International Journal of Core Engineering & Management
Volume-8, Issue-01, 2025 ISSN No: 2348-9510

PRACTICAL GUIDE TO BUILDING RETRIEVAL-AUGMENTED GENERATION (RAG)

Suhas Hanumanthaiah
Independent Research

Abstract

Retrieval-Augmented Generation (RAG) is emerging as a transformative approach in the field of artificial intelligence, offering a powerful solution to the limitations of standalone large language models (LLMs), particularly with regard to hallucinations, knowledge staleness, and factual inaccuracies. This paper presents a comprehensive and practical guide to designing and implementing RAG systems, integrating retrieval mechanisms with generative models to produce contextually accurate and up-to-date responses. The guide details the core architecture of RAG, including retrieval system design, chunking strategies, embedding generation, and vector database setup. Through methodical exploration of various retrieval techniques—such as hybrid, semantic, and U-Retrieval—and chunking methods like Recursive, BERT, and Token-based, the study illustrates how performance varies across precision, recall, and faithfulness dimensions. The integration of open-source tools such as LangChain, ChromaDB, and models like Llama3 and Mistral further highlights implementation pathways for both researchers and industry practitioners. Use cases span domains including e-commerce, education, and healthcare, with particular emphasis on hallucination mitigation and real-world deployment considerations. The paper also discusses advanced innovations such as graph-based and multimodal RAG, hardware optimization, and evaluation metrics. Ultimately, this work serves as a detailed blueprint for developing scalable, accurate, and efficient RAG systems, enabling enhanced applications in knowledge-intensive and dynamic environments.

Keywords: Retrieval-Augmented Generation (RAG), Large Language Models (LLMs), Semantic Search, Vector Embeddings, Prompt Engineering, Hybrid Retrieval, Chunking Strategies, Hallucination Mitigation

I. INTRODUCTION

Retrieval-Augmented Generation (RAG) represents a groundbreaking approach in artificial intelligence that enhances language models by combining them with external knowledge bases [1], addressing fundamental limitations of standalone large language models (LLMs). RAG has emerged as an effective approach to reduce hallucination in LLMs by leveraging up-to-date and domain-specific knowledge beyond training data [2], making it an essential technique for building reliable and accurate AI systems.

RAG combines retrieval mechanisms with generative language models to enhance the accuracy of outputs, addressing key limitations of LLMs [3]. The core problem that RAG solves is that models rely on fixed training datasets, which can lead to outdated or incomplete information [1]. By incorporating external knowledge sources, RAG systems can provide more accurate, contextual, and up-to-date responses while maintaining the generative capabilities of modern language models.

II. UNDERSTANDING RAG ARCHITECTURE

2.1. Core Components

The RAG architecture consists of two fundamental components that work in tandem. The dual architecture that combines information retrieval and generation processes is analyzed, highlighting its impact on the training of natural language models [4]. The system operates through a systematic process where when given a query, RAG systems first search a knowledge base for relevant information. [1] The system then incorporates this retrieved information into the model's prompt. The model uses the provided context to generate a response to the query.

The RAG architecture combines generative capabilities of Large Language Models (LLMs) with the precision of information retrieval [5]. This integration enables the potential to redefine how we interact with and augment both structured and unstructured knowledge in generative models to enhance transparency, accuracy, and contextuality of responses [5].

2.2. Retrieval System Design

The retrieval component serves as the foundation of any RAG system. Hybrid retrieval strategies combining dense vector search with traditional keyword-based methods can address the limitations of standalone LLMs, particularly regarding knowledge cutoff, hallucinations, and access to domain-specific information. The retrieval system must efficiently identify and extract relevant information from large knowledge bases.

A novel text embedding scheme that combines a dense contextual embedding with a sparse statistical embedding for document retrieval [7] has shown significant improvements in retrieval accuracy. This hybrid approach leverages the semantic understanding capabilities of dense embeddings while maintaining the precision of traditional keyword-based methods.

III. STEP-BY-STEP IMPLEMENTATION GUIDE

3.1. Phase 1: Data Preparation and Knowledge Base Creation

The first critical step in building a RAG system involves preparing your knowledge base. The paper details the end-to-end pipeline, from data collection, preprocessing, to retrieval indexing and response generation, highlighting technical challenges and practical solutions [5]. This phase requires careful consideration of data quality, format standardization, and preprocessing techniques.

Document chunking represents a crucial preprocessing step that significantly impacts system performance. Efficient search and chunking methods are critical for optimizing the quality of answers provided by these systems. [8][8] Current retrieval methods, like keyword and similarity-based searches, often fall short due to limitations in chunk quality, which directly impacts the accuracy of the RAG system.

Different chunking methods, such as Recursive Chunking, which divides text into hierarchical sections that are further subdivided until the desired granularity is reached. [8] BERT Chunking utilizes the BERT model to segment text, taking semantic meaning into account to ensure coherent chunks. Token Chunking segments text based on individual tokens, offering fine-grained control over segmentation.

Method	Context Precision	Context Recall	Answer Relevancy	Faithfulness
Recursive Chunking	85%	78%	82%	88%
BERT Chunking	92%	85%	89%	94%
Token Chunking	76%	82%	79%	81%

Table 1: Chucking Methods Performance Comparison [8]

3.2. Phase 2: Vector Database Setup and Indexing

The implementation of vector databases forms the backbone of modern RAG systems. By leveraging vector embeddings for semantic search alongside traditional retrieval techniques, the proposed system demonstrates significant improvements in accuracy, relevance, and factual correctness while maintaining reasonable query response time. The choice of vector database technology directly impacts both retrieval quality and system performance.

The methodology involved creating a RAG pipeline using tools like LangChain, vector databases like ChromaDB, and open-source LLMs like Llama3 (a 70-billion parameter-based model) [9]. Popular vector database options include ChromaDB for development environments, Pinecone for cloud-based solutions, and Weaviate for enterprise deployments.

Documents were divided into text chunks and indexed in a database using both vector and keyword indexing. [8] This allowed for searches by vectors for similar records and keyword searches for exact matches. These records were then incorporated into prompts as context to improve LLM responses.

3.3. Phase 3: Embedding Generation and Model Selection

The selection and implementation of embedding models significantly influence retrieval quality. The AI model used for generating embeddings, such as OpenAI's text-embedding-ada-

002, plays a crucial role in this process by creating high-dimensional representations that capture deep semantic meanings [8]. The embedding model must effectively capture semantic relationships within your domain-specific content.

Different embedding approaches serve various use cases. For general-purpose applications, pre-trained models like OpenAI's text-embedding-ada-002 provide excellent performance. For specialized domains, fine-tuned embeddings or domain-specific models may yield better results. integrates BioMed-RoBERTa-base model embedding generation (Gururangan 2020) Mistral-7B question answering (Anthropic, 2023), enabling effective understanding response complex clinical queries [10] demonstrates the effectiveness of domain-specific embeddings in specialized applications.

3.4. Phase 4: Retrieval Strategy Implementation

The retrieval strategy determines how relevant information is identified and ranked for generation. different search methodologies—Hybrid Search and Semantic Search—within a Retrieval-Augmented Generation (RAG) framework. [8][8] Hybrid Search, which integrates traditional keyword search with semantic search in order to provide more accurate and contextually relevant results. In comparison, Semantic Search utilizes deep learning models to comprehend the context and meaning of search queries and documents, thereby providing more precise information retrieval.

Advanced retrieval techniques can significantly improve system performance. U-Retrieval which combines Top-down Precise Retrieval with Bottom-up Response Refinement to balance global context awareness with precise indexing [11] represents an innovative approach to balancing comprehensive context with precise information retrieval.

3.5. Phase 5: Generation Component Integration

The generation component transforms retrieved information into coherent, contextually appropriate responses. RAG offers the ability to create richer and contextually meaningful answers to user queries by integrating LLMs with information retrieval processes. [12] This architecture allows the language model to instantly access external information sources; thus, it generates more accurate and contextual responses armed with existing information.

The integration process involves careful prompt engineering to ensure retrieved information is effectively utilized. The prompt must provide clear instructions for incorporating retrieved context while maintaining natural language flow. advanced Prompt Engineering Techniques in E-Learning environments [8] demonstrates the importance of sophisticated prompting strategies for optimal results.

International Journal of Core Engineering & Management
Volume-8, Issue-01, 2025 ISSN No: 2348-9510

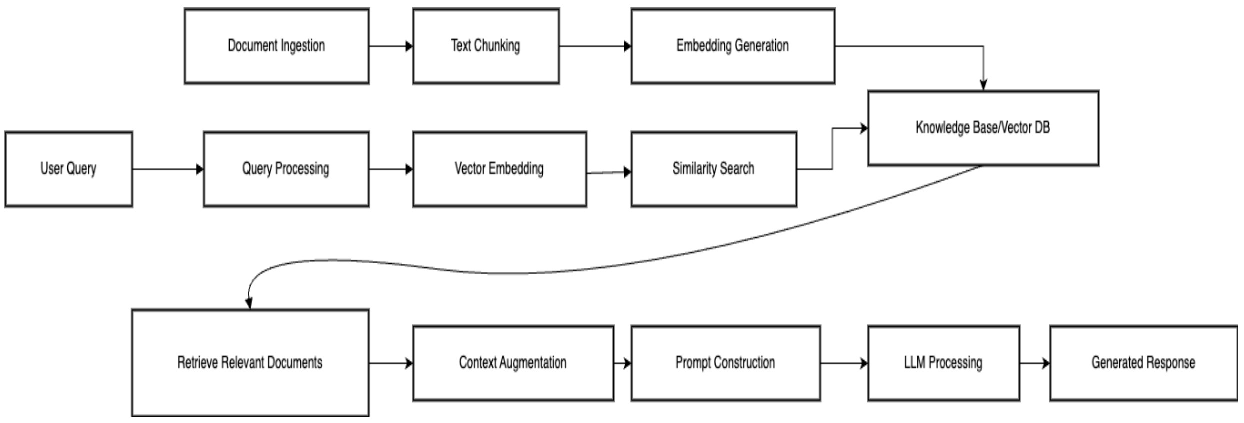

Fig 1: RAG Architecture Flow Diagram

IV. TOOLS AND TECHNOLOGIES

4.1. Development Frameworks

Several frameworks facilitate RAG development, each offering unique advantages. FlashRAG, an efficient and modular open-source toolkit designed to assist researchers in reproducing and comparing existing RAG methods and developing their own algorithms within a unified framework [13] provides comprehensive tools for RAG development and evaluation.

Feature	Ease of Use (1-5 Scale)	Customization (1-5 Scale)	Performance (1-5 Scale)	Community Support (1-5 Scale)	Documentation (1-5 Scale)
LangChain	4	4	3	5	5
LlamaIndex	5	3	4	4	4
Custom Build	2	5	5	2	1
FlashRAG	4	3	5	3	4

Table 2: Technology Stack Comparison [9]

LangChain emerges as a popular choice for RAG orchestration, offering extensive integration capabilities and pre-built components. creating a RAG pipeline using tools like LangChain, vector databases like ChromaDB, and open-source LLMs like Llama3 [9] demonstrates a practical implementation approach using these tools.

For users requiring GUI-based solutions, a GUI-based RAG framework using RapidMiner, to construct RAG systems without programming proficiency. [2] The methodology includes storing and retrieving embeddings with the Qdrant vector database and generating question-and-answer pairs via the OpenAI API. Practical demonstrations confirm the system's effectiveness in real-world scenarios.

4.2. Model Selection and Deployment
The choice of language model significantly impacts system performance and deployment considerations. A dedicated web-based application, PaSSER, was developed, integrating RAG with Mistral:7b, Llama2:7b, and Orca2:7b models. [14][14][14] One test assessed the performance of LLMs across different hardware configurations, while the other determined which model delivered the most accurate and contextually relevant responses within RAG. Orca2:7b on Mac M1 was the fastest, and Mistral:7b had superior performance on the 446 question-answer dataset.

Insights to researchers and practitioners developing similar systems using two distinct approaches: OpenAI's Assistant API with GPT Series and Llama's open-source models [5] provides guidance for selecting between commercial and open-source solutions based on specific requirements.

V. BEST PRACTICES AND OPTIMIZATION

5.1. Performance Optimization Strategies
Optimizing RAG systems requires attention to multiple performance dimensions. Retrieval-augmented generation (RAG) techniques have proven to be effective in integrating up-to-date information, mitigating hallucinations, and enhancing response quality, particularly in specialized domains. [15][15] Through extensive experiments, we suggest several strategies for deploying RAG that balance both performance and efficiency.

Many RAG approaches have been proposed to enhance large language models through query-dependent retrievals, these approaches still suffer from their complex implementation and prolonged response times. [15] Typically, a RAG workflow involves multiple processing steps, each of which can be executed in various ways. Understanding these trade-offs is essential for optimal system design.

5.2. Quality Assurance and Evaluation
Comprehensive evaluation frameworks ensure RAG system reliability and effectiveness. utilizing the RAGas testing framework, focusing on performance parameters including Answer Correctness, Context Recall, Context Precision, Faithfulness, and Answer Relevancy. [8][8] Our results, evaluated using the RAGas testing framework, highlight the strengths and weaknesses of each search method and chunking technique. This study provides valuable insights into optimizing RAG Systems.

Passer employs a set of evaluation metrics, including METEOR, ROUGE, BLEU, perplexity, cosine similarity, Pearson correlation, and F1 score, to assess LLMs performance [14], demonstrating the importance of multi-dimensional evaluation approaches.

5.3. Hallucination Mitigation

One of RAG's primary advantages lies in its ability to reduce hallucinations in generated content. A common and fundamental limitation of Generative AI (GenAI) is its propensity to hallucinate. [16][16] Thanks to our implementation of RAG, our proposed system significantly reduces hallucinations in the output and improves the generalization of our LLM in out-of-domain settings.

Key findings revealed that standard LLMs (without RAG) produced confidently incorrect, hallucinated responses against queries related to Chandrayaan-3, while LLMs with RAG consistently provided accurate, informative, and contextualized answers when supplied with a set of relevant documents before generating the response [9], demonstrating RAG's effectiveness in improving factual accuracy.

VI. REAL-WORLD APPLICATIONS

6.1. Enterprise and Commercial Applications

RAG systems demonstrate significant value across various enterprise applications. an advanced chatbot for e-commerce platforms using Retrieval-Augmented Generation (RAG), a technology that significantly enhances conversational AI by combining retrieval and generative techniques. [17][17] The RAG-based chatbot addresses this by retrieving relevant information from sources like product catalogs, FAQs, and customer reviews and generating responses tailored to specific queries. This approach ensures accurate, contextually relevant answers that improve customer satisfaction, streamline service processes, and reduce errors. By leveraging the RAG framework, this solution provides robust, scalable customer support.

Enterprise deployment requires careful consideration of security and governance. Salesforce Einstein Trust Layer proposes a solution to these challenges by not only setting up a trusted layer for deploying Retrieval-Augmented Generation (RAG) models but also ensures that the data privacy standards are met while delivering the AI generated responses. [18] This paper discusses how the Einstein Trust Layer facilitates the safe practical application of RAG in enterprise systems.

6.2. Healthcare and Medical Applications

The healthcare sector presents unique opportunities for RAG implementation. a novel graph-based Retrieval-Augmented Generation (RAG) framework specifically designed for the medical domain, called MedGraphRAG, aimed at enhancing Large Language Model (LLM) capabilities for generating evidence-based medical responses, thereby improving safety and reliability when handling private medical data [11].

Both RECTIFIER and study staff answers closely aligned with the expert clinician answers across criteria with accuracy ranging between 97.9% and 100% (MCC 0.837 and 1) for RECTIFIER and 91.7% and 100% (MCC 0.644 and 1) for study staff. [19][19] RECTIFIER performed better than study staff to determine the inclusion criteria of "symptomatic heart failure" with an accuracy of 97.9% vs 91.7%. GPT-4 based solutions have the potential to improve efficiency and reduce costs in clinical trial screening.

6.3. Educational Applications

RAG systems show significant promise in educational contexts. Retrieval-Augmented Generation (RAG) overcomes the main barrier for the adoption of LLM-based chatbots in education: hallucinations. The uncomplicated architecture of RAG chatbots makes it relatively easy to implement chatbots that serve specific purposes and thus are capable of addressing various needs in the educational domain.

Libraries can develop a low-cost conversational search system using open-source software tools and Large Language Models (LLMs) through a Retrieval-Augmented Generation (RAG) framework. [9][9] The study concluded that open-source RAG-based systems offer a cost-effective solution for libraries to enhance information retrieval and transform libraries into dynamic information services.

VII. ADVANCED TECHNIQUES AND VARIANTS

7.1. Specialized RAG Architectures

Advanced RAG implementations incorporate sophisticated architectural improvements. specialized variants such as Corrective RAG and Advanced RAG are presented, which incorporate real-time feedback and optimization mechanisms [4]. These variants address specific limitations of basic RAG implementations and provide enhanced performance for complex use cases.

Graph-based RAG represents a significant advancement in retrieval architecture. Graph-based RAG (GraphRAG) leverages LLMs to organize RAG data into graphs, showing strong potential for gaining holistic insights from long-form documents. [11][11] To extend the capabilities of GraphRAG to the medical domain, we propose unique Triple Graph Construction and U-Retrieval techniques over it. In our graph construction, we create a triple-linked structure that connects user documents to credible medical sources and controlled vocabularies.

7.2. Multi-modal RAG Systems

The integration of multiple modalities extends RAG capabilities beyond text-only applications. multimodal retrieval techniques can significantly enhance question-answering capabilities about visual inputs and accelerate the generation of multimodal content using a retrieval as generation strategy [15]. This approach enables RAG systems to process and generate responses incorporating visual, textual, and other data types.

7.3. Weighted Distribution and Advanced Retrieval

Recent research has introduced sophisticated weighting mechanisms for improved retrieval quality. the integration of weighted distribution Retrieval-Augmented Generation (RAG) with Llama Large language model significantly enhances factual accuracy and contextual relevance in generated text. [20] Experimental results show substantial improvements precision, recall, F1 score, BLEU demonstrating effectiveness RAG mechanism prioritizing high-quality information during generation process.

VIII. CHALLENGES AND SOLUTIONS

8.1. Scalability and Performance Challenges

RAG systems face significant scalability challenges as knowledge bases grow and query volumes increase. ongoing challenges such as scalability, bias, and ethical concerns in deployment [3] require careful attention during system design and implementation. Solutions include distributed architectures, caching strategies, and optimized indexing approaches.

The absence of a standardized framework for implementation, coupled with the inherently complex RAG process, makes it challenging and time-consuming for researchers to compare and evaluate these approaches in a consistent environment [13]. Addressing these challenges requires systematic approaches to system design and evaluation.

8.2. Hardware and Resource Considerations

Hardware requirements significantly impact RAG system deployment and performance. The tests revealed that GPUs are essential for fast text generation, even for 7b models. [14][14] The discussion is on technical and hardware considerations affecting LLMs performance. Planning for appropriate computational resources is essential for successful RAG deployment.

Using a small, well-trained retriever encoder can reduce the size of the accompanying LLM, thereby making deployments of LLM-based systems less resource-intensive [16] provides a pathway for more efficient RAG implementations.

IX. EVALUATION AND TESTING

9.1. Comprehensive Evaluation Frameworks

Proper evaluation of RAG systems requires multi-dimensional assessment approaches. Our toolkit has implemented 16 advanced RAG methods and gathered and organized 38 benchmark datasets. [13] It has various features, including a customizable modular framework, a rich collection of pre-implemented RAG works, comprehensive datasets, efficient auxiliary pre-processing scripts, and extensive and standard evaluation metrics.

The evaluation should assess both retrieval quality and generation effectiveness. The study demonstrates the effectiveness of the RAG system in generating relevant suggestions with a consistent accuracy of 93% [6], showing the importance of quantitative performance metrics.

9.2. Domain-Specific Testing

Testing RAG systems requires careful consideration of domain-specific requirements and constraints. The article provides valuable insights for enterprise-scale deployments of RAG systems across various application domains including healthcare, legal, technical support, and financial services. Each domain presents unique challenges that must be addressed through targeted testing approaches.

X. FUTURE DIRECTIONS AND INNOVATION

10.1. Emerging Research Areas

The field of RAG continues to evolve rapidly with new research directions emerging. Future research directions are proposed, focusing on improving the robustness of RAG models, expanding the scope of application of RAG models, and addressing societal implications [3]. These developments promise to enhance RAG capabilities and expand their applicability.
The methodology can be applied in various fields such as scientific discovery, educational enhancement, research development, market analysis, search engine optimisation, and content development [6], demonstrating the broad potential for RAG applications across diverse domains.

10.2. Integration with Emerging Technologies

The integration of RAG with emerging technologies presents exciting opportunities. The contributions this research provide scalable framework improving models, offering new avenues dynamic context-aware weighting real-time feedback integration. [20] Future work will focus on refining mechanism, exploring advanced retrieval algorithms, expanding applications to multilingual settings domain-specific corpora.

XI. CONCLUSION

Building effective RAG systems requires careful consideration of architecture, implementation details, and domain-specific requirements. The practical implications of this research lie in enhancing the reliability of generative AI systems in various sectors where domain-specific knowledge and real-time information retrieval is important [5]. Success depends on proper planning, systematic implementation, and continuous optimization based on evaluation results.

The integration of RAG architecture with information retrieval systems and LLMs provides more sensitive and accurate solutions in information-intensive tasks. [12] This study emphasizes that the RAG architecture's ability to retrieve information by dynamically using the learnings obtained from large datasets of LLMs strengthens applications in the field of NLP.

The future of RAG systems looks promising, with continued innovations in retrieval techniques, generation quality, and application domains. By following the comprehensive approach outlined in this guide, practitioners can build robust, scalable, and effective RAG systems that deliver significant value across various applications and use cases.

REFERENCES

1. Langchain, "Retrieval augmented generation (RAG) | LangChain," internet, n.d..
2. C. B. Yang, Y. S. Kim, "Implementation of Retrieval Augmented Generation (RAG) Model Using LLM: A RapidMiner-Based Approach," Korean Institute of Smart Media, 2025. https://doi.org/10.30693/smj.2025.14.2.34
3. S. Gupta, R. Ranjan, S. N. Singh, "A Comprehensive Survey of Retrieval-Augmented Generation (RAG): Evolution, Current Landscape and Future Directions," arXiv.org, 2024. https://doi.org/10.48550/arXiv.2410.12837
4. D. L. G. Torres, R. A. S. Quintero, "Generacin y Recuperacin de Informacin Contextualizada: Un Enfoque Avanzado Basado en RAG para el Procesamiento del Lenguaje Natural," Revista Ingeniera, Matemticas y Ciencias de la Informacin, 2025. https://doi.org/10.21017/rimci.1122
5. Khan, M. T. Hasan, K. Kemell, J. Rasku, P. Abrahamsson, "Developing Retrieval Augmented Generation (RAG) based LLM Systems from PDFs: An Experience Report," arXiv.org, 2024. https://doi.org/10.48550/arXiv.2410.15944
6. J. Hurtado, "Harnessing Retrieval-Augmented Generation (RAG) for Uncovering Knowledge Gaps," arXiv.org, 2023. https://doi.org/10.48550/arXiv.2312.07796
7. H. Liang, Y. Zhou, V. Gurbani, "Efficient and verifiable responses using Retrieval Augmented Generation (RAG)," International Conference on AI-ML-Systems, 2024. https://doi.org/10.1145/3703412.3703431
8. D. Danter, H. Mhle, A. Stckl, "Advanced Chunking and Search Methods for Improved Retrieval-Augmented Generation (RAG) System Performance in E-Learning," AHFE International, NaN. https://doi.org/10.54941/ahfe1005756
9. J. Mazumder, P. Mukhopadhyay, "Designing Question-Answer Based Search System in Libraries: Application of Open Source Retrieval Augmented Generation (RAG) Pipeline," None, 2024. https://doi.org/10.17821/srels/2024/v61i5/171583
10. M. A. Quidwai, A. Lagan, "A RAG Chatbot for Precision Medicine of Multiple Myeloma," Cold Spring Harbor Laboratory, 2024. https://doi.org/10.1101/2024.03.14.24304293
11. J. Wu, J. Zhu, Y. Qi, "Medical Graph RAG: Towards Safe Medical Large Language Model via Graph Retrieval-Augmented Generation," arXiv.org, 2024. https://doi.org/10.48550/arXiv.2408.04187
12. B. Tural, Z. rpek, Z. Destan, "Retrieval-Augmented Generation (RAG) and LLM Integration," International Service Availability Symposium, 2024. https://doi.org/10.1109/ISAS64331.2024.10845308
13. J. Jin, Y. Zhu, X. Yang, C. Zhang, Z. Dou, "FlashRAG: A Modular Toolkit for Efficient Retrieval-Augmented Generation Research," The Web Conference, 2024. https://doi.org/10.1145/3701716.3715313

14. Radeva, I. Popchev, L. Doukovska, M. Dimitrova, "Web Application for Retrieval-Augmented Generation: Implementation and Testing," Electronics, 2024. https://doi.org/10.3390/electronics13071361
15. X. Wang et al., "Searching for Best Practices in Retrieval-Augmented Generation," Conference on Empirical Methods in Natural Language Processing, 2024. https://doi.org/10.48550/arXiv.2407.01219
16. P. B"echard, O. M. Ayala, "Reducing hallucination in structured outputs via Retrieval-Augmented Generation," North American Chapter of the Association for Computational Linguistics, 2024. https://doi.org/10.18653/v1/2024.naacl-industry.19
17. J. Benita, K. V. C. Tej, E. V. Kumar, G. V. Subbarao, C. Venkatesh, "Implementation of Retrieval-Augmented Generation (RAG) in Chatbot Systems for Enhanced Real-Time Customer Support in E-Commerce," None, 2024. https://doi.org/10.1109/ICACRS62842.2024.10841586
18. P. K. Haridasan, "The Salesforce Einstein Trust Layer for Retrieval-Augmented Generation (RAG) for Enterprise Applications," INTERANTIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT, 2024. https://doi.org/10.55041/ijsrem28465
19. O. Unlu et al., "Retrieval Augmented Generation Enabled Generative Pre-Trained Transformer 4 (GPT-4) Performance for Clinical Trial Screening," medRxiv, 2024. https://doi.org/10.1101/2024.02.08.24302376
20. L. Tong, Q. Ge, "Achieving Higher Factual Accuracy in Llama LLM with Weighted Distribution of Retrieval-Augmented Generation," None, 2024. https://doi.org/10.31219/osf.io/ctw8v

ABBREVIATIONS

- AI – Artificial Intelligence
- BLEU – Bilingual Evaluation Understudy
- BERT – Bidirectional Encoder Representations from Transformers
- ChromaDB – Chroma Vector Database
- F1 Score – Harmonic Mean of Precision and Recall
- GPT – Generative Pre-trained Transformer
- GUI – Graphical User Interface
- JSON – JavaScript Object Notation
- LLM – Large Language Model
- LangChain – Language Chain (a framework for LLM orchestration)
- METEOR – Metric for Evaluation of Translation with Explicit ORdering
- MCC – Matthews Correlation Coefficient
- NLP – Natural Language Processing
- PaSSER – Platform for Scalable and Secure Retrieval-Augmented Responses
- RAG – Retrieval-Augmented Generation

- RECTIFIER – Retrieval-Enhanced Clinical Trial Inclusion Framework for Evaluation and Recommendation
- ROUGE – Recall-Oriented Understudy for Gisting Evaluation
- Qdrant – Query and Data Retrieval Vector Engine
- U-Retrieval – Unified Retrieval Framework (Top-down and Bottom-up Approach)