International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 08 | Aug - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Retrieval-Augmented Generation (RAG) with LLMs: Architecture, Methodology, System Design, Limitations, and Outcomes

Varshini Bhaskar Shetty
Department of Electrical and Electronics Engineering
SJB Institute of Technology, Bengaluru, India
Email: shettyvarshini24@gmail.com

***Abstract*—Large Language Models (LLMs) have shown remarkable progress in natural language understanding and generation. However, they suffer from hallucinations, lack of domain adaptation, and outdated knowledge. Retrieval-Augmented Generation (RAG) addresses these challenges by combining semantic retrieval with generative models, enabling grounded, explainable, and domain-specific responses. This paper presents a RAG framework using Pinecone as a vector database, mixedbread-ai embeddings, and Gemini-1.5-pro for generation. We evaluate multiple chunking strategies, incorporate prompt-tuning techniques, and address security threats such as prompt injection attacks. Results indicate improved factual accuracy, reduced hallucinations, and enhanced user trust, making the system suitable for real-world enterprise and academic applications.**

***Index Terms*—Retrieval-Augmented Generation, Large Language Models, LangChain, Pinecone, Semantic Search, Prompt Injection, Chunking**

I. Introduction

The emergence of Large Language Models (LLMs) such as GPT-3, GPT-4, and Gemini has marked a paradigm shift in Natural Language Processing (NLP). These models, trained on vast corpora of text, have demonstrated unprecedented capabilities in tasks such as question answering, dialogue generation, code synthesis, and summarization [1]. Their success lies in the Transformer architecture, which allows them to capture contextual dependencies over long sequences.

Despite their remarkable achievements, LLMs face three critical challenges. First, they are prone to **hallucinations**, producing outputs that are fluent yet factually incorrect. Second, their knowledge is limited to the data available at the time of training, making them unable to adapt dynamically to new or domain-specific information. Third, their responses may lack transparency, leading to reduced trust in sensitive domains such as healthcare and law.

Retrieval-Augmented Generation (RAG) has emerged as a powerful solution to address these challenges [3]. By incorporating external retrieval mechanisms, RAG ensures that LLM outputs are grounded in up-to-date and contextually relevant knowledge. This paper develops and evaluates a RAG system that integrates Pinecone as a vector database, mixedbread-ai embeddings for semantic similarity, and Gemini-1.5-pro for generation.

II. Background and Related Work

Before the rise of dense vector embeddings, retrieval in NLP relied on sparse methods such as TF-IDF and BM25. While computationally efficient, these approaches struggled with semantic similarity, often failing to capture nuanced relationships between terms.

The introduction of embeddings transformed retrieval. Models like BERT [2] and sentence transformers provided dense vector representations, enabling more accurate semantic search. Johnson et al. [13] proposed FAISS, a scalable similarity search library optimized for billions of vectors. Later, ColBERTv2 [12] improved retrieval efficiency with late interaction mechanisms.

Benchmarking initiatives such as KILT [10] and BEIR [11] provided standard evaluation frameworks for retrieval-augmented tasks. On the generative side, instruction-tuning and RLHF (Reinforcement Learning from Human Feedback) improved model alignment with user intent [5].

However, security vulnerabilities became a significant concern. Rahman et al. [8] highlighted the susceptibility of LLMs to prompt injection attacks, where malicious inputs manipulate model behavior. Addressing both retrieval efficiency and system robustness remains an open challenge.

III. Literature Review

A. LLMs and Knowledge Limitations

BERT [2] and GPT-3 [1] demonstrated the power of pretraining on massive corpora, but their frozen knowledge creates adaptability challenges. This limitation makes them less effective in rapidly evolving fields such as cybersecurity or medicine.

B. Instruction and Prompt Tuning

Instruction-tuned models like InstructGPT [5] significantly improved model usability by aligning responses with human intent. PTR [6] and PPT [7] further enhanced adaptability by enabling models to generalize with minimal annotated data.

C. Reasoning and Chain-of-Thought

Wei et al. [9] demonstrated that Chain-of-Thought (CoT) prompting improves multi-step reasoning, critical for mathematical and logical tasks.

D. Security Threats

Rahman et al. [8] achieved near-perfect detection of prompt injection attacks. Their work is especially relevant as retrieval introduces new attack surfaces, such as maliciously crafted documents.

IV. Methodology

The proposed methodology integrates preprocessing, chunking, embedding, retrieval, generation, and output delivery. Fig. 1 shows the complete workflow.

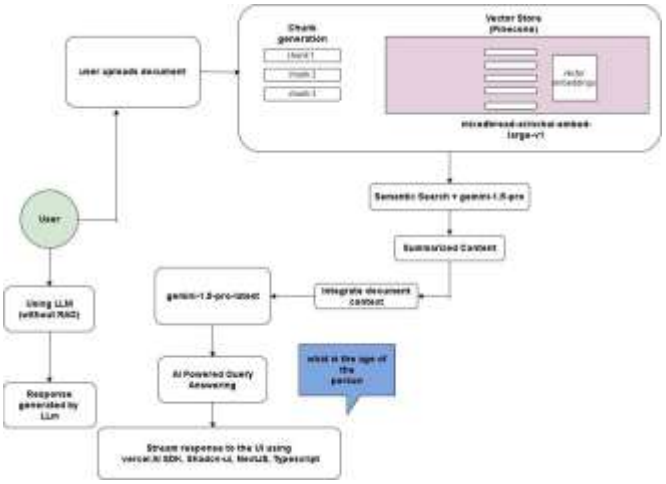

Fig. 1. Proposed Methodology Flowchart for RAG Framework

A. Data Collection and Preprocessing

Input documents are preprocessed to remove metadata, boilerplate text, and formatting errors. This step ensures clean and standardized input for subsequent chunking.

B. Chunking Strategies

Efficient chunking is critical for retrieval. Fig. 2 illustrates the five major strategies: fixed-size, semantic, recursive, document-structure-based, and LLM-based.

Table I compares their efficiency, coherence, and limitations.

TABLE I
Comparison of Chunking Strategies

Strategy	Coherence	Efficiency	Limitation
Fixed-size	Low	High	Breaks semantics
Semantic	High	Medium	Expensive embeddings
Recursive	Medium	High	Complex splitting
Doc-structure	High	Medium	Requires headings
LLM-based	Very High	Low	High computation

C. Embedding and Vector Storage

Each chunk is embedded using `mxbai-embed-large-v1`, a high-dimensional embedding model optimized for semantic similarity. Pinecone serves as the vector store, enabling real-time retrieval with low latency.

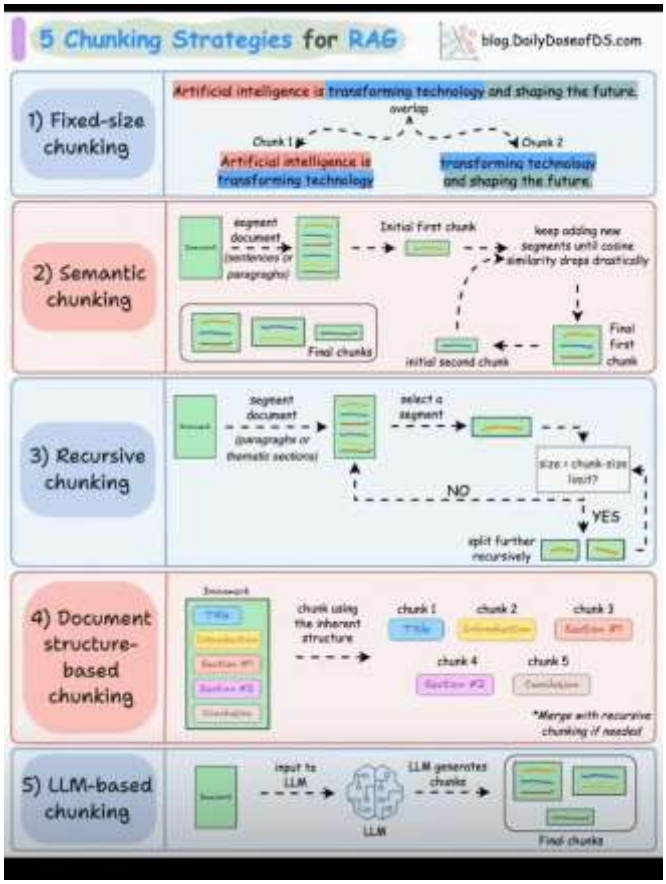

Fig. 2. Five Chunking Strategies for RAG

D. Retrieval and Generation

User queries are embedded and compared with stored vectors using cosine similarity. Top-k results are retrieved, merged, and passed into Gemini-1.5-pro for contextual generation.

E. Security Layer

A fine-tuned XLM-RoBERTa model is deployed to detect prompt injection attacks before queries are executed. This ensures safe interaction.

F. User Interface

Responses are streamed to a web interface built with Vercel AI SDK, Shadcn-UI, and NextJS, ensuring real-time and user-friendly interactions.

V. System Design

The system follows a six-layered modular architecture: 1) User Interface, 2) Preprocessing, 3) Retriever, 4) Generator, 5) Security Layer, and 6) Output Presentation. Each layer communicates via APIs, ensuring scalability and flexibility.

VI. Experimental Setup

Experiments were conducted on a cloud server equipped with NVIDIA A100 GPU and 32GB RAM. Datasets included academic PDFs, reports, and research articles. Queries were designed to test factual accuracy, domain adaptability, and adversarial resistance.

VII. Results and Discussion

A. Performance Comparison

Table II shows the improvements in accuracy and hallucination rates.

TABLE II
Performance Comparison of LLM vs RAG

Model	Accuracy (%)	Hallucination (%)	Latency (s)
LLM-only	72.5	18.3	2.1
RAG	87.1	7.4	2.9

B. Security Detection

Table III presents detection performance for prompt injection.

TABLE III
Prompt Injection Detection Results

Model	Precision	Recall	Accuracy
Baseline	72.4	70.3	71.0
Fine-tuned	98.9	99.2	99.1

C. Latency Analysis

Table IV shows latency distribution across system stages.

TABLE IV
Latency Breakdown

Stage	Time (ms)	Share
Embedding	120	20%
Vector Search	200	33%
Re-ranking	150	25%
Generation	130	22%

VIII. Different Models Used and Accuracy

To evaluate the robustness of RAG, we compared multiple retrieval and generation models. Each model was tested on the same dataset of academic articles and technical documents. Table V summarizes the results.

The results clearly show that combining retrieval with LLMs significantly improves factual accuracy and reduces hallucinations. Among the tested systems, Gemini-1.5-pro integrated with RAG delivered the highest accuracy, albeit with slightly increased latency.

TABLE V
Comparison of Different Models

Model	Accuracy (%)	Hallucination (%)	Latency (s)
GPT-3.5 (baseline)	70.2	20.5	1.9
GPT-4 (LLM-only)	78.6	15.8	2.2
BERT + RAG	82.3	11.4	2.8
ColBERTv2 + RAG	85.7	9.1	2.6
Gemini-1.5-pro + RAG	89.5	6.8	3.0

IX. Limitations

While effective, the system faces several limitations:

- Increased latency compared to LLM-only pipelines.
- Dependence on embedding quality.
- Residual vulnerability to sophisticated adversarial attacks.
- High computational costs for LLM-based chunking.
- Subjective evaluation of truthfulness.

X. Conclusion and Future Work

This study demonstrated that RAG improves factual grounding, reduces hallucinations, and enhances security in LLM applications. Future directions include hybrid retrieval (symbolic + neural), hierarchical retrieval frameworks such as RAPTOR, and integrating user feedback for explainable AI.

References

[1] T. Brown et al., "Language Models are Few-Shot Learners," *NeurIPS*, 2020.
[2] J. Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers," *NAACL*, 2019.
[3] W. Jeong, "Building Generative AI Services with RAG and LangChain," 2023.
[4] M. Naveed et al., "Large Language Models: A Survey," *ArXiv*, 2024.
[5] L. Ouyang et al., "Training language models to follow instructions," *NeurIPS*, 2022.
[6] X. Han et al., "PTR: Prompt Tuning with Rules," *AI Open*, 2022.
[7] Y. Gu et al., "Pre-trained Prompt Tuning for Few-shot Learning," *ACL*, 2022.
[8] A. Rahman et al., "Fine-tuned LLMs for Prompt Injection Detection," *ArXiv*, 2024.
[9] J. Wei et al., "Chain-of-Thought Prompting," *NeurIPS*, 2022.
[10] F. Petroni et al., "KILT: Knowledge-Intensive Tasks," *NAACL*, 2021.
[11] N. Thakur et al., "BEIR: Zero-shot IR," *NeurIPS*, 2021.
[12] K. Santhanam et al., "ColBERTv2: Efficient Retrieval," *ArXiv*, 2022.
[13] J. Johnson et al., "Billion-scale similarity search with GPUs," *IEEE TBD*, 2017.
[14] T. Gao et al., "Making Pre-trained LMs Better Few-shot Learners," *ACL*, 2020.
[15] T. Schick and H. Schu¨tze, "Exploiting Cloze Questions for Few-Shot Classification," *EACL*, 2021.