Asian Journal of Engineering and Applied Technology
ISSN: 2249-068X (P); 3049-1959 (E)
Vol.14 No.2, 2025, pp.29-34

www.crijournals.org
DOI: https://doi.org/10.70112/ajeat-2025.14.2.4327

Improving RAG Accuracy Through Multi-Retrieval Integration and Rank Base Retrieved Chunk Selection

Suraj Singh Patwal[1*], Devashish Chauhan[2] and Shyam Ji[3]
[1*]Department of Mechanical Engineering, IIT Patna, Bihar, India
[2]Department of Computer Science and Engineering, Graphic Era, Dehradun, India
[3]Department of Electronics and Communication, NIT Hamirpur, India
E-mail: devashishch018@gmail.com, shyamnit432@gmail.com
*Corresponding Author: patwalsuraj11@gmail.com
(Received 1 October 2025; Revised 20 October 2025; Accepted 15 November 2025; Available online 2 December 2025)

***Abstract* - In a standard RAG pipeline, source documents are split into smaller chunks, and embedding models generate vector representations for these chunks. The embeddings are stored in a vector database, which retrieves relevant chunks using vector similarity or keyword-based search. The retrieved chunks are then combined with the user query and passed to an LLM to generate the final response. Although effective, traditional RAG systems depend heavily on the choice of embedding model, retrieval method, and the number of retrieved chunks, all of which significantly impact accuracy and hallucination levels. Results show that the proposed RAG system significantly outperforms individual retrieval systems. It achieves a correctness score of 79.75% and a similarity score of 78.7%, surpassing all baseline RAG pipelines. Furthermore, experiments varying the number of retrieved chunks per retriever (from 1 to 10) reveal an interesting trend: performance peaks at several even-numbered retrieval counts, indicating local maxima in correctness and similarity when using even numbers of retrieved documents before applying Reciprocal Rank Fusion (RRF). Overall, this study demonstrates that combining multiple retrieval mechanisms with RRF yields more accurate, contextually aligned, and consistent outputs compared to traditional single-retriever RAG implementations. The proposed framework enhances RAG reliability without fine-tuning and provides empirically validated insights into the impact of retrieval volume on performance. The work repository is publicly maintained at: https://github.com/Surajxyz/RAG_PAPER**
***Keywords:* Retrieval-Augmented Generation (RAG), Vector Embeddings, Reciprocal Rank Fusion (RRF), Large Language Models (LLMs), Information Retrieval**

I. INTRODUCTION

Large Language Models (LLMs) are trained on extremely large corpora of text, allowing them to learn patterns, linguistic structures, factual knowledge, and reasoning capabilities. All of this acquired information is stored in the form of numerical parameters, commonly referred to as model weights. Once training is completed, the model's knowledge becomes static; it is limited to the data available up to the point at which the training dataset was collected and processed. As a result, an LLM cannot inherently learn or update itself after deployment unless it undergoes another training or fine-tuning cycle. This creates a natural restriction: the model is unaware of events, facts, and domain-specific changes that occurred after its pretraining cutoff date. When users ask questions whose answers depend on information outside the model's training knowledge, the model attempts to infer or "guess" the answer based on its learned patterns. This behavior often results in hallucinations, where the model confidently generates incorrect or fabricated responses. To mitigate hallucination, there are two primary approaches. The first is fine-tuning, where the model is trained further on new domain-specific datasets. Although fine-tuning can update the model's knowledge and improve performance on specialized tasks, it has several drawbacks. Fine-tuning requires significant computational resources, making it expensive for large-scale or frequent updates. It also carries the risk of catastrophic forgetting, in which the model loses some of its original general knowledge while adapting to new data. Additionally, organizations may not always have access to the necessary GPU resources or permissions to perform fine-tuning on proprietary models. Because of these limitations, fine-tuning is often impractical, especially when the goal is simply to provide up-to-date or domain-specific factual information.

The second and far more practical approach is Retrieval-Augmented Generation (RAG). RAG avoids modifying the model's weights and instead provides the model with the information it needs at query time. This makes RAG especially effective for scenarios in which the model must answer domain-specific questions or respond to information that has changed over time—before or after the model's training cutoff. Rather than relying solely on the internal knowledge of the LLM, RAG allows the system to reference an external knowledge source, enabling the model to produce more accurate, grounded, and up-to-date answers. The RAG pipeline typically begins with data ingestion, where raw documents such as PDFs, webpages, manuals, or reports are processed into smaller, manageable text segments called chunks. Chunking is important because most embedding models and vector databases operate more efficiently on smaller, coherent pieces of text. Once chunks are created, each chunk is converted into an embedding vector using an embedding model, which may be an open-source model such as Sentence Transformers or a closed-source model such as OpenAI's text-embedding series. The

chunk text and its corresponding embedding vector are then stored in a vector store, which supports fast similarity search. After ingestion, the second major RAG stage is retrieval and generation. When a user submits a query, the system generates an embedding for the query and performs a similarity search within the vector store. The goal is to retrieve the chunks that are most relevant to the user's question. Retrieval can be based on pure vector similarity or supplemented with keyword-based methods such as BM25. The selected chunks form the context, which is combined with the user's query and fed to the LLM. The LLM now has access to external factual information retrieved specifically for the query, enabling it to generate a more accurate and grounded response. This augmented approach significantly reduces hallucination and enhances the model's reliability for real-world tasks.

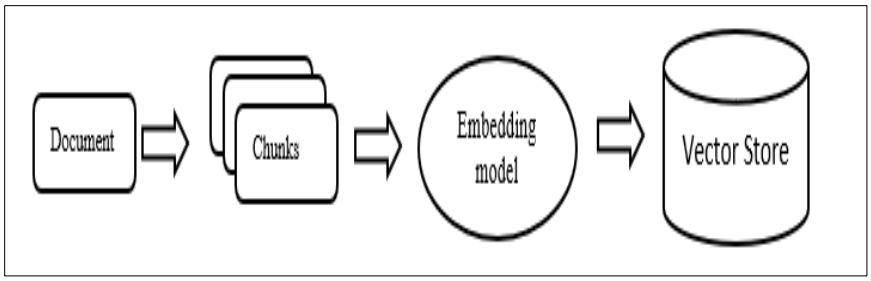

Fig.1 Chunk and Embedding Ingestion Process

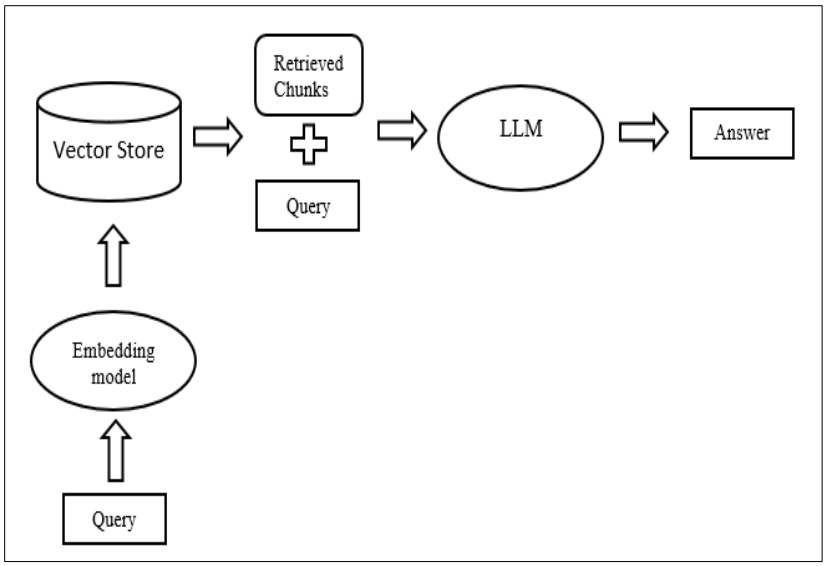

Fig.2 Retrieval of Relevant Context and Generation of an Answer

II. RELATED WORK

Large Language Models (LLMs) cannot effectively capture low-popularity information or domain-specific knowledge that lies outside their pretraining datasets; therefore, fine-tuning or Retrieval-Augmented Generation (RAG) is typically employed. In fine-tuning, a major challenge encountered is catastrophic forgetting, in which LLMs begin to lose previously learned knowledge. As a result, fine-tuning alone often performs worse compared to RAG or a combined fine-tuning and RAG approach, as reported by Soudani *et al.,* [1]. In their study, English and Chinese datasets were used with different LLMs to evaluate RAG across four capabilities: noise robustness, negative rejection, information integration, and counterfactual robustness. For long-answer generation, Chen *et al.,* [4] and Jiang *et al.,* [5] proposed Forward-Looking Active Retrieval-Augmented Generation. RAG has been shown to support a wide range of tasks, including dialogue response generation, machine translation, language modeling, summarization, paraphrase generation, and data-to-text generation, as surveyed by Li *et al.,* [6]. However, lower RAG accuracy is often attributed to suboptimal retrieval performance, which can result in irrelevant chunk selection. To address this issue, reranking techniques such as Reciprocal Rank Fusion (RRF) are employed. This approach, which is also adopted in our work, enhances retrieval effectiveness by generating similar queries and increasing retrieval depth. Rackauckas *et al.,* [7] reported that the retrieval time using RRF is approximately 1.77 times higher than that of traditional RAG.

In addition, the FILCO technique filters retrieved contexts using a lexical approach and trains a filtering model to select the most relevant chunks for generation, as proposed by Wang *et al.,* [12]. Retrieval evaluation has also been explored by Salemi *et al.,* [2] through the introduction of eRAG. Their findings indicate that retrieval accuracy decreases when selected chunks originate from the middle

of a document and improves when chunks are selected from the beginning or end. This phenomenon, referred to as the "loss in the middle," was identified by Liu *et al.,* [8]. Beyond vector databases, some studies employ graph-based databases in which data are represented as nodes and edges, with nodes storing content and edges representing relationships. When graphs are used as the retrieval backend, the approach is referred to as GraphRAG or LightRAG, as described by Peng *et al.,* [9], Han *et al.,* [10], and Guo *et al.,* [14]. For large document collections, LongRAG has been proposed, leveraging long-context window generation models to achieve significantly improved performance, as demonstrated by Jiang *et al.,* [13]. For RAG evaluation, the RAGAS framework has been introduced; however, in some cases, it produces NaN values when the LLM fails to generate scores. Consequently, the OpenEvals library is used instead of RAGAS in this work, following the observations of Es *et al.,* [21].

III. METHODOLOGY

In this paper, we construct a dataset consisting of 20 questions and corresponding ground-truth answers, randomly extracted from a 603-page PDF document. This PDF is ingested into the RAG pipeline, where it is segmented into chunks of 500 tokens each. Each chunk is represented as a document in which the *page_content* field contains the chunk text and the *metadata* field includes the *chunk_id* and source information (e.g., Document (page_ content="chunk content", metadata = { " source": "pdf", "chunk_id": 0})). Each document is stored in a FAISS-based vector store using different embedding models, namely *text-embedding-3-small*, *text-embedding-3-large*, and *all-mpnet-base-v2*. Each embedding model is used to construct a traditional RAG pipeline. In addition, a fourth RAG pipeline is implemented using BM25 retrieval, which performs keyword-based search without a vector store.

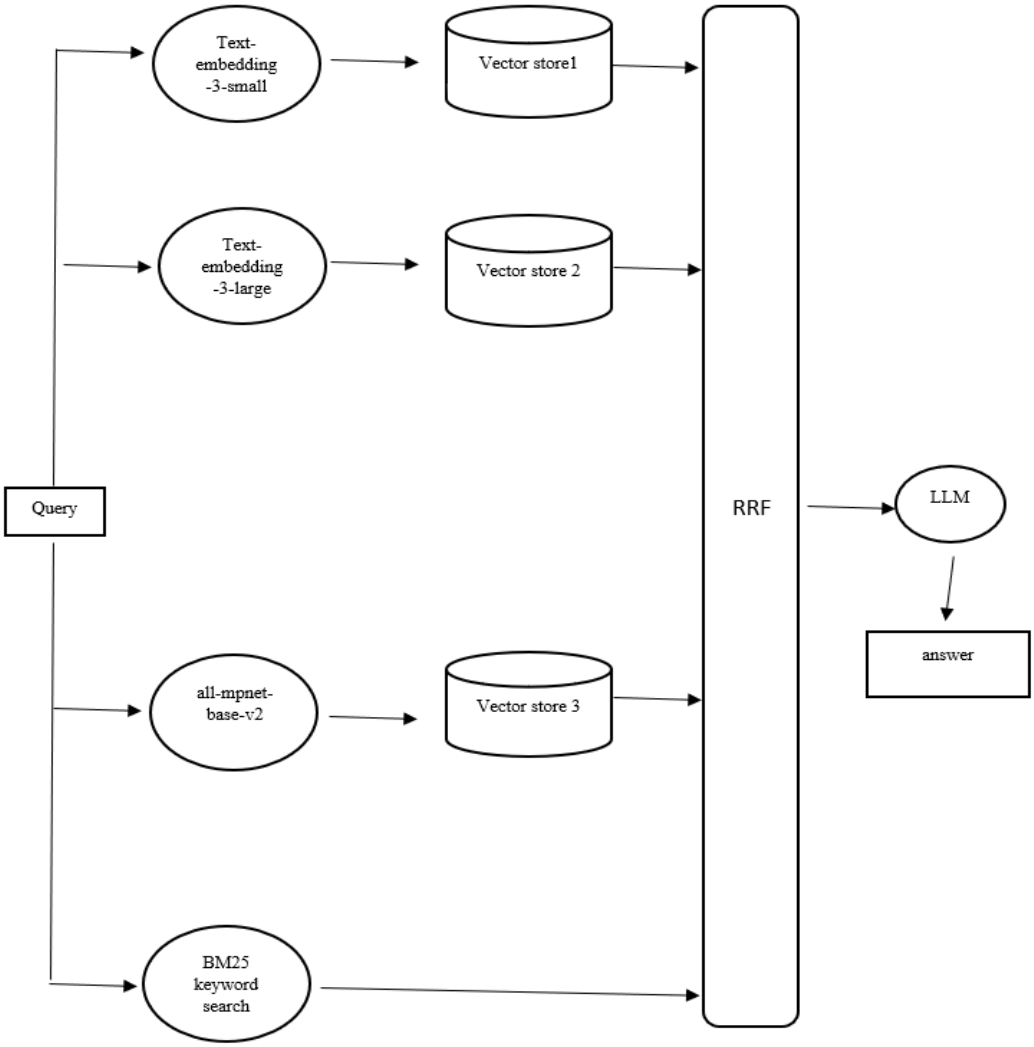

Fig.3 Our RAG Technique with Four Retrievals and Reciprocal Rank Fusion

Across all four traditional RAG pipelines, *gpt-3.5-turbo* is used as the common generation model. In this work, we further introduce a RAG framework that integrates all the retrieval methods described above. Each retriever generates four candidate chunks, after which Reciprocal Rank Fusion (RRF) is applied to assign a score to each retrieved chunk

based on its rank across the individual retrieval results. The chunk with the highest aggregated score is then selected as context for answer generation using *gpt-3.5-turbo*. The dataset is evaluated using the five RAG pipelines described above, resulting in five tables containing the question, ground truth, retrieved context, and generated answer. These outputs are subsequently evaluated using the OpenEvals library in terms of correctness, conciseness, hallucination, and similarity, with *gpt-4o-mini* serving as the evaluation model. In the RRF process, a score is assigned to each unique document or context appearing in the four retrieval result sets. The final score for a document is computed as the sum of the reciprocals of its ranks across the retrieval sets in which it appears. The document or chunk with the highest total score is selected for answer generation, as defined by the RRF scoring formula.

$$\text{SCORE (Retrieved Doc)} = \sum_{1}^{4} \frac{1}{(\text{rank}+60)}$$

Note: If the document or context is not present in a retrieved set, a value of zero is added to the equation.

IV. RESULTS AND DISCUSSION

If the number of contexts retrieved from each retriever varies from 1 to 10 in our RAG, where OUR RAG 10 means the RAG used has four retrievers and the output of each is 10, which are passed to the RRF technique and extract one context having the highest score for generation.

TABLE I EVALUATION METRICS FOR ALL FIVE RAGS

Evaluation Metrics	Correctness	Conciseness	Hallucination	Similarity
RAG contains BM25 retriever with one context for the generation	71	88.25	88.5	74.05
RAG contains text-embedding-3-large retriever with one context for the generation	75.25	85.99	95.5	76.7
RAG contains text-embedding-3-small retriever with one context for the generation	74.5	86	86.5	72.75
RAG contains all-mpnet-base-v2 retriever with one context for the generation	69	87	73	72.15
OUR RAG with all four retrievers with four contexts each, and the RRF technique which gives one context for generation	79.75	83.49	90	78.7

TABLE II VARIATION OF EVALUATION METRICS

	Correctness	Conciseness	Hallucination	Similarity
OURRAG10	78	82.25	96	76.1
OUR RAG 9	78.24	87.74	96.5	80.2
OUR RAG 8	79.25	89.25	91.75	75.9
OUR RAG 7	76.5	89	88.5	76.25
OUR RAG 6	76.25	83.24	94	76.4
OUR RAG 5	52.75	20.5	68.5	69.4
OUR RAG 4	79.75	83.49	90	78.7
OUR RAG 3	73.25	87.25	96	74.05
OUR RAG 2	74.75	81	95.5	77.5
OUR RAG 1	71.25	90.24	85.24	73.85

After obtaining the output files from all RAG pipelines, each containing the 20 questions, ground truth answers, retrieved context, and generated responses, the results are evaluated using the OpenEvals library. The evaluation is performed in terms of correctness, conciseness, hallucination, and similarity between the ground truth and the generated answer, as shown in Table I. In addition, for the proposed RAG framework, varying the number of retrieved contexts per retriever from 1 to 10 results in observable variations in the evaluation metrics, as presented in Table II. Four evaluation metrics, defined in the OpenEvals library, are used to assess all RAG pipelines using the GPT-4o mini model. These metrics are described as follows:

1. Correctness: The inputs for this metric are the question, the generated answer, and the ground truth. GPT-4o mini assigns a score between 0 and 1 based on the correctness of the generated answer with respect to the ground truth, where a value closer to 1 indicates better performance. The built-in correctness prompt provided to the evaluator model is as follows:

"You are an expert data labeler evaluating model outputs for correctness. Your task is to assign a score based on the following rubric: <Rubric> A correct answer: – Provides accurate and complete information – Contains no factual errors – Addresses all parts of the question – Is logically consistent – Uses precise and accurate

terminology When scoring, you should penalize: – Factual errors or inaccuracies – Incomplete or partial answers – Misleading or ambiguous statements – Incorrect terminology – Logical inconsistencies – Missing key information </Rubric> <Instructions> – Carefully read the input and output – Check for factual accuracy and completeness – Focus on correctness of information rather than style or verbosity </Instructions> <Reminder> The goal is to evaluate factual correctness and completeness of the response.</Reminder> <input> {inputs} </input> <output> {outputs} </output> Use the reference outputs below to help you evaluate the correctness of the response: <reference_outputs> {reference_outputs} </reference_outputs>"

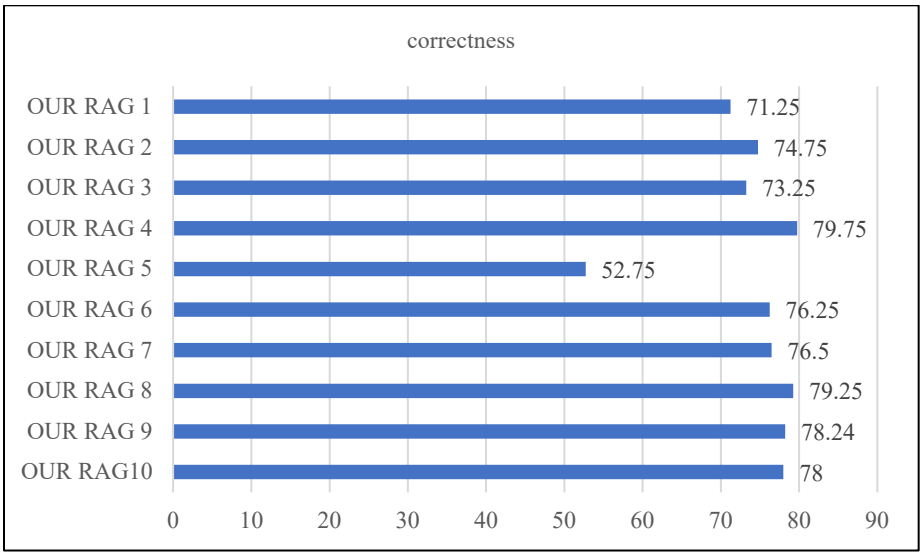

Fig.4 Variation of Correctness with the Number of Retrieved Contexts or Documents Before the RRF Technique in Our RAG.

2. Conciseness: The inputs for this metric are the question and the generated answer only. GPT-4o mini assigns a score between 0 and 1, where a value closer to 1 indicates a more concise response. The conciseness prompt, provided by the OpenEvals library, is as follows:

"You are an expert data labeler evaluating model outputs for conciseness. Your task is to assign a score based on the following rubric: <Rubric> A perfectly concise answer: – Contains only the exact information requested – Uses the minimum number of words necessary to convey the complete answer – Omits pleasantries, hedging language, and unnecessary context – Excludes meta-commentary about the answer or the model's capabilities – Avoids redundant information or restatements – Does not include explanations unless explicitly requested When scoring, you should deduct points for: – Introductory phrases such as "I believe" or "The answer is" – Hedging language such as "probably" or "as far as I know" – Unnecessary context or background information – Explanations when not requested – Follow-up questions or offers for more information – Redundant information or restatements – Polite phrases such as "hope this helps" </Rubric> <Instructions> – Carefully read the input and output – Identify unnecessary elements based on the rubric </Instructions> <Reminder> The goal is to reward responses that provide complete answers with no extraneous information. </Reminder> <input> {inputs} </input> <output> {outputs}</output>"

3. Hallucination: The inputs for this metric are the question, ground truth, retrieved context, and generated answer. GPT-4o mini assigns a score between 0 and 1, where a value closer to 0 indicates fewer hallucinations. The hallucination prompt provided by the OpenEvals library is as follows:

"You are an expert data labeler evaluating model outputs for hallucinations. Your task is to assign a score based on the following rubric: <Rubric> A response without hallucinations: – Contains only verifiable facts directly supported by the input context – Makes no unsupported claims or assumptions – Does not add speculative or imagined details – Maintains accuracy in dates, numbers, and specific details – Appropriately indicates uncertainty when information is incomplete </Rubric> <Instructions> – Read the input context thoroughly – Identify all claims made in the output – Cross-reference each claim with the input context – Note unsupported or contradictory information </Instructions> <Reminder> Focus solely on factual accuracy and contextual support. Style and presentation should not influence scoring. </Reminder> <context> {context} </context> <input> {inputs} </input> <output> {outputs} </output> <reference_outputs> {reference_outputs} </reference_outputs>"

4. Similarity: This metric measures the semantic similarity between the generated answer and the ground truth and assigns a score between 0 and 1, where a value closer to 1 indicates higher similarity.

V. CONCLUSION

The proposed RAG framework demonstrates superior performance in terms of correctness and similarity compared to other systems, achieving a correctness score of 79.75% and a similarity score of 78.7%. In addition, experiments were conducted to analyze the impact of varying the number of retrieved documents per retriever prior to applying the Reciprocal Rank Fusion (RRF) technique. The results indicate the presence of local maxima at even numbers of retrieved documents, with higher performance observed at even retrieval counts compared to their adjacent odd values. Furthermore, when varying the number of retrieved results from 1 to 10, a decrease in accuracy is observed at the midpoint, specifically when five documents are retrieved, as illustrated in Figure 4.

ACKNOWLEDGEMENT

Special thanks are extended to the Controller General of Accounts (CGA) for making the Civil Accounts Manual open source, which served as the 603-page PDF used to create the 20-question dataset with corresponding answers and to conduct the entire experimental evaluation.

Declaration of Conflicting Interests

The authors declare no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

Use of Artificial Intelligence (AI)-Assisted Technology for Manuscript Preparation

The authors confirm that no AI-assisted technologies were used in the preparation or writing of the manuscript, and no images were altered using AI.

ORCID

Suraj Singh Patwal https://orcid.org/0009-0006-9013-6885
Devashish Chauhan https://orcid.org/0009-0002-8868-6213
Shyam Ji https://orcid.org/0009-0007-9282-9711

REFERENCES

[1] H. Soudani, E. Kanoulas, and F. Hasibi, "Fine-tuning vs. retrieval-augmented generation for less popular knowledge," in *Proc. 2024 Annu. Int. ACM SIGIR Conf. Res. Dev. Inf. Retrieval Asia Pac. Reg.*, Dec. 2024, pp. 12–22.

[2] A. Salemi and H. Zamani, "Evaluating retrieval quality in retrieval-augmented generation," in *Proc. 47th Int. ACM SIGIR Conf. Res. Dev. Inf. Retrieval*, Jul. 2024, pp. 2395–2400.

[3] H. Yu, A. Gan, K. Zhang, S. Tong, Q. Liu, and Z. Liu, "Evaluation of retrieval-augmented generation: A survey," in *Proc. CCF Conf. Big Data*, Singapore, 2024, pp. 102–120.

[4] J. Chen, H. Lin, X. Han, and L. Sun, "Benchmarking large language models in retrieval-augmented generation," in *Proc. AAAI Conf. Artif. Intell.*, vol. 38, no. 16, Mar. 2024, pp. 17754–17762.

[5] Z. Jiang, F. F. Xu, L. Gao, Z. Sun, Q. Liu, J. Dwivedi-Yu, *et al.*, "Active retrieval-augmented generation," in *Proc. Conf. Empir. Methods Nat. Lang. Process.* (EMNLP), Dec. 2023, pp. 7969–7992.

[6] H. Li, Y. Su, D. Cai, Y. Wang, and L. Liu, "A survey on retrieval-augmented text generation," *arXiv preprint*, arXiv:2202.01110, 2022.

[7] Z. Rackauckas, "RAG-Fusion: A new take on retrieval-augmented generation," *arXiv preprint*, arXiv:2402.03367, 2024.

[8] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang, "Lost in the middle: How language models use long contexts," *Trans. Assoc. Comput. Linguist.*, vol. 12, pp. 157–173, 2024.

[9] B. Peng, Y. Zhu, Y. Liu, X. Bo, H. Shi, C. Hong, *et al.*, "Graph retrieval-augmented generation: A survey," *ACM Trans. Inf. Syst.*, 2024.

[10] H. Han, Y. Wang, H. Shomer, K. Guo, J. Ding, Y. Lei, *et al.*, "Retrieval-augmented generation with graphs (GraphRAG)," *arXiv preprint*, arXiv:2501.00309, 2024.

[11] R. T. de Lima, S. Gupta, C. B. Ramis, L. Mishra, M. Dolfi, P. Staar, and P. Vagenas, "Know your RAG: Dataset taxonomy and generation strategies for evaluating RAG systems," in *Proc. 31st Int. Conf. Comput. Linguistics: Ind. Track*, Jan. 2025, pp. 39–57.

[12] Z. Wang, J. Araki, Z. Jiang, M. R. Parvez, and G. Neubig, "Learning to filter context for retrieval-augmented generation," *arXiv preprint*, arXiv:2311.08377, 2023.

[13] Z. Jiang, X. Ma, and W. Chen, "LongRAG: Enhancing retrieval-augmented generation with long-context LLMs," *arXiv preprint*, arXiv:2406.15319, 2024.

[14] Z. Guo, L. Xia, Y. Yu, T. Ao, and C. Huang, "LightRAG: Simple and fast retrieval-augmented generation," *arXiv preprint*, arXiv:2410.05779, 2024.

[15] X. Wang, Z. Wang, X. Gao, F. Zhang, Y. Wu, Z. Xu, *et al.*, "Searching for best practices in retrieval-augmented generation," in *Proc. Conf. Empir. Methods Nat. Lang. Process.* (EMNLP), Nov. 2024, pp. 17716–17736.

[16] C. M. Chan, C. Xu, R. Yuan, H. Luo, W. Xue, Y. Guo, and J. Fu, "RQ-RAG: Learning to refine queries for retrieval-augmented generation," *arXiv preprint*, arXiv:2404.00610, 2024.

[17] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, *et al.*, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 9459–9474.

[18] Y. Yu, W. Ping, Z. Liu, B. Wang, J. You, C. Zhang, *et al.*, "RankRAG: Unifying context ranking with retrieval-augmented generation in LLMs," in *Adv. Neural Inf. Process. Syst.*, vol. 37, 2024, pp. 121156–121184.

[19] X. Cheng, X. Wang, X. Zhang, T. Ge, S. Q. Chen, F. Wei, *et al.*, "XRAG: Extreme context compression for retrieval-augmented generation with one token," in *Adv. Neural Inf. Process. Syst.*, vol. 37, 2024, pp. 109487–109516.

[20] K. Zhu, Y. Luo, D. Xu, Y. Yan, Z. Liu, S. Yu, *et al.*, "RAGEval: Scenario-specific RAG evaluation dataset generation framework," in *Proc. 63rd Annu. Meet. Assoc. Comput. Linguistics* (ACL), vol. 1, Jul. 2025, pp. 8520–8544.

[21] S. Es, J. James, L. E. Anke, and S. Schockaert, "RAGAS: Automated evaluation of retrieval-augmented generation," in *Proc. 18th Conf. Eur. Chapter Assoc. Comput. Linguistics: Syst. Demonstrations*, Mar. 2024, pp. 150–158.