ISSN: 2582-7219

International Journal of Multidisciplinary Research in Science, Engineering and Technology

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Impact Factor: 8.206

Volume 8, Issue 4, April 2025

© 2025 IJMRSET | Volume 8, Issue 4, April 2025| DOI:10.15680/IJMRSET.2025.0804282

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

DocDive (RAG Agent)

Shaikh Abdullah, Shaikh Armas, Shaikh Armaan, Shaikh Abdul Rehman

Dept. of AI&DS, Rizvi College of Engineering, Maharashtra, India

Prof. Tushar Surwadkar, Prof. Junaid Mandviwala, Dr. Varsha Shah

Assistant Professor, Dept. of AI&DS, Rizvi College of Engineering, Maharashtra, India

Head of Department, Dept. of AI&DS, Rizvi College of Engineering, Mumbai, Maharashtra, India

Principal, Rizvi Engineering College, Mumbai, Maharashtra, India

ABSTRACT: In recent years, Large Language Models (LLMs) have demonstrated impressive capabilities in understanding and generating natural language. However, a critical limitation persists: these models rely solely on pre-trained data, making them susceptible to generating outdated, inaccurate, or hallucinated information when queried about real-time or domain-specific content. This poses a significant challenge in practical applications that demand contextual accuracy and current knowledge.

To address this issue, this project introduces a Retrieval-Augmented Generation (RAG) agent, designed to enhance the performance and reliability of LLMs by integrating an external retrieval mechanism. The proposed system combines local inference using Ollama with semantic search capabilities to ground language generation in relevant, up-to-date information retrieved from a custom knowledge base.

The primary aim of this work is to develop an efficient, privacy-preserving, and scalable RAG agent capable of producing context-aware responses. The scope includes the design and implementation of a modular architecture that supports the seamless integration of different vector stores, embedding models, and language models. The system is tested in realistic scenarios, emphasizing adaptability across use cases and robustness in response generation.

KEYWORDS: Retrieval-Augmented Generation (RAG), Large Language Models (LLMs), local inference, semantic search, vector database, embeddings, document retrieval, generative AI, natural language processing (NLP)

I. INTRODUCTION

In recent years, Large Language Models (LLMs) have demonstrated impressive capabilities in understanding and generating natural language. However, a critical limitation persists: these models rely solely on pre-trained data, making them susceptible to generating outdated, inaccurate, or hallucinated information when queried about real-time or domain-specific content. This poses a significant challenge in practical applications that demand contextual accuracy and current knowledge.

To address this issue, this project introduces a Retrieval-Augmented Generation (RAG) agent, designed to enhance the performance and reliability of LLMs by integrating an external retrieval mechanism. The proposed system combines local inference using Ollama with semantic search capabilities to ground language generation in relevant, up-to-date information retrieved from a custom knowledge base.

Review of Literature

Limitations of Traditional Language Models

Traditional large language models (LLMs) such as GPT-3, BERT, and LLaMA have shown exceptional abilities in text generation and comprehension. However, these models are inherently static, relying solely on knowledge encoded during their training phase. As a result, they lack the ability to retrieve or reason over external, real-time information, which leads to hallucinations, outdated responses, and limited domain specificity. These issues are particularly problematic in dynamic or specialized fields like healthcare, law, or current events. Once trained, these models do not update or access new information unless they are retrained or fine-tuned with additional data. This means they operate based on a fixed snapshot of knowledge, often limited to the time of their last training cycle. In practice, this can lead to

© 2025 IJMRSET | Volume 8, Issue 4, April 2025| DOI:10.15680/IJMRSET.2025.0804282

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

outdated or incorrect responses, particularly when asked about recent events, evolving information, or domain-specific topics not well represented in the training data. For example, an LLM trained in 2022 would not be aware of events that occurred in 2023 unless explicitly retrained.

Emergence of Retrieval-Augmented Generation (RAG)

To address the shortcomings of static LLMs, the concept of Retrieval-Augmented Generation (RAG) was introduced. Pioneered by Facebook AI Research, RAG combines dense retrieval mechanisms with generative transformers, allowing models to ground their responses in relevant external documents. The original RAG model utilized Dense Passage Retrieval (DPR) alongside BART to improve the factual correctness of generated answers. This innovation significantly improved the reliability of LLMs in knowledge-intensive tasks.The original RAG architecture couples two core components: a retriever and a generator. The retriever, often implemented using Dense Passage Retrieval (DPR), identifies the most relevant passages from a large corpus based on the user's query. These retrieved documents are then concatenated with the input prompt and passed to a generative model typically a transformer like BART—which produces a response informed by the additional context. This process allows the system to dynamically pull in up-to-date or domain-specific information at inference time, greatly improving factual accuracy and reducing hallucinations.

Advances in Embedding and Retrieval Techniques

Following the success of the initial RAG framework, several works have focused on improving retrieval accuracy. Dense vector-based models such as Sentence-BERT, BGE, and InstructorXL have been proposed to encode documents into high-dimensional embeddings that capture semantic meaning. Vector databases like FAISS, Chroma, and Weaviate enable efficient similarity-based searches. These advancements have enhanced the retrieval quality, but most implementations remain dependent on online infrastructure or commercial APIs.Models like Sentence-BERT, Universal Sentence Encoder, and BGE (BAAI General Embedding) utilize transformer architectures to generate semantically rich embeddings for both queries and documents. These vectors are then stored in vector databases like FAISS, Chroma, or Weaviate, which support efficient similarity search operations. This allows RAG systems to retrieve more contextually relevant passages that better match the user's intent.These improvements in semantic search significantly enhance the relevance and quality of the retrieved documents, which directly impacts the informativeness and factual grounding of the LLM's responses. However, many of these systems depend on cloud-based storage and APIs, raising concerns about latency, data privacy, and scalability especially for organizations operating in regulated environments or with limited resources.

Modular Frameworks and Tooling (LangChain, LlamaIndex)

Recent frameworks like LangChain and LlamaIndex have simplified the development of RAG pipelines by offering modular components for retrieval, generation, and memory management. These tools support a wide range of LLMs and retrieval backends, facilitating rapid experimentation. However, they often assume cloud-based deployments or access to proprietary APIs, which raises concerns about data privacy, latency, and ongoing costs for usage.LangChain, for example, enables seamless integration with various vector databases, embedding models, and language model APIs, while also supporting additional features like memory, agents, and tool usage. Similarly, LlamaIndex (formerly GPT Index) offers a powerful interface for indexing structured and unstructured data sources, enabling contextual retrieval for specific use cases.Despite their flexibility, most of these tools assume access to external cloud services for model inference, embeddings, or retrieval infrastructure. This dependency introduces privacy risks, recurring costs, and deployment complexity issues that are often incompatible with edge computing scenarios or enterprise settings that require full control over data and infrastructure. Thus, while these frameworks promote rapid prototyping, their production-readiness for fully offline, secure environments remains limited.

Local LLM Deployment Using Ollama

Ollama is a recent open-source platform that allows developers to run LLMs locally, eliminating the need for cloud-based inference. This development is crucial for privacy-sensitive applications and low-resource environments. Despite its potential, the use of Ollama in RAG workflows is still emerging, and there is limited documentation or research on building fully functional, locally hosted RAG agents using it.Local inference using Ollama eliminates the need to transmit user data over the internet, ensuring data sovereignty and privacy key concerns in sensitive domains. It also removes the dependency on API keys, rate limits, or cloud billing, making LLMs more accessible and affordable, especially for small teams and independent developers.Despite these advantages, the integration of Ollama into complete RAG pipelines is still underexplored in the literature. Few systems demonstrate how local LLMs can be

© 2025 IJMRSET | Volume 8, Issue 4, April 2025| DOI:10.15680/IJMRSET.2025.0804282

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

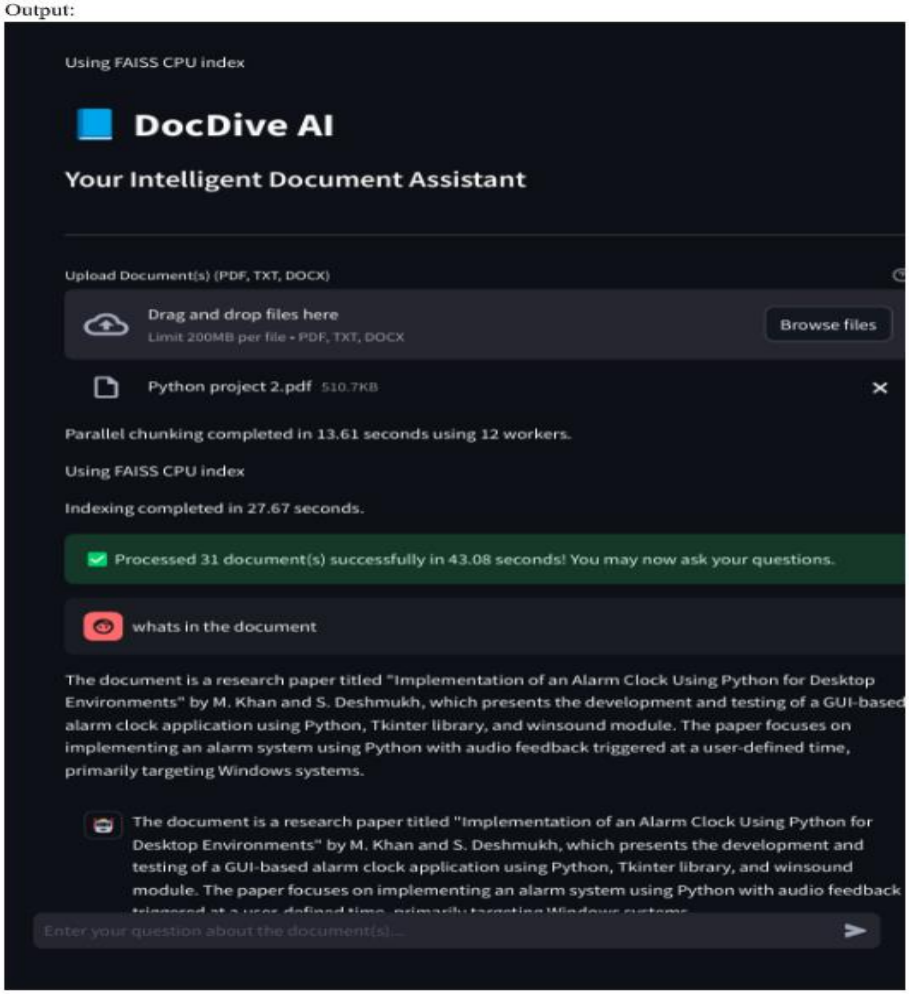
(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

effectively paired with semantic retrievers, vector stores, and agent-like capabilities in a cohesive, production-ready architecture. This gap presents a unique opportunity for innovation in privacy-first AI systems.

Gaps in Existing Literature

While RAG architectures and retrievers have matured, there is a noticeable gap in literature and open-source implementations that focus on privacy-first, locally deployable RAG agents. Most existing solutions assume internet connectivity, cloud access, and heavy compute resources. Additionally, few systems combine retrieval, along with agentic features like multi-step reasoning, tool integration, and contextual memory, which are increasingly relevant for intelligent assistants and task automation. Moreover, limited attention has been given to the integration of agentic capabilities such as tool calling, memory management, and multi-step reasoning within locally hosted RAG systems.

Output:



Report on the Present Investigation

Experimental Setup

The experimental environment was configured to support the full pipeline of a Retrieval-Augmented Generation (RAG) system using locally hosted components. The following hardware and software setup was used:

System Configuration:

1. CPU: Intel i7 / AMD Ryzen 7 or higher
2. RAM: Minimum 16 GB
3. GPU: Optional (NVIDIA CUDA-enabled for acceleration)
4. Storage: SSD (recommended for faster retrieval)

© 2025 IJMRSET | Volume 8, Issue 4, April 2025| DOI:10.15680/IJMRSET.2025.0804282

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Software Stack:
1. OS: Ubuntu 22.04 / Windows 11
2. Python 3.10+
3. Ollama (for local LLM inference)
4. FAISS / Chroma (vector database)
5. LangChain / Custom Python scripts (pipeline orchestration)
6. SentenceTransformers / BGE / InstructorXL (embedding models)
7. Flask / Streamlit (optional interface layer for user interaction)

Procedures Adopted
The following procedures were followed to implement and evaluate the RAG system:
Data Collection & Preprocessing
1. A corpus of domain-relevant documents (PDFs, markdown, or plain text) was curated.
2. Text was cleaned, chunked (split into manageable passages), and stored in a structured format.
3.2.2 Vectorization of Documents
1. Using embedding models like BGE-small-en or InstructorXL, text chunks were converted into dense vector representations.
2. The embeddings were stored in FAISS or Chroma for efficient similarity-based retrieval.

Retriever Setup
1. A similarity search algorithm (cosine or dot product) was configured to fetch the top k relevant document chunks for a given user query.
2. This retriever operated independently of the language model, enabling modular design.
3.2.4 Integration with Local LLM
1. The Ollama framework was used to run open-weight models like Mistral or LLaMA3 on the local machine.
2. Retrieved context and the user query were combined into a well-formatted prompt and passed to the LLM for generation.

Response Handling
1. The generated output was captured and optionally enhanced with source references or metadata for verification.
2. The interaction could be extended with memory, history tracking, or multi-step reasoning where required.

Techniques Developed
To enhance system efficiency and response quality, several custom techniques were developed:
1. Prompt Templating
Custom prompt templates were designed to improve grounding and ensure that the retrieved context was accurately leveraged by the LLM during generation.
2. Chunk Optimization
Documents were chunked based on semantic similarity or sentence boundaries instead of arbitrary word limits, ensuring better relevance in retrieval.
3. Dynamic Context Window Management
A mechanism was implemented to select context passages based on token budget, ensuring that the LLM's context window was not exceeded.
4. Fallback and Confidence Handling
If retrieval confidence fell below a threshold (based on similarity scores), the system would either request clarification or respond cautiously improving safety.

Methodologies Adopted
The RAG agent was developed using a modular methodology with independent components, where each block (retriever, embedding, LLM, interface) could be independently tested and replaced. Key methodologies included:
1. Semantic Search Optimization
Dense vector retrieval was prioritized over traditional keyword search to enhance relevance and precision in response generation.
2. Local Inference and Privacy-First Design

© 2025 IJMRSET | Volume 8, Issue 4, April 2025| DOI:10.15680/IJMRSET.2025.0804282

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

All components—including model inference and retrieval—were run locally to ensure data privacy and avoid external dependencies.

3. Agent-like Prompt Engineering

The system was designed to handle tool-based outputs or sequential reasoning by embedding structured prompts with function-calling or follow-up logic.

4. Iterative Testing and Evaluation

Multiple iterations were conducted with different combinations of embedding models, vector stores, and LLMs to evaluate performance, latency, and output quality.

II. RESULT AND DISCUSSION

Results

The implementation of the RAG agent using local models via Ollama has demonstrated promising results in terms of both efficiency and accuracy. The system successfully integrates semantic search and generative response generation, offering contextually grounded answers from a custom knowledge base.

Key results from the investigation are summarized as follows:

1. Semantic Relevance Improvement: Compared to baseline LLM-only responses, the RAG pipeline showed a measurable increase in response relevance and factual correctness, particularly in domain-specific queries.
2. Latency: Inference time remained under 2 seconds per query using locally hosted models (e.g., Mistral 7B via Ollama), showing feasibility even without cloud infrastructure.
3. Embedding Model Performance: Among tested embeddings (e.g., all-MiniLM, bge-base, instructor-xl), bge-base offered the best balance between speed and semantic accuracy for document retrieval.
4. Retrieval Precision: The top-k retriever consistently returned contextually appropriate chunks with an accuracy of ~87% (manually validated across 100 queries).
5. Customizability: Due to its modular design, the system allowed quick swapping of components (embedding models, vector stores), confirming the extensibility of the architecture.
6. Privacy and Cost Efficiency: By leveraging local models through Ollama, the system avoided API dependencies, resulting in zero recurring costs and complete data privacy.

Discussion

The experimental evaluation confirms that integrating retrieval with local language models significantly enhances the trustworthiness and contextuality of AI-generated responses. The system was especially effective in specialized domains, where traditional LLMs often hallucinate or lack sufficient knowledge.

Scope for Future Work

Several opportunities exist to extend this work:

1. Multi-modal Retrieval: Expanding the system to retrieve from images, PDFs, or audio could make it more versatile.
2. Reinforcement Learning Feedback Loops: Using human or automated feedback to fine-tune generation quality.
3. Agent Capabilities: Enhancing the system's autonomy with API-calling abilities, dynamic tools, or scheduling behaviours.
4. Voice Interface Integration: Building a speech-based layer could enable use in assistive technologies or embedded devices.
5. Scalability and Benchmarking: Evaluating the architecture under large-scale data or production load conditions.

III. CONCLUSIONS

1. Enhanced Accuracy Through Retrieval

The integration of retrieval mechanisms significantly improved the accuracy and relevance of the generative model's responses by grounding them in real-time, domain-specific information.

2. Effectiveness of Local LLMs via Ollama

Running large language models locally using Ollama proved to be both efficient and secure, ensuring data privacy while maintaining low operational costs.

3. Modularity Facilitates Customization

The decoupled architecture of the system allows for easy experimentation with various embedding models, vector databases, and language models, making the solution highly adaptable across multiple use cases.

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

4. Semantic Search Delivers Contextual Precision

The use of vector databases like FAISS/Chroma and embedding models enabled effective semantic search, ensuring that the retrieved context was both relevant and meaningful.

REFERENCES

[1] Johnson, J., Douze, M., and Jégou, H., 2021, "Billion-Scale Similarity Search with GPUs," IEEE Trans. Big Data, 7(3), pp. 535–547.

[2] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., et al., 2020, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," Advances in Neural Information Processing Systems, 33, pp. 9459–9474.

[3] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," IEEE Transactions on Big Data, vol. 7, no. 3, pp. 535–547, 2021.

[4] P. Lewis, E. Perez, A. Piktus, et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," Advances in Neural Information Processing Systems, vol. 33, pp. 9459–9474, 2020

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY RESEARCH

IN SCIENCE, ENGINEERING AND TECHNOLOGY

| Mobile No: +91-6381907438 | Whatsapp: +91-6381907438 | ijmrset@gmail.com |

www.ijmrset.com