarantee that the information will form a coherent reasoning chain.

A multi-hop RAG system therefore needs to address several requirements:

1) Evidence identification: Retrieve the individual facts and documents required to answer the question.
2) Relationship discovery: Identify connections between entities and information distributed across different sources.
3) Sequential retrieval: Use intermediate findings to identify additional information required for subsequent reasoning steps.
4) Context integration: Combine the retrieved evidence while preserving the relationships between individual facts.
5) Reasoning completeness: Ensure that the generated response is supported by the complete set of evidence required to answer the question.
6) Evidence grounding: Maintain consistency between the retrieved information and the final generated response.

These requirements explain why multi-hop reasoning remains a challenging problem for retrieval-augmented generation. While dense and sparse retrieval provide effective mechanisms for identifying relevant information, iterative retrieval, hierarchical representations, and graph-based retrieval introduce additional capabilities for discovering and connecting evidence.

F. Summary

The development of Retrieval-Augmented Generation reflects a gradual shift from conventional passage retrieval towards more structured and reasoning-oriented retrieval architectures. Early approaches established the integration of external knowledge with language models, while subsequent developments introduced improvements in retrieval quality, query processing, contextual organization, and retrieval control.

More recent approaches incorporate hierarchical representations, feedback mechanisms, and knowledge graphs to support queries that require information beyond individual text passages. These developments are particularly relevant to multi-hop question answering, where the system must retrieve multiple pieces of evidence and establish the relationships required to derive an answer.

Understanding this evolution provides the foundation for examining individual retrieval mechanisms and their capabilities in supporting complex reasoning. The following sections discuss these approaches in greater detail, beginning with sparse, dense, and hybrid retrieval techniques.

III. TRADITIONAL RETRIEVAL TECHNIQUES: SPARSE, DENSE AND HYBRID RETRIEVAL

A. Overview of Traditional Retrieval Methods

Information retrieval is a fundamental component of Retrieval-Augmented Generation, as it determines which information is provided to the language model during response generation. The quality of the retrieved context directly influences the relevance and factual grounding of the generated answer. Traditional RAG systems primarily rely on three retrieval approaches: sparse retrieval, dense retrieval, and hybrid retrieval.

Sparse retrieval identifies relevant documents through lexical matching between query terms and document terms. Dense retrieval represents queries and documents as continuous numerical vectors and identifies relevant information based on semantic similarity. Hybrid retrieval combines these approaches to make use of both lexical and semantic information.

Although these methods differ in their underlying representations and ranking mechanisms, they share a common objective of identifying relevant information from a document collection.

Their effectiveness, however, varies depending on query characteristics, document structure, terminology, and the relationships required to answer a question.

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

B. Sparse Retrieval

Sparse retrieval is one of the established approaches in information retrieval. It represents documents and queries using terms from a vocabulary, where each dimension corresponds to a particular term. Most dimensions in these representations are zero because a document or query contains only a small subset of the complete vocabulary. Relevance is generally estimated using term occurrence, term importance, and statistical properties of the document collection.

1) Term Frequency-Inverse Document Frequency (TF-IDF)

Term Frequency-Inverse Document Frequency (TF-IDF) is a statistical weighting technique used to estimate the importance of a term within a document relative to an entire document collection. It combines two components: Term Frequency (TF), which measures how frequently a term appears in a document, and Inverse Document Frequency (IDF), which reduces the importance of terms that occur frequently across the collection.

A commonly used representation is:

$$TF\text{-}IDF(t,d) = TF(t,d) \times IDF(t)$$

where TF(t,d) represents the frequency of term t in document d, and IDF(t) represents the inverse document frequency of that term across the collection.

The IDF component assigns greater importance to terms that occur in fewer documents, while common terms receive lower weights. This helps distinguish documents containing specific terminology from those containing frequently occurring words.

Salton and Buckley discussed the effectiveness of term-weighting approaches in automatic text retrieval and established the importance of appropriately assigning weights to individual terms [12].

TF-IDF is relatively simple to implement and provides an interpretable representation of document content. However, its reliance on exact term occurrence limits its ability to identify semantically related expressions. For example, a query containing the term "automobile" may fail to retrieve a document that primarily uses the term "car", even though both terms refer to the same concept.

In RAG systems, TF-IDF can provide a useful baseline for retrieving documents containing specific keywords, technical terms, and named entities. However, it may be insufficient when queries involve paraphrasing, implicit relationships, or information expressed using different terminology.

2) Best Matching 25 (BM25)

Best Matching 25 (BM25) is a probabilistic ranking function widely used in information retrieval. It extends the traditional term-frequency-based ranking approach by incorporating term frequency saturation and document length normalization.

Robertson and Zaragoza provided a detailed discussion of the probabilistic relevance framework and the development of BM25 as an effective document-ranking method [13].

The BM25 score for a query Q and document D can be expressed as:

$$BM25(D,Q) = \Sigma\, IDF(q_i) \times [f(q_i,D)(k_1 + 1)] / [f(q_i,D) + k_1(1 - b + b \times |D|/avgdl)]$$

where:

- Q represents the input query.
- q_i represents an individual query term.
- $f(q_i,D)$ represents the frequency of query term q_i in document D.
- |D| represents the length of document D.
- avgdl represents the average document length in the collection.
- k_1 controls the term-frequency saturation.
- b controls the degree of document length normalization.
- $IDF(q_i)$ represents the inverse document frequency of the query term.

Unlike simple term-frequency scoring, BM25 prevents a term from receiving an indefinitely increasing score as its frequency increases. It also accounts for document length, reducing the tendency to favour longer documents merely because they contain more occurrences of query terms.

BM25 is particularly effective when queries contain specific keywords, technical terminology, identifiers, and named entities. Its efficiency and interpretability make it a widely used retrieval baseline in RAG applications.

However, BM25 remains dependent on lexical overlap. It does not inherently understand synonyms, contextual meaning, or relationships between entities. Consequently, it may retrieve documents containing the correct keywords without identifying documents that provide semantically relevant information expressed differently.

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

3) Neural Sparse Retrieval

Traditional sparse retrieval methods rely primarily on manually defined term-weighting mechanisms. More recent approaches incorporate neural language models to improve sparse representations while retaining the advantages of lexical indexing.

SPLADE (Sparse Lexical and Expansion Model for First Stage Ranking) is an example of a neural sparse retrieval approach. It uses contextualized language representations to generate sparse term-weighted vectors and supports document expansion, allowing related terms to contribute to retrieval even when they are not explicitly present in the original text [14].

Unlike conventional TF-IDF and BM25, neural sparse models can learn term importance and expand representations using contextual information. At the same time, they preserve sparse representations that can be efficiently searched using inverted indexes.

Neural sparse retrieval therefore provides an intermediate approach between traditional lexical retrieval and dense semantic retrieval. Nevertheless, its effectiveness depends on model training, vocabulary representation, and the computational cost of generating and indexing sparse representations.

C. Dense Retrieval

Dense retrieval represents queries and documents as continuous numerical vectors, commonly known as embeddings. These embeddings are generated using neural language models that learn semantic representations of text. The objective is to represent semantically related queries and documents in nearby regions of a vector space, even when they do not contain identical words.

1) Embedding-Based Retrieval

An embedding model converts an input text segment into a fixed-dimensional numerical representation:

$E(d) = [v_1, v_2, ..., v_n]$

where E(d) represents the embedding of document d and n denotes the embedding dimensionality.

Similarly, the input query is converted into a query embedding:

$E(q) = [u_1, u_2, ..., u_n]$

The retrieval system compares the query embedding with document embeddings and ranks candidate documents according to their similarity.

Unlike sparse retrieval, dense retrieval does not require exact keyword overlap. It can identify passages that express similar meanings using different words or sentence structures.

Karpukhin et al. introduced Dense Passage Retrieval (DPR), which uses a dual-encoder architecture to independently encode questions and passages into dense vectors [2]. The retriever learns representations in which relevant question-passage pairs are positioned closer together than irrelevant pairs. DPR demonstrated the effectiveness of dense representations for open-domain question answering and reported improvements over a strong BM25 baseline on several retrieval benchmarks.

Sentence-BERT, introduced by Reimers and Gurevych, is another important development in sentence embedding research. It uses Siamese and triplet network structures to generate semantically meaningful sentence representations that can be compared efficiently using similarity measures such as cosine similarity [15].

In RAG systems, dense embeddings are commonly generated for document chunks during indexing. At query time, the query is embedded using the corresponding model, and the vector index retrieves the nearest document representations.

The effectiveness of dense retrieval depends on the embedding model's ability to capture the relevant semantic information. Domain-specific terminology, ambiguous queries, and differences between the training data and target application can affect retrieval quality.

2) Similarity Measures

After converting queries and documents into dense vectors, the retrieval system requires a similarity or distance function to compare their representations. The commonly used measures include cosine similarity, dot product, and Euclidean distance.

Cosine Similarity: Cosine similarity measures the cosine of the angle between two vectors. It is calculated as:

$\text{Cosine}(q,d) = (q \cdot d) / (||q|| \times ||d||)$

where q and d represent the query and document embeddings, respectively.

It measures directional similarity and is commonly used when the orientation of the embedding vectors is more important than their magnitude.

Dot Product: The dot product is calculated as:

$\text{Dot}(q,d) = q \cdot d$

It measures the combined effect of vector direction and magnitude. Its suitability depends on how the embedding model was trained and whether vector magnitude carries useful information.

Euclidean Distance: Euclidean distance measures the straight-line distance between two vectors:

$$\text{Distance}(q,d) = \sqrt{\Sigma(q_i - d_i)^2}$$

A smaller distance indicates that the vectors are closer in the embedding space.

The choice of similarity measure should be consistent with the embedding model's training objective and the vector index configuration. Cosine similarity, dot product, and Euclidean distance should not be treated as interchangeable in every retrieval *system.*

3) Vector Indexing and Approximate Nearest Neighbor Search

Dense retrieval requires comparing a query embedding with document embeddings stored in a vector index. For small collections, exact nearest-neighbour search can be performed by calculating the similarity between the query vector and every stored vector. However, this approach becomes computationally expensive as the number of indexed documents increases.

Approximate Nearest Neighbor (ANN) search addresses this challenge by reducing the number of vectors examined during retrieval. Instead of evaluating every stored representation, ANN algorithms use indexing structures to identify a smaller set of likely nearest neighbours. This improves search efficiency, although it may introduce a trade-off between retrieval accuracy, latency, memory consumption, and indexing cost.

Several indexing approaches are commonly used:

Hierarchical Navigable Small World (HNSW): HNSW organizes vectors into a hierarchical structure of proximity graphs. Search begins at higher levels and progressively moves towards more detailed graph layers to identify nearby vectors. Malkov and Yashunin demonstrated that this approach supports efficient approximate nearest-neighbour search with strong retrieval performance [16].

Inverted File Index (IVF): IVF partitions the vector collection into clusters. During retrieval, the system identifies relevant clusters and searches within selected partitions rather than examining the entire collection. This reduces the number of vector comparisons required.

Product Quantization (PQ): Product quantization compresses high-dimensional vectors by dividing them into smaller subspaces and representing these subspaces using learned codebooks. It reduces memory requirements and supports efficient approximate distance calculations.

FAISS (Facebook AI Similarity Search) is a widely used library that provides efficient similarity search and clustering operations over dense vectors. Johnson et al. described GPU-based approaches for large-scale similarity search and demonstrated the potential of optimized nearest-neighbour algorithms and product quantization for handling very large vector collections [17].

Vector databases and vector search libraries provide practical implementations of these indexing approaches. Systems such as FAISS, Milvus, Weaviate, Chroma, and Pinecone support different combinations of vector storage, indexing, similarity search, and metadata filtering.

The choice of indexing structure depends on the size of the document collection, retrieval latency requirements, memory constraints, and the acceptable loss in nearest-neighbour accuracy.

D. Hybrid Retrieval

Sparse and dense retrieval methods have complementary strengths. Sparse retrieval is effective at identifying exact keyword matches, technical expressions, and named entities, whereas dense retrieval can identify semantically related passages even when the wording differs.

Hybrid retrieval combines multiple retrieval mechanisms to improve the coverage and relevance of retrieved information. A common implementation performs sparse and dense searches independently and then combines their candidate results into a unified ranking.

For example, a technical query containing a specific algorithm name may benefit from BM25 because exact terminology is important. At the same time, dense retrieval may identify passages that explain the algorithm using alternative terminology or related concepts. Combining both result sets can provide broader retrieval coverage.

One approach to combining rankings is Reciprocal Rank Fusion (RRF). Cormack et al. introduced RRF as a method for combining document rankings produced by multiple retrieval systems [18]. It assigns scores based on the reciprocal ranks of documents in the individual result lists.

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

The RRF score can be expressed as:

$RRF(d) = \Sigma\ 1 / (k + rank_i(d))$

where $rank_i(d)$ represents the rank of document d in the result list produced by retriever i, and k is a constant used to control the influence of individual ranks.

Documents appearing near the top of multiple result lists receive higher combined scores. This makes RRF useful when combining sparse and dense retrieval rankings without directly comparing their raw scores, which may have different numerical scales.

Hybrid retrieval can improve the likelihood of retrieving relevant information across different query types. However, combining rankings does not automatically ensure that the retrieved passages contain all the evidence required for a complex question. If both retrieval methods fail to identify an important intermediate document or relationship, the final result may still be incomplete.

E. Comparative Analysis of Sparse, Dense and Hybrid Retrieval

The three retrieval approaches differ in their representation mechanisms, relevance calculations, and suitability for different query characteristics.

Characteristic	Sparse Retrieval	Dense Retrieval	Hybrid Retrieval
Representation	Sparse term vectors	Dense embeddings	Combination of sparse and dense representations
Main mechanism	Lexical matching	Semantic similarity	Combined lexical and semantic ranking
Common methods	TF-IDF, BM25	DPR, embedding models	BM25 + vector search
Exact keyword matching	Strong	Depends on embedding model	Strong
Semantic matching	Limited	Strong	Strong
Synonym handling	Limited in traditional methods	Generally better	Benefits from both approaches
Interpretability	Relatively high	Comparatively lower	Depends on component retrievers
Indexing	Inverted index	Vector index	Multiple indexes
Main limitation	Lexical mismatch	Weak exact-term sensitivity in some cases	Additional retrieval and ranking complexity

The selection of a retrieval approach depends on the characteristics of the underlying knowledge source and the requirements of the application. Sparse retrieval is useful when exact terminology and identifiers are important. Dense retrieval is more suitable when semantic relationships and variations in wording need to be captured. Hybrid retrieval provides a combination of both mechanisms, although it introduces additional indexing and ranking requirements.

From a multi-hop reasoning perspective, these methods primarily support the identification of relevant evidence. Sparse retrieval may identify documents containing explicit entity names or relationship terms, while dense retrieval may identify semantically related information expressed using different terminology. Hybrid retrieval can improve candidate coverage by combining these capabilities. However, none of these retrieval mechanisms inherently represents the sequence of relationships connecting multiple pieces of evidence. Retrieving several individually relevant passages does not necessarily establish how those passages are connected or whether they collectively form a complete reasoning chain. This limitation motivates the investigation of query transformation, iterative retrieval, reranking, and structured knowledge retrieval techniques discussed in the following sections.

F. Summary

Sparse, dense, and hybrid retrieval form the foundation of many Retrieval-Augmented Generation systems. Sparse methods such as TF-IDF and BM25 provide efficient lexical matching, while dense retrieval uses learned embeddings to capture semantic similarity. Hybrid retrieval combines these mechanisms to improve retrieval coverage across different query types.

The development of ANN indexing techniques has further enabled dense retrieval to operate efficiently over large document collections. Nevertheless, improving the relevance of retrieved passages does not necessarily solve the problem of multi-hop reasoning. Traditional retrieval methods primarily identify candidate evidence, whereas complex questions require the system to discover relationships, retrieve intermediate information, and integrate evidence across multiple sources.

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

These limitations provide the motivation for advanced retrieval architectures and knowledge graph-based approaches, which are examined in the subsequent sections.

IV. ADVANCED RETRIEVAL-AUGMENTED GENERATION TECHNIQUES

A. *Introduction to Advanced RAG*

Traditional Retrieval-Augmented Generation systems generally follow a fixed retrieval process in which the original user query is used to retrieve a set of relevant document chunks. Although this approach works well for straightforward questions, it may produce incomplete or irrelevant context when queries are ambiguous, contain insufficient information, or require evidence from multiple sources.

Advanced RAG techniques introduce additional processing mechanisms to improve the retrieval process before the retrieved information reaches the language model. These mechanisms include query transformation, document reranking, contextual enrichment, hierarchical retrieval, and iterative retrieval. Some approaches also introduce feedback mechanisms that evaluate retrieval quality and determine whether additional information is required.

Unlike traditional retrieval methods, which primarily focus on matching queries with documents, advanced RAG techniques attempt to improve different stages of the retrieval pipeline. Their effectiveness depends on the particular limitation they address. For example, query expansion can improve query coverage, reranking can improve the ordering of retrieved documents, and iterative retrieval can identify additional evidence based on intermediate findings.

The following subsections examine these techniques and discuss their relevance to complex question answering and multi-hop reasoning.

B. *Query Transformation and Query Expansion*

A major limitation of conventional retrieval systems is their dependence on the original user query. User questions are often short, ambiguous, or expressed differently from the terminology used in the knowledge base. Consequently, a retrieval system may fail to identify relevant documents even when the required information is available.

Query transformation addresses this problem by modifying the original query before retrieval. Depending on the approach, the query may be rewritten, expanded, decomposed into smaller questions, or represented using generated hypothetical content.

1) *Query Rewriting*

Query rewriting transforms the original question into an alternative representation that is more suitable for information retrieval. A language model can be used to identify the main intent, remove ambiguity, expand incomplete expressions, and produce a more explicit search query.

For example, consider the question:

"Which technologies were used in the projects developed by the organization founded by researcher X?"

A conventional retriever may attempt to match the entire question against individual document chunks. A rewritten query could make the relevant entities and information requirements more explicit, helping the retrieval system identify documents containing the organization, its projects, and their associated technologies.

Query rewriting is particularly useful for conversational systems, where a user's current question may depend on information provided in earlier messages. In such cases, the query can be rewritten to include the missing context before retrieval.

However, rewriting does not necessarily identify the complete sequence of intermediate facts required to answer a multi-hop question. A rewritten query may improve the initial retrieval results while still failing to retrieve information that becomes relevant only after an intermediate entity has been identified.

2) *Query Expansion*

Query expansion improves retrieval by adding related terms, concepts, or additional information to the original query. The expanded representation increases the possibility of matching documents that use terminology different from the original question.

Traditional query expansion methods use techniques such as thesauri, relevance feedback, and term association. More recent approaches use large language models to generate additional information for retrieval.

Wang et al. proposed Query2doc, a query expansion technique that uses an LLM to generate a pseudo-document based on the original query. The generated text is combined with the original query and used to improve both sparse and dense retrieval systems [20]. Their experiments demonstrated improvements in BM25 and dense retrieval across different information retrieval benchmarks.

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

Query expansion can improve document coverage when relevant information is expressed using different terms. However, generated expansion terms may introduce unrelated concepts or incorrect assumptions. If the generated content is not sufficiently aligned with the original question, the expanded query may retrieve irrelevant documents.

3) Hypothetical Document Embeddings (HyDE)

Hypothetical Document Embeddings (HyDE), proposed by Gao et al., introduces a different approach to query transformation [19]. Instead of directly embedding the original question, the system first generates a hypothetical document that attempts to answer the query. The generated document is then converted into a dense embedding and used to retrieve similar real documents from the knowledge corpus.

The process consists of the following steps:

a) The user submits a query.
b) A language model generates a hypothetical document related to the query.
c) The hypothetical document is converted into a dense vector using an embedding model.
d) The vector is compared with document embeddings stored in the vector index.
e) Similar real documents are retrieved and supplied to the language model.

An important characteristic of HyDE is that the hypothetical document does not need to be factually correct. Its purpose is to generate a semantic representation that can guide retrieval towards relevant documents. The retrieved documents are obtained from the actual knowledge collection rather than being accepted directly from the generated hypothetical content.

HyDE can improve retrieval when the original query is too short or differs substantially from the language used in relevant documents. However, the quality of retrieval still depends on the generated hypothetical document and the embedding model. It also introduces an additional generation step, which increases computational cost and latency.

From a multi-hop reasoning perspective, HyDE primarily improves the semantic representation of the initial query. It does not inherently perform sequential evidence retrieval or establish relationships between intermediate entities.

C. Reranking and Contextual Retrieval

Initial retrieval systems are generally optimized for efficiently identifying a candidate set of relevant documents from a large collection. However, the ranking produced during the first retrieval stage may not always place the most useful documents at the top. Reranking and contextual retrieval address different aspects of this problem by improving candidate selection and preserving information required to understand individual chunks.

1) Reranking

Reranking is a two-stage retrieval process in which an initial retriever identifies a candidate set of documents, followed by a more detailed relevance evaluation.

The first stage generally uses BM25, dense retrieval, or hybrid retrieval to retrieve a relatively large set of candidate chunks. In the second stage, a reranking model evaluates the relationship between the original query and each retrieved candidate. The documents are then reordered according to their estimated relevance.

Cross-encoder models are commonly used for reranking. Unlike bi-encoder models, which independently encode queries and documents, cross-encoders process the query and candidate document together. This allows the model to capture interactions between the query and document tokens during relevance estimation.

Although cross-encoders can provide more detailed relevance scores, they are computationally more expensive because each query-document pair must be evaluated separately. Therefore, reranking is generally applied to a smaller candidate set obtained through an initial retrieval stage.

ColBERT, proposed by Khattab and Zaharia, introduces a late-interaction architecture that retains token-level representations of queries and documents. Instead of compressing each document into a single vector, ColBERT independently encodes query and document tokens and performs a finer-grained interaction during relevance scoring [21]. This approach provides a balance between the efficiency of independent encoding and the expressive power of detailed query-document interaction.

Reranking can improve the ordering of retrieved evidence and reduce the inclusion of irrelevant passages in the final context. However, it cannot recover evidence that was not included in the initial candidate set. Consequently, a high-quality reranker may still produce incomplete results when the initial retrieval stage fails to identify important documents.

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

2) Contextual Retrieval

Document chunking is widely used in RAG systems because it allows large documents to be divided into smaller units that can be indexed and retrieved independently. However, chunking can remove information that is necessary to understand the meaning of an individual segment.

For example, a document may contain a paragraph describing a particular research method without explicitly mentioning the name of the research project. If the paragraph is retrieved independently, the system may not have sufficient context to identify what the method refers to.

Contextual retrieval attempts to address this limitation by adding relevant document-level information to individual chunks before indexing. The additional context may include the document title, section heading, surrounding information, or a short description of the chunk's relationship to the complete document.

Anthropic introduced Contextual Retrieval as an approach that combines contextual embeddings with contextual BM25. In this method, contextual information is added to chunks before they are indexed, allowing both semantic and lexical retrieval to make use of the additional context [23].

Contextual retrieval can improve the identification of document segments that would otherwise be difficult to interpret independently. However, the approach introduces preprocessing requirements and additional storage considerations. The quality of contextual descriptions also depends on the accuracy of the generated context.

For multi-hop question answering, contextual retrieval can help preserve information about the source and subject of individual passages. Nevertheless, adding context to chunks does not automatically establish relationships between entities distributed across separate documents.

D. Hierarchical and Multi-Level Retrieval

Conventional retrieval systems generally operate on individual document chunks. Although this provides access to detailed information, it may not capture the broader context of lengthy documents or collections containing multiple related sections.

Hierarchical retrieval addresses this limitation by organizing information at different levels of abstraction. Instead of representing the knowledge base exclusively through individual text chunks, the system creates higher-level representations such as section summaries, document summaries, or clusters of related passages.

1) RAPTOR

Recursive Abstractive Processing for Tree-Organized Retrieval (RAPTOR), proposed by Sarthi et al., introduces a hierarchical retrieval architecture that recursively clusters and summarizes text chunks [11].

The approach begins by dividing source documents into smaller text segments and generating embeddings for them. Similar segments are grouped into clusters, and language models generate summaries representing the information contained within each cluster. The summaries are subsequently embedded and clustered again, producing additional levels of abstraction.

This process creates a tree-like structure in which the leaf nodes contain original text chunks, while higher-level nodes contain summaries of increasingly larger groups of information.

During retrieval, the system can retrieve information from different levels of the tree. Detailed questions may benefit from retrieving original text chunks, whereas broader questions may benefit from retrieving higher-level summaries.

RAPTOR is particularly relevant to questions requiring information distributed across lengthy documents because it provides access to both local details and broader contextual representations. Its hierarchical organization can also support certain forms of multi-step reasoning by exposing information that may not be captured through individual chunk retrieval.

However, summarization introduces the possibility of information loss. Important details, specific entities, and intermediate relationships may be omitted when information is compressed into higher-level summaries. Furthermore, constructing the hierarchical representation requires additional preprocessing and computational resources.

E. Iterative and Multi-Step Retrieval

One of the major limitations of conventional RAG is its reliance on retrieving all relevant information using the original query. This becomes problematic when the information required for a later retrieval step depends on the results obtained during an earlier step.

Iterative retrieval addresses this limitation by repeatedly performing retrieval based on intermediate findings. Instead of retrieving all evidence at once, the system retrieves an initial set of documents, processes the information, identifies additional information requirements, and performs subsequent retrieval operations.

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

1) Retrieval with Query Decomposition

Query decomposition divides a complex question into smaller sub-questions that can be processed individually. Each sub-question represents a specific information requirement associated with the original query.

For example, consider the question:

"Which programming language was used in the project developed by the organization founded by researcher X?"

The question can be decomposed into the following information requirements:

1. Identify the organization founded by researcher X.
2. Identify the projects developed by that organization.
3. Determine the programming language used in the relevant project.

The system retrieves information for each sub-question and combines the intermediate findings to generate the final answer.

Query decomposition can make complex questions easier to process by converting them into smaller retrieval tasks. However, the quality of the final response depends on whether the decomposition correctly identifies the required reasoning steps and whether the retrieved information is sufficient to establish the relationships between them.

2) Interleaving Retrieval with Chain-of-Thought Reasoning (IRCoT)

Trivedi et al. introduced Interleaving Retrieval with Chain-of-Thought Reasoning (IRCoT), an approach specifically designed for knowledge-intensive multi-step question answering [22].

The central observation behind IRCoT is that the information required for subsequent retrieval may depend on what has already been retrieved and derived. A single retrieval operation based only on the original question may therefore be insufficient.

IRCoT addresses this problem by alternating between reasoning and retrieval. The system generates an intermediate reasoning step, uses that step to guide the next retrieval operation, and incorporates the retrieved information into subsequent reasoning.

The process can be represented as follows:

a) The system receives a complex question.
b) The language model generates an initial reasoning step.
c) The reasoning step is used to formulate a retrieval query.
d) Relevant documents are retrieved and incorporated into the reasoning context.
e) The language model generates the next reasoning step.
f) Retrieval and reasoning continue until sufficient information is obtained to answer the original question.

IRCoT was evaluated on multi-hop question-answering datasets including HotpotQA, 2WikiMultiHopQA, MuSiQue, and IIRC. The authors reported improvements in both retrieval performance and downstream question-answering accuracy compared with one-step retrieval approaches [22].

This approach is particularly important because it explicitly connects retrieval operations with intermediate reasoning results. Rather than assuming that all relevant information can be retrieved from the original query, it allows each retrieval operation to be guided by the knowledge acquired in previous steps.

However, iterative retrieval also introduces additional model calls and retrieval operations. Errors in an intermediate reasoning step can influence subsequent queries, potentially propagating incorrect assumptions through the reasoning process. Therefore, intermediate evidence verification and retrieval stopping criteria remain important considerations.

F. Adaptive and Feedback-Based Retrieval

Advanced retrieval systems increasingly incorporate mechanisms that evaluate whether retrieval is necessary and whether the retrieved information is sufficiently relevant to support generation. These approaches move beyond fixed retrieval pipelines by introducing feedback and conditional decision-making.

1) Self-RAG

Self-RAG, proposed by Asai et al., introduces a self-reflective mechanism that allows a language model to determine when retrieval is required and evaluate the relevance and usefulness of retrieved information [9].

The framework uses special reflection tokens to guide the retrieval and generation process. These include signals associated with retrieval decisions, document relevance, support for generated statements, and overall response quality.

Rather than retrieving a fixed number of documents for every query, Self-RAG enables retrieval to be performed on demand. The model can evaluate retrieved passages and reflect on whether its generated response is adequately supported by the available evidence.

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

This approach improves the flexibility of RAG systems because retrieval decisions are influenced by the requirements of the query and the quality of available information.

For multi-hop reasoning, adaptive retrieval may help determine when additional evidence is required. However, the reflection mechanism does not automatically guarantee that all intermediate relationships have been discovered. Its effectiveness depends on the model's ability to assess retrieval relevance and generation quality accurately.

2) Corrective Retrieval-Augmented Generation (CRAG)

Corrective Retrieval-Augmented Generation (CRAG), proposed by Yan et al., addresses situations in which the initial retrieval process returns irrelevant or insufficient information [10].

The framework introduces a retrieval evaluator that assesses the quality of retrieved documents and produces a confidence-based assessment. Depending on this assessment, the system can trigger different corrective actions.

When the retrieved documents are considered reliable, the system can proceed with the available information. When retrieval quality is uncertain or insufficient, additional retrieval mechanisms can be activated. The original CRAG framework also explores the use of web search to supplement information obtained from a static knowledge corpus.

Another component of CRAG is a knowledge refinement process that decomposes retrieved documents into smaller units and filters information according to its relevance. This helps the system focus on useful information while reducing irrelevant context.

CRAG demonstrates the importance of evaluating retrieval results rather than assuming that the initial retrieval operation is always successful. Nevertheless, its effectiveness depends on the quality of the retrieval evaluator and the corrective retrieval mechanisms.

In multi-hop applications, corrective retrieval can help address missing or weak evidence. However, assessing the relevance of individual documents is different from verifying whether the complete set of retrieved documents forms a valid reasoning chain.

G. Comparative Analysis of Advanced RAG Techniques

Advanced RAG approaches address different limitations of conventional retrieval systems. Some improve the representation of the original query, while others refine document rankings, preserve contextual information, or introduce repeated retrieval operations.

Technique	Primary objective	Main mechanism	Multi-hop relevance
Query rewriting	Improve query clarity	Reformulates the original question	Helps identify relevant information
Query expansion	Increase query coverage	Adds related terms or generated content	May improve evidence discovery
HyDE	Improve semantic retrieval	Embeds a hypothetical document	Improves initial retrieval representation
Reranking	Improve candidate ordering	Re-evaluates retrieved documents	Helps prioritize relevant evidence
Contextual retrieval	Preserve chunk context	Adds document-level context	Helps identify relationships within source documents
RAPTOR	Support multi-level retrieval	Recursive clustering and summarization	Supports broader contextual retrieval
Query decomposition	Break down complex questions	Generates smaller sub-questions	Explicitly separates information requirements
IRCoT	Connect retrieval with reasoning	Alternates reasoning and retrieval	Directly supports sequential evidence discovery
Self-RAG	Control retrieval and generation	Reflection-based retrieval decisions	Supports conditional evidence acquisition
CRAG	Correct weak retrieval results	Evaluates and refines retrieved information	Helps address incomplete or unreliable retrieval

The techniques discussed above should not be considered mutually exclusive. A single RAG architecture can combine query rewriting, hybrid retrieval, reranking, hierarchical indexing, and iterative retrieval. Their combination depends on the complexity of the application, available computational resources, and the quality requirements of the generated responses.

From the perspective of multi-hop reasoning, query transformation and reranking primarily improve the identification and ordering of relevant evidence. Hierarchical retrieval provides access to information at different levels of abstraction, while iterative retrieval allows subsequent searches to depend on intermediate findings. Feedback-based approaches further introduce mechanisms for evaluating retrieval quality and deciding whether additional information is required.

H. Limitations and Research Implications

Although advanced RAG techniques improve several aspects of conventional retrieval, they introduce additional design and implementation challenges.

First, query transformation methods depend on the quality of generated queries or hypothetical documents. Incorrect expansions may introduce irrelevant information and negatively affect retrieval performance.

Second, reranking and contextual enrichment increase computational requirements. Reranking requires additional model inference, while contextual retrieval introduces preprocessing and storage overhead.

Third, hierarchical retrieval relies on clustering and summarization. Information loss during summarization may affect the retrieval of specific entities, facts, and relationships.

Fourth, iterative retrieval introduces dependencies between successive retrieval and reasoning operations. Errors in intermediate results may propagate into later steps, affecting the correctness of the final response.

Finally, feedback-based retrieval requires reliable mechanisms for evaluating evidence quality. A document may be relevant to an individual query while still being insufficient to establish the complete relationship required by a multi-hop question.

These limitations indicate that improving individual retrieval operations is not sufficient to guarantee reliable multi-hop reasoning. A system must also preserve relationships between retrieved evidence, identify missing intermediate information, and ensure that the final response is supported by the complete reasoning process.

I. Summary

Advanced RAG techniques extend conventional retrieval through query transformation, reranking, contextual enrichment, hierarchical organization, iterative retrieval, and feedback-based control. These approaches address different weaknesses of traditional RAG and provide greater flexibility in handling complex information needs.

Among these techniques, iterative retrieval and query decomposition are particularly relevant to multi-hop question answering because they allow retrieval to be guided by intermediate information requirements. RAPTOR provides hierarchical access to information, while Self-RAG and CRAG introduce mechanisms for evaluating and controlling retrieval quality.

However, these approaches primarily operate on retrieved documents, chunks, or hierarchical summaries. They do not necessarily provide an explicit representation of relationships between entities across a knowledge collection. This limitation motivates the use of Knowledge Graph-based Retrieval-Augmented Generation, where information is represented through entities and their relationships to support structured retrieval and connected evidence discovery.

V. KNOWLEDGE GRAPH-BASED RETRIEVAL-AUGMENTED GENERATION

A. Introduction to Knowledge Graph-Based RAG

Traditional Retrieval-Augmented Generation systems primarily retrieve information using lexical matching or semantic similarity. Although these approaches are effective for identifying relevant text passages, they do not explicitly represent the relationships between entities and facts distributed across different documents. This limitation becomes particularly important when answering questions that require connecting multiple pieces of information.

Knowledge Graph-based Retrieval-Augmented Generation addresses this limitation by incorporating structured representations of entities and their relationships into the retrieval process. A Knowledge Graph (KG) represents information through nodes and edges, where nodes correspond to entities and edges represent relationships between them. Additional attributes can be associated with nodes and edges to preserve descriptions, properties, source information, and other relevant details.

A conventional knowledge graph is commonly represented using triples of the form:

(h, r, t)

where h represents the head entity, r represents the relationship, and t represents the tail entity.

For example, the following triples can represent information about an organization and its research activities:

- (Researcher A, founded, Organization X)
- (Organization X, developed, Project Y)
- (Project Y, uses, Technology Z)

These triples form a connected representation of information. A query concerning the technology used in a project developed by an organization founded by a particular researcher can be answered by following the corresponding relationships across the graph.

Unlike conventional vector retrieval, which identifies relevant passages based on similarity, graph-based retrieval can exploit the explicit connections between entities. This makes knowledge graphs particularly useful for applications involving multi-hop question answering, relationship discovery, and complex information retrieval.

Peng et al. presented a broader overview of GraphRAG methodologies and organized the workflow into graph-based indexing, graph-guided retrieval, and graph-enhanced generation [24]. These stages provide a useful foundation for examining how knowledge graphs are incorporated into RAG systems.

B. Knowledge Graph Construction

The effectiveness of graph-based RAG depends heavily on the quality of the underlying knowledge graph. Unlike vector indexes, which can be constructed by embedding document chunks, knowledge graph construction requires the identification of entities and the relationships connecting them.

A typical graph construction pipeline consists of document processing, entity extraction, relationship extraction, entity resolution, graph construction, and indexing.

1) Entity Extraction

Entity extraction identifies important objects or concepts within a text collection. These entities may include people, organizations, locations, products, research papers, technologies, events, and other domain-specific concepts.

For example, consider the sentence:

"Organization X developed Project Y using Python and PostgreSQL."

An entity extraction system may identify the following entities:

- Organization X
- Project Y
- Python
- PostgreSQL

Large Language Models can be used to identify entities from unstructured text by following predefined extraction instructions. Alternatively, traditional Natural Language Processing approaches such as Named Entity Recognition (NER) can identify entities using trained sequence-labeling models.

The extracted entities become candidate nodes in the knowledge graph. Their descriptions and source references may also be retained to provide additional context during retrieval.

However, entity extraction is not always reliable. An extraction model may fail to identify domain-specific terms, assign incorrect entity types, or extract irrelevant expressions as entities. These errors can affect the structure and usefulness of the resulting graph.

2) Relationship Extraction

Relationship extraction identifies semantic connections between entities appearing in the same or related text segments. The objective is to represent the information contained in natural language as structured relationships.

For example, the sentence:

"Organization X developed Project Y using Python."

can be represented through the following relationships:

- Organization X → developed → Project Y
- Project Y → uses → Python

The extracted relationships form the edges of the knowledge graph.

Relationship extraction can be performed using rule-based methods, supervised relation extraction models, dependency parsing, or LLM-based extraction. LLM-based methods provide flexibility when processing unstructured documents containing complex sentence structures and domain-specific relationships.

Microsoft GraphRAG uses LLM-based extraction to identify entities and relationships from text units. The extracted information is subsequently used to construct an entity knowledge graph and generate additional graph-level representations [3].

The accuracy of relationship extraction is particularly important for multi-hop reasoning. If an important relationship is omitted or incorrectly extracted, the system may fail to identify the path connecting the relevant entities. Similarly, an incorrectly generated relationship can introduce misleading connections into the graph.

3) Entity Resolution and Knowledge Integration

The same real-world entity may appear in multiple documents using different names, abbreviations, or descriptions. Without appropriate entity resolution, a graph construction process may create multiple nodes for the same underlying entity.

For example, the expressions "International Business Machines" and "IBM" may refer to the same organization. If these expressions are represented as independent nodes, the graph may contain disconnected information that should have been connected.

Entity resolution attempts to identify and merge different references to the same entity. This can involve string similarity, contextual similarity, embedding-based matching, domain-specific rules, and LLM-assisted entity alignment.

Knowledge integration further combines information extracted from multiple sources into a consistent graph representation. Source references and provenance information should ideally be preserved so that retrieved relationships can be traced back to their original documents.

Entity resolution improves graph connectivity and reduces unnecessary duplication. However, incorrect entity merging can be equally problematic because it may combine information belonging to different real-world entities.

4) Graph Construction and Indexing

After entities and relationships have been extracted and aligned, they are organized into a graph structure. Each entity becomes a node, while the relationships between entities become edges.

A graph can also store additional attributes such as entity descriptions, relationship confidence, document identifiers, timestamps, and source text references.

The resulting graph may be stored using a graph database or an in-memory graph representation. Graph databases provide mechanisms for storing connected data and executing queries involving relationships, paths, and neighborhood exploration.

The graph construction stage may also include additional processing, such as community detection, entity summarization, graph embedding generation, and the creation of indexes for efficient retrieval.

An important design consideration is maintaining the connection between structured graph elements and their original textual evidence. A graph edge may represent a relationship extracted from a particular sentence or paragraph. Retaining the source text allows the system to retrieve supporting evidence rather than relying exclusively on an extracted triple.

C. Graph-Based Retrieval Mechanisms

Once a knowledge graph has been constructed, the retrieval system must identify the graph elements relevant to a given query. Unlike vector retrieval, which primarily ranks text chunks according to similarity, graph retrieval can use entity matching, relationship exploration, path search, subgraph selection, and community-level representations.

1) Entity-Based Retrieval

Entity-based retrieval identifies entities mentioned in or associated with the user query and uses them as starting points for graph exploration.

The process generally involves identifying query entities, matching them with graph nodes, and retrieving their associated relationships and neighboring entities.

For example, a query containing the name of a researcher may identify the corresponding researcher node. The system can then retrieve connected organizations, research projects, publications, and other associated entities.

Entity-based retrieval is useful when a query contains recognizable names or concepts that can be matched to nodes in the graph. However, it may retrieve too much information when an entity has a large number of connections. Entity disambiguation is also necessary when multiple entities share similar names.

2) Path-Based Retrieval

Path-based retrieval focuses on identifying sequences of relationships connecting two or more entities in a knowledge graph.

A path can be represented as:

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

$P = (v_1, e_1, v_2, e_2, \ldots, e_n, v_{n+1})$

where v represents a node and e represents an edge connecting consecutive nodes.

For example, consider the following path:

Researcher A → founded → Organization X → developed → Project Y → uses → Technology Z

This path provides an explicit representation of the relationships required to connect the researcher to the technology.

Path-based retrieval is particularly relevant to multi-hop question answering because it provides a structured way to identify intermediate entities and relationships. The retrieved path can be converted into textual evidence and supplied to a language model for answer generation.

However, graph traversal may produce multiple candidate paths between entities. Some paths may be irrelevant, overly long, or based on incorrect relationships. Therefore, path selection requires relevance estimation and appropriate traversal constraints.

3) Subgraph Retrieval

Subgraph retrieval identifies a connected portion of the knowledge graph that contains the information relevant to a query.

Instead of retrieving individual nodes or isolated relationships, the system retrieves a collection of connected entities and edges that collectively represent the relevant information.

For example, a question concerning an organization's research activities may require information about the organization, its projects, associated researchers, and technologies. A subgraph containing these connected entities can provide a broader representation of the evidence required to answer the question.

Subgraph retrieval can support multi-hop reasoning because it preserves relationships between multiple entities. It also allows the retrieval system to provide a structured context rather than a collection of independently ranked text passages.

However, subgraph selection introduces challenges involving graph size, traversal depth, node relevance, and context-window limitations. A large subgraph may contain unnecessary relationships, while an excessively restricted subgraph may exclude important intermediate evidence.

4) Community-Based Retrieval

Community-based retrieval organizes related graph entities into groups based on their structural connections. Community detection algorithms identify groups of nodes that are more densely connected internally than with the rest of the graph.

One commonly used approach is the Leiden algorithm, which identifies communities by optimizing graph modularity while addressing connectivity issues associated with some community detection methods.

Microsoft GraphRAG uses hierarchical community detection to organize its entity graph. It generates community summaries that represent the information contained within different groups of related entities. These summaries are subsequently used to support query-focused summarization [3].

Community-based retrieval is particularly useful for questions requiring an understanding of broader themes, relationships, or patterns across a large document collection.

However, community summaries operate at a higher level of abstraction and may omit specific facts or relationships required for detailed multi-hop questions. Consequently, community-level retrieval is often more appropriate for global information discovery than for questions requiring precise entity-to-entity reasoning.

D. Representative Knowledge Graph-Based RAG Architectures

Several research systems have explored different ways of integrating knowledge graphs into retrieval-augmented generation. Although these approaches share the use of structured knowledge, they differ in graph construction, retrieval strategies, and the extent to which graph structures participate in reasoning.

1) Microsoft GraphRAG

Microsoft GraphRAG, introduced by Edge et al., is a graph-based approach designed to improve question answering over large document collections, particularly for queries requiring an understanding of information across an entire corpus [3].

The system constructs an entity knowledge graph by extracting entities and relationships from source documents. It then applies hierarchical community detection to organize related entities and generates community summaries to represent the information within each group.

The original approach introduced two important retrieval modes:

- Local Search: Local Search focuses on specific entities and their surrounding relationships. It retrieves information associated with relevant entities and their graph neighborhoods, making it useful for questions involving particular people, organizations, and related concepts.
- Global Search: Global Search uses community summaries to answer questions concerning broader themes and patterns across the document collection. Instead of relying exclusively on individual text passages, it aggregates information from multiple community-level summaries.

Microsoft's subsequent GraphRAG implementation also documents additional query modes, including DRIFT Search and Basic Search, extending the available retrieval configurations [25].

GraphRAG is particularly useful for global sensemaking tasks, where relevant information is distributed across large collections of documents. However, its graph construction process can introduce considerable computational costs because entity extraction, relationship extraction, summarization, and community generation require substantial processing.

Furthermore, community-based retrieval does not automatically guarantee that every retrieved relationship is necessary or sufficient for a particular multi-hop question. The quality of the final response still depends on the retrieved evidence and the language model's ability to interpret it correctly.

2) LightRAG

Guo et al. proposed LightRAG as a graph-enhanced retrieval framework designed to improve contextual awareness while maintaining efficient retrieval [26]. Unlike approaches that rely exclusively on flat document representations, LightRAG incorporates graph structures into the indexing and retrieval process. It combines entity and relationship information with vector representations to support retrieval at different levels.

Its dual-level retrieval mechanism distinguishes between:

- Low-level retrieval: Focuses on specific entities and their associated relationships.
- High-level retrieval: Focuses on broader concepts and related groups of information.

The framework combines graph-based information with vector retrieval to identify relevant entities, relationships, and textual context.

LightRAG also introduces an incremental update mechanism to support the integration of newly available information without requiring the entire knowledge base to be reconstructed.

This architecture is useful when queries require both specific entity-level information and broader contextual relationships. However, its effectiveness depends on the quality of the extracted graph and the relevance of the retrieved information. As with other graph-based approaches, graph construction errors and irrelevant neighborhood expansion can affect answer quality.

3) HippoRAG

Gutiérrez et al. introduced HippoRAG, a retrieval framework inspired by the hippocampal indexing theory of human long-term memory [27].

The approach combines LLMs, knowledge graphs, and Personalized PageRank to retrieve relevant information from interconnected knowledge structures. The framework extracts entities and relationships from documents and organizes them into a knowledge graph. During retrieval, relevant query entities are identified, and graph propagation is used to determine the importance of related nodes. Personalized PageRank helps distribute relevance across connected parts of the graph, allowing the system to identify information that may not be directly matched by the original query.

An important feature of HippoRAG is its focus on integrating information across documents. By using graph connectivity, it can identify related information through shared entities and associations.

The authors evaluated HippoRAG on multi-hop question-answering tasks and reported improvements over several existing retrieval approaches. They also demonstrated that its graph-based retrieval could reduce the computational requirements associated with repeated retrieval in certain settings [27].

HippoRAG is particularly relevant to this survey because it demonstrates how graph propagation can support multi-hop evidence discovery without requiring the system to perform a separate LLM-based reasoning step for every retrieval operation.

Nevertheless, its performance depends on the quality of the graph representation, entity linking, and relevance propagation. Highly connected but weakly relevant nodes may also influence retrieval if graph propagation is not sufficiently constrained.

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

4) GNN-RAG

Mavromatis and Karypis introduced GNN-RAG, which combines Graph Neural Networks (GNNs) with LLM-based generation for knowledge graph question answering [28].

Unlike approaches that depend primarily on LLM-based graph traversal, GNN-RAG uses a GNN to estimate the relevance of graph nodes and retrieve candidate answers from a knowledge graph.

The framework follows a two-stage process. First, the GNN identifies relevant answer candidates by reasoning over a dense knowledge graph. It then extracts shortest paths connecting the entities mentioned in the question to the candidate answers. These paths are converted into textual representations and supplied to an LLM for final answer generation.

The approach separates graph-level retrieval from natural language generation. The GNN performs structured graph reasoning, while the LLM interprets the retrieved paths and generates the response.

The authors evaluated GNN-RAG using WebQSP and CWQ benchmarks and reported improvements on multi-hop and multi-entity questions compared with several LLM-based retrieval approaches [28].

This architecture is particularly relevant because it explicitly incorporates graph reasoning into the retrieval stage rather than using the knowledge graph only as a source of additional context.

However, GNN-RAG requires a structured knowledge graph and a trained graph neural network. Its applicability may therefore depend on the availability of suitable graph data, training resources, and domain-specific knowledge representations.

5) Knowledge Augmented Generation (KAG)

Liang et al. proposed Knowledge Augmented Generation (KAG), a framework designed to combine knowledge graphs and vector retrieval for professional-domain question answering [29].

The approach addresses limitations associated with conventional vector similarity retrieval, particularly when questions involve numerical information, temporal relationships, domain-specific rules, and logical constraints.

KAG integrates structured knowledge and original textual information through a mutual-indexing mechanism. This allows the system to retrieve relevant graph knowledge together with the supporting text chunks.

Another important component is its logical-form-guided hybrid reasoning engine. Complex questions can be represented through structured logical forms that guide retrieval and reasoning operations.

The framework also introduces knowledge alignment mechanisms to improve the consistency between structured knowledge and textual information.

KAG is particularly relevant to multi-hop reasoning because it combines retrieval from structured and unstructured sources with explicit logical reasoning. The authors reported improvements on multi-hop question-answering benchmarks, including HotpotQA and 2WikiMultiHopQA [29].

However, its implementation involves additional components for knowledge representation, alignment, logical-form generation, and reasoning. This introduces architectural complexity and requires the underlying knowledge base to be sufficiently accurate and well organized.

6) Knowledge Graph-Guided Retrieval-Augmented Generation (KG²RAG)

Zhu et al. introduced KG²RAG, a framework that uses knowledge graphs to improve the retrieval and organization of text chunks [30].

The approach begins with conventional semantic retrieval to identify relevant seed chunks. It then uses a knowledge graph to expand the retrieved set by identifying chunks connected through fact-level relationships.

After expansion, the system organizes the retrieved chunks using graph relationships to produce a more coherent contextual representation.

The framework consists of three major operations:

1. Seed retrieval: Identify initial relevant text chunks using semantic retrieval.
2. KG-guided expansion: Retrieve additional chunks connected to the seed chunks through knowledge graph relationships.
3. KG-based organization: Arrange the retrieved chunks according to their relationships to improve contextual coherence.

KG²RAG is an important example of hybrid retrieval because it does not replace semantic retrieval with graph traversal. Instead, it uses semantic retrieval to identify starting points and graph relationships to expand and organize the retrieved evidence.

The authors evaluated the framework on HotpotQA and its variants and reported improvements in retrieval and response quality over the evaluated RAG baselines [30].

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

This approach is particularly relevant to multi-hop question answering because it attempts to preserve relationships between retrieved chunks rather than treating each chunk as an independent result.

However, its effectiveness depends on the quality of the constructed graph and the relationships connecting individual chunks. Incorrect or incomplete graph edges can affect evidence expansion and organization.

E. Comparative Analysis of Graph-Based RAG Approaches

The graph-based approaches discussed above demonstrate different ways of incorporating structured knowledge into retrieval and generation.

Architecture	Graph representation	Main retrieval mechanism	Multi-hop reasoning contribution
Microsoft GraphRAG	Entity graph and communities	Local and community-based retrieval	Supports entity relationships and corpus-level reasoning
LightRAG	Entity and relationship graph with vectors	Dual-level graph and vector retrieval	Connects low-level entities with broader concepts
HippoRAG	Knowledge graph	Personalized PageRank	Propagates relevance across connected entities
GNN-RAG	Structured knowledge graph	GNN-based candidate retrieval and shortest paths	Uses graph neural reasoning to identify answer paths
KAG	Integrated knowledge graph and text chunks	Logical-form-guided hybrid retrieval	Supports structured and logical multi-hop reasoning
KG²RAG	Fact-level relationships between chunks	Semantic seed retrieval and graph-guided expansion	Expands and organizes related evidence

These architectures should not be interpreted as direct competitors using identical retrieval objectives. Microsoft GraphRAG emphasizes corpus-level understanding and community summaries, whereas GNN-RAG focuses on graph-based answer candidate retrieval. HippoRAG uses graph propagation for information retrieval, while KAG incorporates logical reasoning and structured knowledge alignment.

Similarly, LightRAG and KG²RAG demonstrate different approaches to combining graph structures with conventional retrieval. Their architectures illustrate that knowledge graphs can support multiple retrieval functions, including entity discovery, evidence expansion, relationship organization, and reasoning-path identification.

The comparison also indicates that the presence of a knowledge graph alone is not sufficient to determine the reasoning capability of a RAG system. The retrieval mechanism, graph traversal strategy, relationship representation, and evidence integration process must also be considered.

F. Challenges in Knowledge Graph-Based RAG

Despite their advantages, knowledge graph-based RAG systems introduce several challenges that affect their reliability, scalability, and practical deployment.

1) Graph Construction Cost: Entity extraction, relationship extraction, entity resolution, and graph indexing can require substantial computational resources. LLM-based extraction can improve flexibility but may increase indexing latency and operational cost.

2) Entity and Relationship Extraction Errors: Incorrect entities and relationships can introduce noise into the graph. Missing relationships may disconnect information that should be connected, while incorrect relationships may create misleading paths.

3) Entity Resolution and Ambiguity: Different references to the same entity may result in duplicate nodes, whereas incorrectly merging distinct entities can produce false relationships. Maintaining consistent entity identities across documents remains an important challenge.

4) Graph Expansion and Retrieval Noise: Traversing multiple graph edges can retrieve information that is structurally connected but not relevant to the query. Increasing traversal depth may improve evidence coverage, but it can also introduce unnecessary nodes and relationships.

5) Graph Scalability: Large knowledge graphs may contain millions of entities and relationships. Efficient graph indexing, traversal, subgraph selection, and storage become increasingly important as the knowledge base grows.

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

6) Dynamic Knowledge Updates: Knowledge sources may change over time. Updating graph nodes, relationships, entity descriptions, and community summaries while maintaining consistency is a non-trivial task.
7) Context Window Limitations: Even when a graph contains all the required information, the retrieved subgraph may be too large to include directly in the LLM context. Graph pruning and evidence prioritization are therefore necessary.
8) Evidence Provenance and Verification: Extracted relationships should be traceable to their source documents. Without provenance, it becomes difficult to verify whether a retrieved edge is supported by the original evidence.
9) Reasoning-Chain Completeness: A graph may contain several relevant entities and relationships without providing a complete path connecting the information required for an answer. Retrieval systems must distinguish between partial evidence and a complete reasoning chain.

These challenges indicate that graph-based RAG requires more than the construction of a connected entity network. Reliable multi-hop question answering depends on accurate graph construction, relevant retrieval, controlled graph expansion, and preservation of supporting evidence.

G. Summary

Knowledge Graph-based Retrieval-Augmented Generation extends conventional RAG by representing information through entities and their relationships. This representation enables retrieval systems to explore connected information, identify intermediate entities, and retrieve structured evidence distributed across multiple documents.

Different graph-based architectures address this problem through different mechanisms. Microsoft GraphRAG uses entity graphs and community summaries, LightRAG combines graph and vector retrieval, HippoRAG uses Personalized PageRank, GNN-RAG applies graph neural networks for answer candidate retrieval, and KAG integrates knowledge graphs with logical-form-guided reasoning. KG²RAG further demonstrates how graph relationships can be used to expand and organize semantically retrieved text chunks.

Although these approaches provide additional capabilities for structured retrieval and multi-hop evidence discovery, they also introduce challenges involving graph construction, extraction accuracy, scalability, retrieval noise, and reasoning-chain completeness.

The comparison suggests that graph-based retrieval should be evaluated not only by the relevance of retrieved nodes or subgraphs, but also by its ability to preserve the relationships required for answering complex questions. These observations provide the foundation for the next section, which examines multi-hop reasoning mechanisms and the different ways retrieval architectures support reasoning across multiple pieces of evidence.

VI. MULTI-HOP REASONING IN RETRIEVAL-AUGMENTED GENERATION

A. Introduction

Multi-hop reasoning is an important capability of Retrieval-Augmented Generation (RAG) systems because many information-seeking questions cannot be answered using a single passage or an isolated fact. Instead, answering such questions requires identifying and connecting multiple pieces of evidence, where the result of one reasoning step may determine what information is required in the next step. For example, a question asking for the university attended by the director of a particular film requires the system to first identify the director and then retrieve information about that person's education. The final answer therefore depends on a chain of related facts rather than a single retrieval result.

Traditional RAG architectures generally retrieve passages according to their relevance to the original query. Although this approach works well for direct factual questions, it may fail when the required evidence is distributed across multiple documents or when an intermediate entity is needed to discover the next relevant passage. Increasing the number of retrieved passages does not necessarily solve this problem because the retrieved context may contain relevant information without establishing the relationships required to answer the question.

Multi-hop RAG addresses this limitation by coordinating retrieval and reasoning across multiple steps. Depending on the architecture, this coordination may involve query decomposition, iterative retrieval, entity linking, graph traversal, path selection, or the combination of structured and unstructured knowledge. Approaches such as IRCoT interleave retrieval with reasoning, while Think-on-Graph (ToG) and Reasoning on Graphs (RoG) use knowledge graph structures to guide the discovery and verification of reasoning paths [22], [32], [33].

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

This section examines the principal approaches to multi-hop reasoning in RAG, their retrieval mechanisms, and the challenges that affect their reliability.

B. Understanding Multi-Hop Reasoning

A multi-hop question requires two or more dependent reasoning steps to reach an answer. Each step may involve retrieving a fact, identifying an entity, establishing a relationship, or using previously obtained information to determine the next retrieval operation.

A simplified multi-hop reasoning process can be represented as:

1) Question interpretation: Identify the entities, relationships, and information requested by the question.
2) Intermediate evidence retrieval: Retrieve the first set of passages or structured facts relevant to the question.
3) Intermediate reasoning: Extract the required information and identify the entity or relationship needed for the next step.
4) Subsequent retrieval: Use the intermediate result to retrieve additional supporting evidence.
5) Evidence integration: Combine the retrieved facts and verify that they form a logically connected chain.
6) Answer generation: Generate the final response using the collected evidence.

The important characteristic is the dependency between reasoning steps. In a genuine multi-hop question, the information obtained in an earlier step contributes to identifying or interpreting the evidence required in a later step.

For a question involving two reasoning steps, the process can be represented as:

Question → First evidence → Intermediate entity or fact → Second evidence → Final answer

This distinction is important because a system may retrieve several relevant passages without actually connecting them. Multi-hop reasoning therefore requires both sufficient evidence retrieval and correct evidence integration.

C. Taxonomy of Multi-Hop Retrieval and Reasoning Approaches

Multi-hop RAG systems can be classified according to how they identify intermediate information, retrieve additional evidence, and establish relationships between facts. The following categories provide an analytical classification of the approaches discussed in the literature. They are not mutually exclusive, as a single architecture may combine multiple strategies.

1) Entity-Centric Reasoning

Entity-centric reasoning uses entities mentioned in the question or discovered during retrieval as the primary points for navigating additional information. The system identifies relevant entities, retrieves their associated facts, and uses the extracted information to determine subsequent retrieval targets.

For example, consider a question asking for the country in which the university attended by a particular author's spouse is located. The system must identify the author, determine the spouse, retrieve the university attended by that person, and finally identify the university's country.

Entity-centric retrieval is particularly useful when the question contains identifiable people, organizations, locations, or other named entities. Knowledge graphs can support this process by representing entities as nodes and their relationships as edges. Entity linking and resolution are important because different names or references may correspond to the same real-world entity.

However, entity-centric reasoning depends heavily on the accuracy of entity extraction and linking. Incorrect entity identification can direct the retrieval process towards unrelated information and affect all subsequent reasoning steps.

2) Path-Based Reasoning

Path-based reasoning focuses on discovering a sequence of relationships connecting the starting entity to the required answer. Instead of retrieving facts independently, the system searches for a connected path that represents the reasoning process.

In a knowledge graph, a reasoning path can be represented as:

Entity A → Relationship 1 → Entity B → Relationship 2 → Entity C

The final entity is obtained by following the relationships from the starting entity through intermediate entities. This representation makes the dependency between reasoning steps explicit.

Think-on-Graph (ToG) uses an LLM agent to explore a knowledge graph iteratively. It employs a beam-search strategy to select promising entities and relationships, progressively identifying reasoning paths that may lead to the answer [32]. Reasoning on Graphs (RoG) follows a planning-retrieval-reasoning framework. It first generates relation-path plans grounded in the knowledge graph, retrieves matching reasoning paths, and then uses the retrieved evidence to produce the answer [33].

Path-based reasoning provides a structured representation of how an answer is derived. It can also improve traceability because the retrieved relationships can be inspected. However, its effectiveness depends on graph completeness, relationship accuracy, and the ability to identify the correct path among multiple possible alternatives.

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

3) Subgraph-Based Reasoning

Subgraph-based reasoning retrieves a connected portion of a knowledge graph containing entities and relationships relevant to the query. Rather than selecting only one path, the system collects a local subgraph that may contain multiple candidate paths and supporting facts.

The process generally involves identifying seed entities from the query, expanding the graph around those entities, and selecting a relevant subgraph using semantic relevance, graph connectivity, or other ranking criteria. The resulting structure is provided to the reasoning component for evidence integration.

This approach is useful when a question involves several related entities or when multiple paths may contribute to the answer. It can preserve relationships that would be lost if each fact were retrieved independently.

However, expanding a graph can introduce irrelevant nodes and relationships. A large subgraph may also exceed the available context window, making graph pruning and relevance-based selection important. Consequently, subgraph retrieval must balance evidence coverage against the amount of unnecessary information introduced.

4) Iterative and Retrieval-Reasoning Interleaved Approaches

Iterative approaches perform retrieval in multiple stages rather than completing retrieval in a single operation. After obtaining an initial set of evidence, the model reasons over the retrieved information and generates a follow-up query. The next retrieval step uses this query to identify additional evidence.

Interleaving Retrieval with Chain-of-Thought Reasoning (IRCoT) follows this principle by alternating between reasoning and retrieval. The reasoning process helps identify what information is needed next, while each retrieval step supplies additional evidence for subsequent reasoning [22].

A simplified workflow is:

Initial question → Retrieve evidence → Reason → Generate follow-up query → Retrieve additional evidence → Continue reasoning → Generate answer

This approach is particularly useful when the original question does not explicitly contain all the entities required for retrieval. The intermediate reasoning step helps reveal the missing information.

However, iterative retrieval introduces the possibility of error propagation. If the model generates an incorrect intermediate conclusion, subsequent queries may move away from the correct reasoning path. Repeated retrieval can also increase latency, token consumption, and the accumulation of irrelevant evidence. Therefore, effective iterative systems require mechanisms to evaluate intermediate evidence and determine when sufficient information has been collected.

5) Query Decomposition and Dependency-Aware Retrieval

Query decomposition transforms a complex question into smaller sub-questions that can be retrieved and answered individually. The sub-questions are then connected to produce the final answer.

For example, a complex question may be divided into:

- Which person directed the specified film?
- Which university did that person attend?
- In which country is that university located?

The answers to the sub-questions form a dependency chain. The answer to the first question provides information required by the second, and the answer to the second supports the final step.

A key distinction exists between ordinary query decomposition and dependency-aware decomposition. In ordinary decomposition, a system may generate several independent queries. In dependency-aware retrieval, the system explicitly represents how the answer to one sub-question determines the next retrieval operation.

Structured planning approaches attempt to address the limitations of uncontrolled iterative retrieval by establishing a reasoning plan before or during evidence collection. Recent work on structured planning for multi-hop question answering highlights the importance of maintaining retrieval direction and controlling the accumulation of irrelevant information [36].

Although query decomposition improves the organization of complex questions, an incorrect decomposition can omit essential relationships or introduce unnecessary sub-questions. The system must therefore verify that the retrieved answers collectively address the original question.

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

6) Community-Based and Global Reasoning

Community-based reasoning uses clusters of related entities and relationships to retrieve information at a broader level. Instead of focusing exclusively on individual facts or short paths, the system can retrieve summaries describing groups of connected entities and their relationships. Microsoft GraphRAG uses graph-based community detection and hierarchical community summaries to support local and global information retrieval [3]. Local search focuses on information associated with relevant entities and their connections, whereas global search uses community-level summaries to address broader questions involving multiple parts of a corpus. This approach is useful for questions that require synthesizing information across a large collection of documents. However, community-level retrieval should not automatically be considered explicit multi-hop reasoning. A community summary may contain relevant information without identifying a precise sequence of dependent relationships. Its effectiveness for multi-hop question answering therefore depends on whether the retrieved summaries preserve the evidence and relationships needed to support the answer.

7) Hybrid Vector-Graph Reasoning

Hybrid vector-graph reasoning combines semantic retrieval over unstructured documents with structured retrieval over knowledge graphs. Vector retrieval identifies semantically relevant passages, while graph retrieval helps discover relationships between entities and locate additional evidence connected to the initial results. This combination is useful because structured and unstructured representations provide complementary information. Knowledge graphs can support relationship discovery, while source documents provide contextual details and textual evidence that may not be fully represented in the graph. Think-on-Graph 2.0 (ToG-2) illustrates this approach by tightly integrating graph retrieval and document-context retrieval. The framework alternates between structured graph exploration and unstructured context retrieval, allowing each representation to guide the other [34].

Other architectures discussed in Section 5, including HippoRAG, LightRAG, KAG, and KG²RAG, also demonstrate different ways of combining semantic retrieval, entity relationships, graph structures, and textual context [27], [26], [29], [30].

Hybrid approaches can improve evidence discovery, but they introduce additional architectural complexity. The system must maintain consistency between the graph and document indexes, preserve the source of retrieved facts, and avoid treating an inferred graph relationship as verified textual evidence.

D. Comparison of Multi-Hop Reasoning Approaches

The approaches discussed above differ in their primary retrieval units, reasoning mechanisms, and limitations. Table 6 summarizes their main characteristics.

Approach	Primary retrieval unit	Reasoning mechanism	Main advantage	Major limitation
Entity-centric	Entities and associated facts	Entity identification and relationship linking	Effective for entity-oriented questions	Sensitive to entity-linking errors
Path-based	Connected relationship paths	Sequential traversal of relationships	Explicit reasoning chains and traceability	Depends on graph completeness
Subgraph-based	Connected graph regions	Reasoning over multiple related facts	Preserves broader relational context	May introduce irrelevant nodes
Iterative/interleaved	Passages retrieved at successive steps	Alternating retrieval and reasoning	Supports dynamically discovered evidence	Error propagation and increased latency
Query decomposition	Sub-questions	Sequential or dependency-aware question solving	Organizes complex questions into manageable steps	Incorrect decomposition may omit dependencies
Community-based	Community summaries and related entities	Broad synthesis across connected information	Supports corpus-level information discovery	May not preserve explicit reasoning paths
Hybrid vector-graph	Text passages and graph structures	Alternating or combined semantic and relational retrieval	Combines textual context with structured relationships	Index synchronization and architectural complexity

Table 6. Comparison of multi-hop retrieval and reasoning approaches.

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

The comparison indicates that no single approach addresses every multi-hop retrieval requirement. Entity-centric and path-based approaches provide explicit mechanisms for identifying and connecting relationships. Iterative retrieval and query decomposition are useful when the required evidence cannot be identified directly from the original query. Subgraph and community-based methods provide broader contextual information, while hybrid approaches combine the strengths of structured and unstructured retrieval. The appropriate strategy depends on the nature of the question, the availability and quality of structured knowledge, and the amount of evidence required to establish a reliable answer.

E. Benchmarks for Evaluating Multi-Hop Reasoning

Evaluating multi-hop RAG systems requires datasets that test not only answer generation but also the retrieval and integration of supporting evidence. Several benchmarks have been developed to assess different aspects of multi-hop question answering.

1) HotpotQA: HotpotQA is a widely used benchmark containing questions that require reasoning over multiple Wikipedia passages. It includes supporting-fact annotations, allowing researchers to examine whether a system retrieves the evidence needed to support its answer [4].
2) 2WikiMultiHopQA: This dataset focuses on multi-hop question answering using structured and unstructured information from Wikipedia. It includes supporting evidence and reasoning paths, making it useful for evaluating the ability to connect facts across documents [5].
3) MultiHop-RAG: MultiHop-RAG is designed specifically to evaluate retrieval-augmented generation systems on multi-hop queries. It includes questions requiring information to be gathered from multiple documents and provides a setting for studying the interaction between retrieval and answer generation [6].
4) MuSiQue: MuSiQue was developed to address the possibility that some existing multi-hop questions can be answered through shortcuts without performing the intended connected reasoning. It contains approximately 25,000 questions involving two to four reasoning hops, constructed by composing connected single-hop questions. Its design makes it particularly useful for assessing whether a system genuinely follows the dependency between reasoning steps [35].

These datasets provide complementary evaluation settings. HotpotQA and 2WikiMultiHopQA support the study of evidence retrieval and cross-document reasoning, MultiHop-RAG focuses on RAG-oriented multi-document questions, and MuSiQue places greater emphasis on connected reasoning and reducing shortcut-based solutions.

F. Evaluation Dimensions for Multi-Hop RAG Systems

A reliable evaluation framework should distinguish retrieval quality from reasoning quality and final answer generation. A system may retrieve relevant evidence but fail to connect it correctly, or it may generate a plausible answer without retrieving all the necessary supporting facts.

The following dimensions are important when evaluating multi-hop RAG architectures:

1) Retrieval Recall: Measures whether the system retrieves the relevant supporting passages or facts required to answer the question.
2) Supporting Evidence Precision and Recall: Examines whether the retrieved evidence contains the required supporting facts while limiting irrelevant information.
3) Answer Exact Match and F1: Measures the correctness of the generated answer by comparing it with the reference answer. These metrics are commonly used in multi-hop question-answering benchmarks.
4) Reasoning-Path Correctness: Examines whether the relationships or intermediate steps connecting the evidence to the final answer are valid. For graph-based systems, this may involve checking the retrieved path against annotated or verified relationships.
5) Evidence Completeness: Evaluates whether all essential facts needed to derive the answer are present in the retrieved context. This is particularly important when the evidence is distributed across several documents.
6) Faithfulness and Groundedness: Determines whether the generated answer is supported by the retrieved evidence rather than unsupported model knowledge or fabricated relationships.
7) Retrieval and Generation Efficiency: Considers retrieval latency, total response time, token consumption, and computational cost. Multi-step systems may improve evidence coverage but introduce additional retrieval and reasoning overhead.

These dimensions should be reported separately wherever possible. In particular, answer accuracy alone cannot establish whether a system has retrieved complete evidence or followed a valid reasoning chain. Similarly, high retrieval recall does not guarantee that the generator will correctly interpret the retrieved facts.

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

G. Challenges in Multi-Hop Reasoning

Despite improvements in retrieval and reasoning architectures, several challenges continue to affect multi-hop RAG systems.

1) Missing Intermediate Evidence: A system may retrieve information relevant to the original question but fail to identify the intermediate entity or fact required for the next reasoning step. Missing one essential piece of evidence can invalidate the entire reasoning chain.
2) Entity Linking and Relationship Errors: Incorrect entity resolution or relation extraction can create misleading connections between facts. These errors are particularly problematic in knowledge graph-based systems because subsequent traversal may follow relationships that do not represent the intended meaning.
3) Reasoning Error Propagation: Iterative approaches depend on intermediate outputs to generate subsequent queries. An incorrect intermediate conclusion can direct later retrieval operations towards irrelevant evidence, resulting in a progressively incorrect reasoning process.
4) Graph Expansion and Retrieval Noise: Expanding a graph around relevant entities may introduce unrelated nodes and relationships. Similarly, retrieving large numbers of passages can increase the amount of irrelevant context presented to the generator.
5) Incomplete Evidence Integration: Even when all required facts are retrieved, the model may fail to establish the correct relationship between them. This demonstrates that retrieval completeness and reasoning correctness are separate requirements.
6) Context Window and Computational Constraints: Multi-hop systems may require several retrieval iterations, intermediate reasoning steps, and supporting passages. The resulting context can become large, increasing token consumption, latency, and computational cost.
7) Evidence Provenance and Verification: A generated answer should be traceable to the documents or graph relationships that support it. Without source-level provenance, it becomes difficult to determine whether a conclusion is based on retrieved evidence or an unsupported inference made by the language model.
8) Stopping Criteria: Iterative systems must determine when enough evidence has been collected. Stopping too early may leave the reasoning chain incomplete, while continuing unnecessarily can introduce noise and increase computational cost.

H. Summary

Multi-hop reasoning extends conventional RAG by requiring the retrieval and integration of multiple dependent pieces of evidence. The approaches discussed in this section demonstrate that multi-hop question answering can be supported through entity-centric retrieval, relationship-path exploration, subgraph retrieval, iterative reasoning, query decomposition, community-level retrieval, and hybrid vector-graph architectures.

An important observation is that multi-hop reasoning is not restricted to knowledge graph traversal. Iterative retrieval over unstructured documents can also support multi-step reasoning when each retrieval operation depends on the information obtained in previous steps. Similarly, the use of a knowledge graph does not automatically guarantee correct multi-hop reasoning. The system must identify valid relationships, retrieve complete supporting evidence, and correctly integrate the retrieved facts.

Therefore, the effectiveness of multi-hop RAG should be evaluated through a combination of retrieval quality, evidence completeness, reasoning correctness, answer accuracy, faithfulness, and computational efficiency. These requirements provide the foundation for the comparative evaluation framework discussed in the following section.

VII. COMPARATIVE ANALYSIS AND DISCUSSION

A. Comparison of RAG Architectures

The techniques discussed in the previous sections represent different approaches to retrieving and utilizing external knowledge. Traditional retrieval methods primarily focus on identifying relevant passages, while advanced RAG introduces query transformation, reranking, and iterative retrieval. Knowledge graph-based approaches further extend retrieval by representing relationships between entities and supporting structured information discovery.

Table 7 presents a qualitative comparison of these architectures based on their retrieval mechanisms and suitability for multi-hop question answering.

Architecture	Retrieval mechanism	Multi-hop support	Key strength	Main limitation
Sparse RAG	Keyword-based retrieval	Limited	Efficient exact-term matching	Weak semantic understanding
Dense RAG	Embedding similarity	Limited to	Semantic retrieval	May miss relationships

		moderate		between facts
Hybrid RAG	Sparse and dense retrieval	Moderate	Combines lexical and semantic relevance	Requires retrieval fusion
Advanced RAG	Query transformation, reranking and iterative retrieval	Moderate to high	Improves retrieval quality and evidence discovery	Additional latency and complexity
GraphRAG	Entity relationships and community summaries	Depends on retrieval strategy	Structured and corpus-level information retrieval	Graph construction overhead
Hybrid Graph RAG	Vector and graph retrieval	High potential	Combines textual evidence and structured relationships	Requires coordination between retrieval components

Table 7. Qualitative comparison of RAG architectures.

The comparison indicates that retrieval quality alone is insufficient for complex question answering. Dense and hybrid retrieval can identify semantically relevant passages, but they may not explicitly establish the relationships between facts distributed across documents. Advanced RAG addresses some of these limitations through iterative retrieval and query decomposition.

Knowledge graph-based approaches provide an additional representation of relationships, allowing systems to explore connected entities and evidence. However, their effectiveness depends on graph quality and the reasoning mechanism employed. Consequently, the choice of architecture should depend on the complexity of the query rather than the assumption that one retrieval strategy is universally superior.

B. Comparative Analysis of Multi-Hop Approaches

Multi-hop reasoning architectures differ in how they discover and connect intermediate evidence. IRCoT uses iterative retrieval guided by reasoning, whereas ToG explores knowledge graph relationships to identify relevant paths. RoG introduces structured reasoning plans, and HippoRAG uses graph-based knowledge organization and Personalized PageRank to retrieve relevant information [22], [27], [32], [33].

LightRAG and KAG combine structured knowledge with textual retrieval, while KG²RAG uses knowledge graph relationships to organize and expand retrieved chunks [26], [29], [30]. These systems demonstrate that multi-hop retrieval can be implemented through different combinations of graph traversal, semantic retrieval, and reasoning.

An important distinction is that these architectures are not directly comparable through their design descriptions alone. They use different datasets, underlying language models, retrieval configurations, and evaluation settings. Therefore, performance claims should be interpreted using the experimental conditions reported in their respective studies.

C. Key Observations

The comparative analysis leads to the following observations:

1) Retrieval and reasoning are separate requirements: Retrieving relevant passages does not guarantee that a system can identify the correct relationships between them.

2) Graph structures provide relational information: Knowledge graphs make entity relationships explicit and can support path-based retrieval, but their effectiveness depends on accurate graph construction and traversal.

3) Iterative retrieval supports dynamic evidence discovery: Methods such as IRCoT allow subsequent retrieval operations to depend on intermediate reasoning results. However, errors in early steps can affect later retrieval.

4) Hybrid architectures offer complementary capabilities: Combining vector retrieval with graph-based retrieval can provide both semantic relevance and relational context.

5) Evaluation requires multiple dimensions: Answer accuracy should be considered alongside evidence retrieval, reasoning correctness, faithfulness, latency, and computational cost.

Overall, the literature suggests a progression from retrieving relevant information towards retrieving and organizing evidence according to the dependencies within a question. Nevertheless, designing systems that consistently retrieve complete evidence and establish correct reasoning chains remains an important research challenge.

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

VIII. RESEARCH CHALLENGES AND FUTURE DIRECTIONS

Although RAG architectures have evolved significantly, several challenges remain in supporting reliable multi-hop reasoning. Addressing these limitations is important for developing systems that can retrieve complete evidence, establish valid relationships, and generate accurate responses.

1) Retrieval Completeness: Multi-hop questions often require information distributed across multiple documents. Existing retrieval methods may identify relevant passages but miss intermediate evidence needed to connect the facts. Future systems should focus on improving evidence coverage and identifying missing information during retrieval.

2) Knowledge Graph Quality and Scalability: Graph-based RAG depends on accurate entity extraction, relationship identification, and entity resolution. Errors during graph construction can affect subsequent retrieval and reasoning. Efficient graph construction, incremental updates, and reliable entity resolution remain important research directions.

3) Reasoning Reliability: Iterative retrieval and graph traversal can introduce errors when intermediate conclusions are incorrect. Future architectures should incorporate evidence verification and mechanisms to validate reasoning steps before generating the final answer.

4) Retrieval Efficiency: Multi-hop systems may require several retrieval operations, graph expansions, and LLM calls. These operations increase response latency and computational cost. Research should investigate efficient retrieval planning, selective graph expansion, and context compression without sacrificing essential evidence.

5) Evaluation and Explainability: Existing evaluation often emphasizes final answer accuracy, which does not fully capture the quality of intermediate reasoning. Future benchmarks should evaluate supporting evidence, reasoning-path correctness, evidence completeness, and answer faithfulness together. Providing traceable sources for each reasoning step is also important for explainable RAG systems.

Overall, future research should move towards RAG architectures that combine efficient retrieval, reliable multi-step reasoning, evidence verification, and source-level traceability.

IX. CONCLUSION

Retrieval-Augmented Generation has evolved from conventional sparse and dense retrieval towards advanced architectures that incorporate query transformation, iterative retrieval, and structured knowledge representations. While traditional approaches are effective for retrieving relevant passages, complex questions require systems to identify and connect multiple pieces of evidence.

This survey examined the development of RAG techniques with particular emphasis on their ability to support multi-hop reasoning. It discussed sparse, dense, and hybrid retrieval, advanced RAG strategies, knowledge graph-based architectures, and different approaches to multi-hop evidence discovery. The analysis highlights that entity-based retrieval, path exploration, subgraph retrieval, iterative reasoning, and hybrid vector-graph architectures provide complementary mechanisms for addressing complex information needs. An important finding is that the presence of a knowledge graph or multiple retrieved passages does not independently guarantee correct multi-hop reasoning. Reliable performance requires complete evidence retrieval, accurate relationship identification, valid reasoning chains, and grounded answer generation. Therefore, future RAG systems should be evaluated not only by their final answer accuracy but also by their retrieval quality, reasoning reliability, faithfulness, and computational efficiency.

The continued integration of semantic retrieval, structured knowledge, and evidence-guided reasoning offers a promising direction for building more reliable RAG systems capable of addressing complex, multi-document questions.

REFERENCES

[1] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in Advances in Neural Information Processing Systems, vol. 33, pp. 9459–9474, 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html

[2] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W. T. Yih, "Dense Passage Retrieval for Open-Domain Question Answering," in Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP), pp. 6769–6781, 2020, doi: 10.18653/v1/2020.emnlp-main.550. [Online]. Available: https://aclanthology.org/2020.emnlp-main.550/

[3] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, D. Metropolitansky, R. Osazuwa Ness, and J. Larson, "From Local to Global: A Graph RAG Approach to Query-Focused Summarization," arXiv preprint arXiv:2404.16130, 2024. [Online]. Available: https://arxiv.org/abs/2404.16130

[4] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. Cohen, R. Salakhutdinov, and C. D. Manning, "HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering," in Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP), pp. 2369–2380, 2018, doi: 10.18653/v1/D18-1259. [Online]. Available: https://aclanthology.org/D18-1259/

[5] X. Ho, A.-K. Duong Nguyen, S. Sugawara, and A. Aizawa, "Constructing A Multi-hop QA Dataset for Comprehensive Evaluation of Reasoning Steps," in Proc. 28th Int. Conf. Computational Linguistics (COLING), pp. 6609–6625, 2020, doi: 10.18653/v1/2020.coling-main.580. [Online]. Available: https://aclanthology.org/2020.coling-main.580/
[6] Y. Tang and Y. Yang, "MultiHop-RAG: Benchmarking Retrieval-Augmented Generation for Multi-Hop Queries," in Proc. Conf. Language Modeling (COLM), 2024. [Online]. Available: https://proceedings.colm.cc/paper_files/paper/2024/hash/906419cd502575b617cc489a1a696a67-Abstract-Conference.html
[7] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang, "Retrieval Augmented Language Model Pre-Training," Proc. 37th Int. Conf. Machine Learning (ICML), vol. 119, pp. 3929–3938, 2020. proceedings.mlr.press
[8] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, "Retrieval-Augmented Generation for Large Language Models: A Survey," arXiv:2312.10997, 2023. arxiv.org
[9] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection," ICLR, 2024. proceedings.iclr.cc
[10] S.-Q. Yan, J.-C. Gu, Y. Zhu, and Z.-H. Ling, "Corrective Retrieval Augmented Generation," arXiv:2401.15884, 2024. arxiv.org
[11] P. Sarthi, S. Abdullah, A. Tuli, S. Khanna, A. Goldie, and C. D. Manning, "RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval," ICLR, 2024. proceedings.iclr.cc
[12] G. Salton and C. Buckley, "Term-Weighting Approaches in Automatic Text Retrieval," Information Processing & Management, vol. 24, no. 5, pp. 513–523, 1988, doi: 10.1016/0306-4573(88)90021-0. doi.org
[13] S. Robertson and H. Zaragoza, "The Probabilistic Relevance Framework: BM25 and Beyond," Foundations and Trends in Information Retrieval, vol. 3, no. 4, pp. 333–389, 2009, doi: 10.1561/1500000019. doi.org
[14] T. Formal, B. Piwowarski, and S. Clinchant, "SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking," in Proc. 44th Int. ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR), pp. 2288–2292, 2021, doi: 10.1145/3404835.3463098. doi.org
[15] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP-IJCNLP), pp. 3982–3992, 2019, doi: 10.18653/v1/D19-1410. aclanthology.org
[16] Y. A. Malkov and D. A. Yashunin, "Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 42, no. 4, pp. 824–836, 2020, doi: 10.1109/TPAMI.2018.2889473. ieeexplore.ieee.org
[17] J. Johnson, M. Douze, and H. Jégou, "Billion-Scale Similarity Search with GPUs," arXiv:1702.08734, 2017. arxiv.org
[18] G. V. Cormack, C. L. A. Clarke, and S. Büttcher, "Reciprocal Rank Fusion Outperforms Condorcet and Individual Rank Learning Methods," in Proc. 32nd Int. ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR), pp. 758–759, 2009, doi: 10.1145/1571941.1572114. doi.org
[19] L. Gao, X. Ma, J. Lin, and J. Callan, "Precise Zero-Shot Dense Retrieval without Relevance Labels," in Proc. 61st Annu. Meeting Assoc. Comput. Linguistics (ACL), vol. 1, pp. 1762–1777, 2023, doi: 10.18653/v1/2023.acl-long.99. aclanthology.org
[20] L. Wang, N. Yang, and F. Wei, "Query2doc: Query Expansion with Large Language Models," in Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP), pp. 9414–9423, 2023, doi: 10.18653/v1/2023.emnlp-main.585. aclanthology.org
[21] O. Khattab and M. Zaharia, "ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT," in Proc. 43rd Int. ACM SIGIR Conf. Research and Development in Information Retrieval, 2020, doi: 10.1145/3397271.3401075. dl.acm.org
[22] H. Trivedi, N. Balasubramanian, T. Khot, and A. Sabharwal, "Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions," in Proc. 61st Annu. Meeting Assoc. Comput. Linguistics (ACL), vol. 1, pp. 10014–10037, 2023, doi: 10.18653/v1/2023.acl-long.557. aclanthology.org
[23] Anthropic, "Introducing Contextual Retrieval," Sep. 19, 2024. anthropic.com
[24] B. Peng, Y. Zhu, Y. Liu, X. Bo, H. Shi, C. Hong, Y. Zhang, and S. Tang, "Graph Retrieval-Augmented Generation: A Survey," arXiv:2408.08921, 2024. arxiv.org
[25] Microsoft, "GraphRAG Indexing and Query Methods," Microsoft GraphRAG Documentation. microsoft.github.io
[26] Z. Guo, L. Xia, Y. Yu, T. Ao, and C. Huang, "LightRAG: Simple and Fast Retrieval-Augmented Generation," arXiv:2410.05779, 2024. arxiv.org
[27] B. J. Gutiérrez, Y. Shu, Y. Gu, M. Yasunaga, and Y. Su, "HippoRAG: Neurobiologically Inspired Long-Term Memory for Large Language Models," in Advances in Neural Information Processing Systems, vol. 37, 2024. proceedings.neurips.cc
[28] C. Mavromatis and G. Karypis, "GNN-RAG: Graph Neural Retrieval for Efficient Large Language Model Reasoning on Knowledge Graphs," in Findings of the Association for Computational Linguistics: ACL 2025, pp. 16682–16699, 2025, doi: 10.18653/v1/2025.findings-acl.856. aclanthology.org
[29] L. Liang et al., "KAG: Boosting LLMs in Professional Domains via Knowledge Augmented Generation," arXiv:2409.13731, 2024. arxiv.org
[30] X. Zhu, Y. Xie, Y. Liu, Y. Li, and W. Hu, "Knowledge Graph-Guided Retrieval Augmented Generation," in Proc. 2025 Conf. North American Chapter of the Association for Computational Linguistics: Human Language Technologies, vol. 1, pp. 8912–8924, 2025, doi: 10.18653/v1/2025.naacl-long.449. aclanthology.org

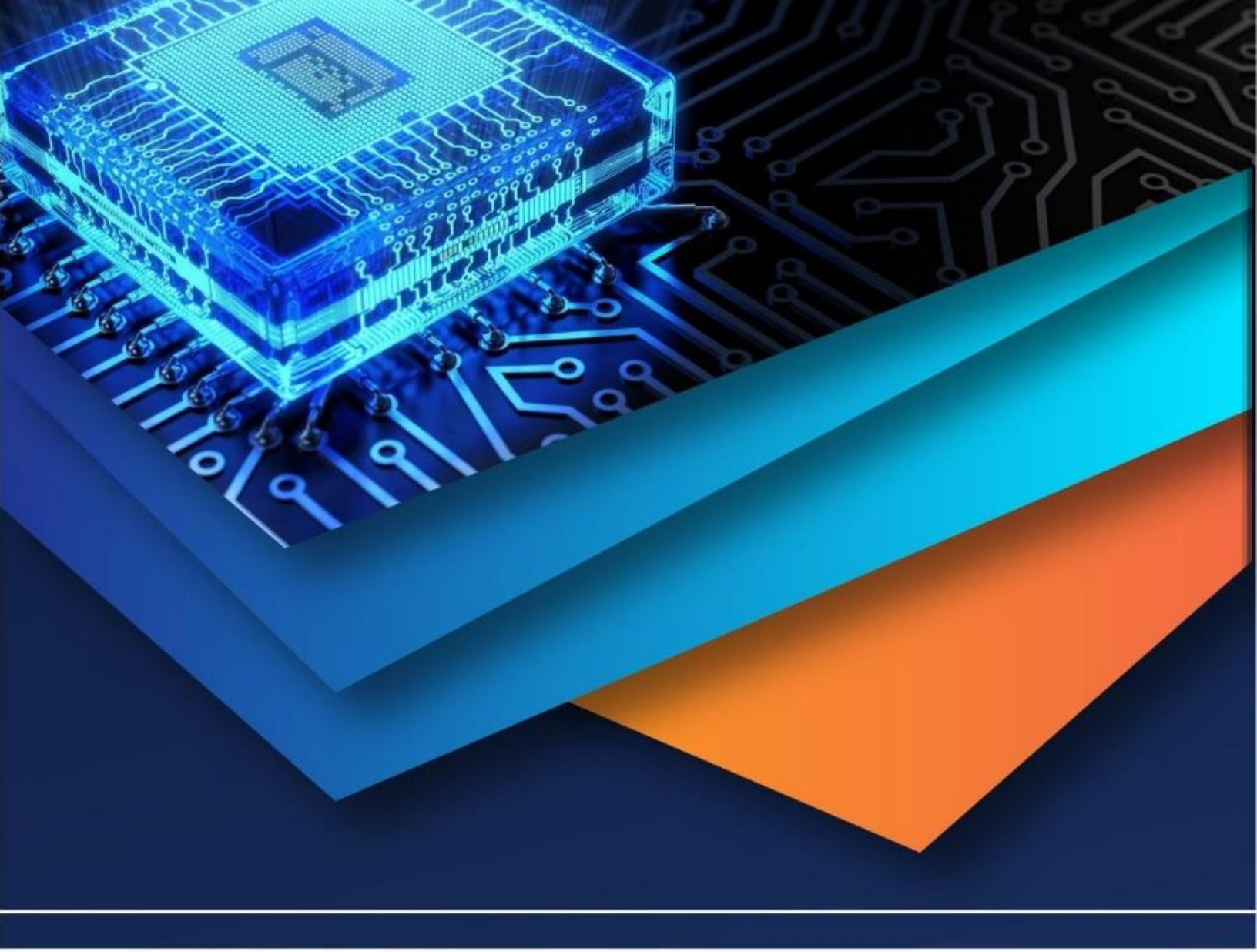
Additional relevant paper for our literature review:

[31] Y. Hu, Z. Lei, Z. Zhang, B. Pan, C. Ling, and L. Zhao, "GRAG: Graph Retrieval-Augmented Generation," in Findings of the Association for Computational Linguistics: NAACL 2025, pp. 4145–4157, 2025, doi: 10.18653/v1/2025.findings-naacl.232.
[32] J. Sun, C. Xu, L. Tang, S. Wang, C. Lin, Y. Gong, L. M. Ni, H. Y. Shum, and J. Guo, "Think-on-Graph: Deep and Responsible Reasoning of Large Language Model on Knowledge Graph," ICLR, 2024. proceedings.iclr.cc
[33] L. Luo, Y. F. Li, G. Haffari, and S. Pan, "Reasoning on Graphs: Faithful and Interpretable Large Language Model Reasoning," ICLR, 2024. proceedings.iclr.cc
[34] S. Ma et al., "Think-on-Graph 2.0: Deep and Faithful Large Language Model Reasoning with Knowledge-guided Retrieval Augmented Generation," ICLR, 2025. proceedings.iclr.cc
[35] H. Trivedi, N. Balasubramanian, T. Khot, and A. Sabharwal, "MuSiQue: Multihop Questions via Single-hop Question Composition," Transactions of the Association for Computational Linguistics, 10, 539–554, 2022. aclanthology.org
[36] Y. Huang, L. Yang, X. Yang, and X. Xu, "Retrieval-Augmented Generation for Multi-Hop Question Answering Based on Structured Planning," ACM Transactions on Knowledge Discovery from Data, 20(3), Article 44, 2026. doi.org

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

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue XI Nov 2026- Available at www.ijraset.com

©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)