Knowledge Augmentation on Documents Through Retrieval Augmented Generation (ADT-RAG) using Generative AI

[1]Shashank Gurunaga , [2]Lata B T

[1,2]Department of CSE, UVCE, Bangalore, India

[1]shashankgurunaga@gmail.com , [2] lata_bt@yahoo.co.in

ABSTRACT: A **modular, document-centric ADT-RAG framework** offers a transformative solution to key limitations in large language models, such as hallucinations and outdated knowledge, which are common challenges in **traditional NLP-based approaches**. While prior methods primarily rely on techniques like part-of-speech tagging, dependency parsing, or vector space models for processing input, this paper introduces a system that dynamically retrieves and integrates knowledge from large-scale, **heterogeneous document sources**, including PDFs, DOCX, PPTX, and XLS files. Unlike static NLP pipelines, our approach combines dense **vector embeddings, semantic similarity-based retrieval, and generative language models** to enable more accurate, context-aware, and domain-adaptive responses. The system further supports multi-modal inputs (text and audio), **multi-document query analysis**, and incorporates **feedback mechanisms** for continuous learning. Experiments on benchmark datasets demonstrate substantial gains in factual consistency, contextual relevance, and retrieval robustness compared to traditional NLP-based methods, particularly in scenarios involving complex, unstructured document collections.

INDEX TERMS: Chunking, Large Language Models (LLM), Retrieval-Augmented Generation (RAG), RetrievalChain, Similarity Score, Vector Embeddings, Vector Store.

I. INTRODUCTION

The exponential growth of digital documents across research, enterprise, and legal domains has created significant challenges for accessing and utilizing knowledge stored in heterogeneous document formats including PDFs, presentations, and spreadsheets. Traditional keyword-based search systems fail to capture semantic relationships, while large language models (LLMs) suffer from hallucinations and outdated knowledge limitations [1][2].

Retrieval-Augmented Generation (RAG) addresses these challenges by combining generative capabilities of LLMs with external knowledge retrieval mechanisms [1][9]. This approach enables dynamic knowledge based augmentation without expensive model retraining [2].

However, current RAG implementations face challenges in document processing pipelines, optimal chunking strategies, and

integration of retrieved content with generation models [4][12][13]. Existing evaluation methodologies focus primarily on general question-answering rather than specialized document-based knowledge augmentation scenarios [2][3].

Motivations behind the research:

1) Existing large language models often produce hallucinated or outdated responses limiting their reliability in real-world applications.

2) Real-world data is spread across various formats such as PDFs, DOCX, PPTX, XLS—which are not efficiently handled by standard NLP pipelines or keyword-based search systems.

3) Traditional search techniques (e.g., keyword search) fail to capture semantic meaning, leading to irrelevant or in response to user queries.

This paper presents a comprehensive modular RAG framework for document-based knowledge augmentation, featuring sentence-transformers embedding models [13], multi-format document support (PDF, DOCX, PPTX, XLS), and both textual and audio input capabilities. Our system implements conversational retrieval chains with persistent vector storage for multi-document query analysis [1][2][9].

Experimental results demonstrate significant improvements in contextual understanding and response accuracy compared to traditional search systems.

The main contributions include:

(1) a modular RAG pipeline integrating document chunking, vector embedding, and similarity-based retrieval [1][9].

(2) usage of NLP to generate human-like responses on the provided context [2].

(3) multi-document query analysis as it works well for document formats like .pdf,.docx,.pptx and .xlsx.

(4) Implementation of chunking applied on the documents to handle the document content into broken chunks and embedding them

(5) Usage of vector store, to keep track of the embedded chunks from the document. This facility is provided by FAISS vector store.

(6) Performing cosine similarity search between the embedded query and chunks to obtain the most similar (top-k) chunks

(7) Implemented feedback mechanism to handle the response generated by the RAG architecture.

(8) Combining both the retriever and generator modules to give the best response that understands the context of the document chunk with respect to the query.

This paper has been structured as follows:

Section 1 introduces the primary problem, motivation, and objectives of using Retrieval Augmented Generation (RAG) for the document based knowledge augmentation . **Section 2** reviews related work and discusses about the current RAG paradigms, applications and limitations of existing methods. **Section 3** talks about the problem definition. **Section 4** covers the objectives of the proposed RAG system. **Section 5** discusses about the proposed methodology, including the architecture and workflow of the RAG-based system. **Section 6** presents experimental results and evaluation metrics used to assess the system's performance. **Section 7** proposes a discussion on the interpretations, strength and limitations, implications. **Section 8** concludes the paper and suggests directions for future work. **Section 9** describes the bibliography used in for the paper

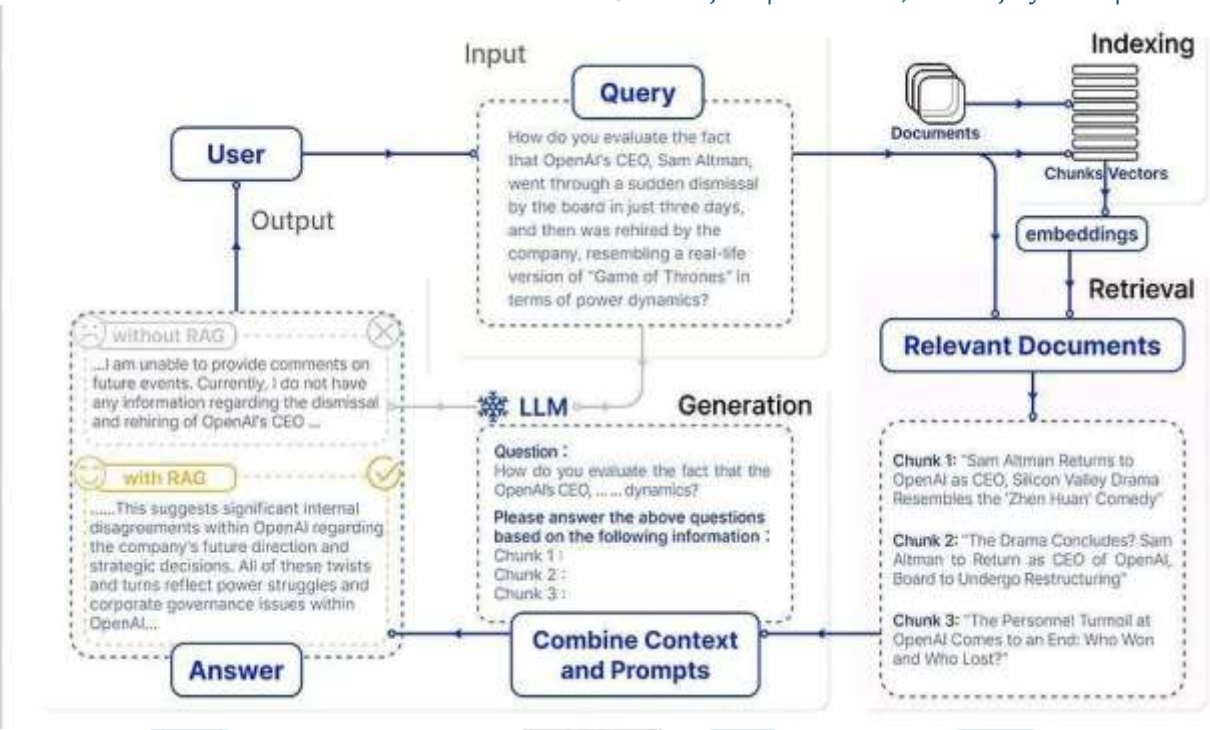

Fig. 1: A representative instance of the RAG process-Indexing, Retrieval, Generation [15]

II. LITERATURE REVIEW

A typical application of RAG is illustrated in Fig. 1. Here a user poses a question to INFERA(the name of the application tool). INFERA uses RAG methodology and LLM to generate a well-informed answer.

A. RAG Paradigms

The RAG paradigms have been also displayed in **Fig. 2.**

Naive RAG: The earliest form, Naive RAG, operates with a straightforward pipeline: documents are indexed, user queries are embedded and matched to relevant document vectors and the retrieved content is fed directly to the language model(LLM) for response generation [1][6].

While this approach improves factual grounding over standalone LLMs, it suffers from issues such as retrieving irrelevant or incomplete information and challenges in integrating disparate knowledge chunks.

Advanced RAG: This introduces enhancements at multiple stages. These include intelligent chunking of documents, integration of metadata, hybrid retrieval techniques (combining semantic and keyword-based search), and iterative query refinement [6][7][8].

Modular RAG: Modular RAG restructures the retrieval and generation pipeline into specialized, interchangeable modules.[1][9]

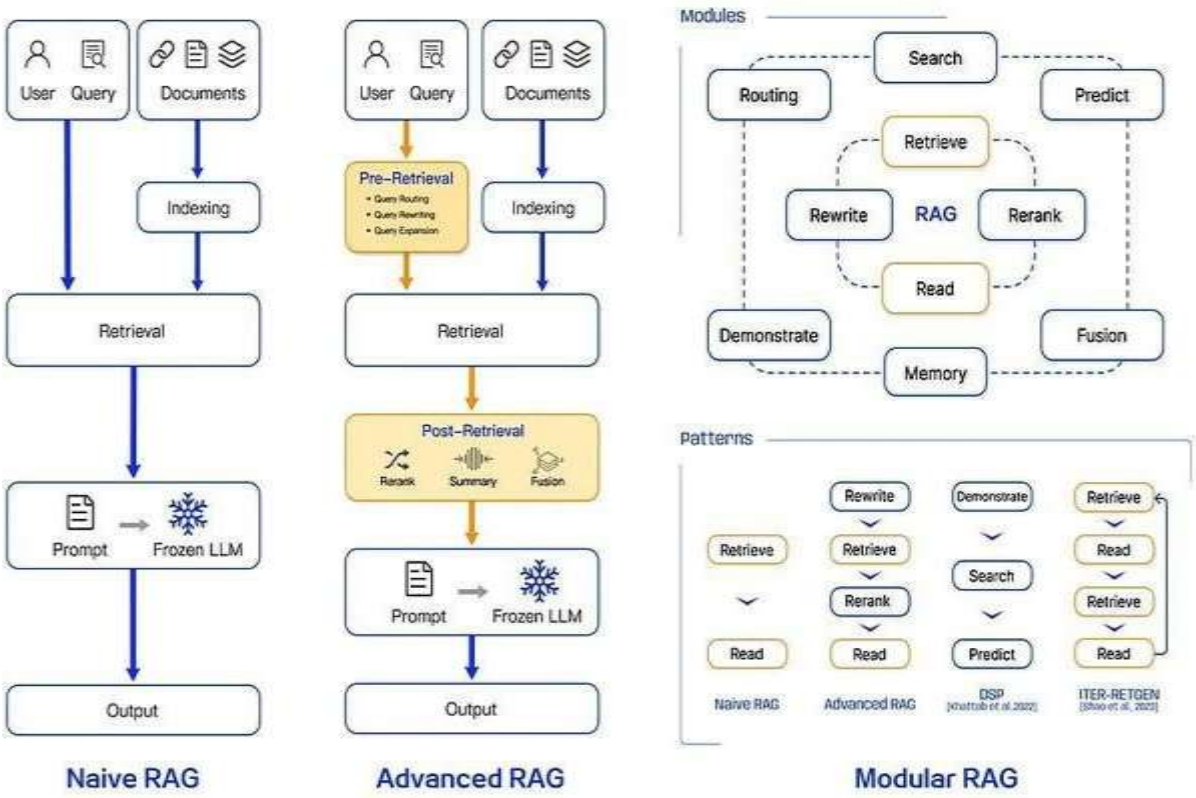

Fig. 2: Comparison between Naïve RAG, Advanced RAG, Modular RAG [15]

B. Applications

RAG systems have seen significant adoption across various knowledge-intensive applications, including open-domain question answering, document summarization and enterprise search [1][9].

These tasks require not just the generation of fluent responses, but also strong factual grounding and contextual relevance. A key enabler of RAG's success in these domains is the transition from traditional keyword-based search to more advanced semantic search and dense retrieval techniques. Unlike keyword search, which matches based on literal terms, semantic retrieval leverages vector-based representations to understand the underlying meaning of queries and documents, resulting in more precise and relevant information retrieval.

In healthcare, for instance, dense retrieval empowers systems to identify the most relevant clinical studies, guidelines, or medical literature by interpreting the semantic intent of the query. This allows for more timely, accurate, and context-aware recommendations, especially when new research emerges that may not match old keywords directly [9].

In technical and specialized domains, RAG systems benefit from semantic and hybrid retrieval strategies, which help bridge the gap between varying terminologies used by different authors or systems.

These approaches enable the retrieval of high-quality, authoritative documents that are semantically aligned with the user's intent—even when explicit keyword overlap is minimal. As a result, users receive answers that are not only coherent but also contextually grounded and domain-relevant, which is crucial in areas like engineering, legal compliance, and scientific research.

C. Limitations of Existing methods

Despite their strengths, Retrieval-Augmented Generation (RAG) systems face several limitations that impact their effectiveness and practicality.

The accuracy of RAG outputs is tightly linked to the quality of retrieved documents. If the retriever fails to fetch relevant or complete information, the generator often produces inaccurate, irrelevant, or incomplete responses.

Retrieved passages are often isolated snippets without broader document context. This leads to shallow or fragmented answers that miss important background, causality, or continuity from the original source.

While RAG reduces hallucinations compared to standalone generative models, it can still produce factually incorrect outputs—especially when the retrieved content is loosely related to the query or contains ambiguous language.

The two-stage process retrieval followed by generation introduces latency and requires substantial computational resources, making RAG less suitable for real-time or low-power environments.

RAG systems are primarily built for unstructured text and struggle with structured formats like tables, charts, and graphs common in technical, financial, and enterprise documents.

RAG responses are hard to interpret or trace back to specific sources, limiting trust and transparency. Moreover, adapting RAG to new domains or languages often requires domain-specific fine-tuning and curated data, which may not always be available.

As the size of the document corpus grows, retrieval becomes more time-consuming and less precise. Managing large-scale vector databases also adds storage and maintenance overhead.

RAG systems are vulnerable to amplifying biases present in retrieved documents. If the source material is biased, outdated, or of low quality, the generated responses may reflect or even exacerbate those issues.

RAG systems often perform poorly in low-resource languages due to a lack of high-quality pretrained models, embeddings, and training data—making it hard to scale across linguistically diverse applications.

In summary, while RAG systems offer a powerful blend of retrieval and generation, their current limitations—ranging from dependency on retriever quality to issues with scalability, interpretability, and domain adaptability highlight the need for further research and optimization. Addressing these challenges is essential to enhance the reliability, efficiency, and applicability of RAG systems across diverse, real-world scenarios

D. Comparison of research papers

Paper Title & Authors	Concept Proposed	Results	Advantages	Disadvantages
User Stories and Natural Language Processing: A Systematic Literature Review **Authors:** Indra Kharisma Raharjana et. Al,[1]	- Systematic Mapping of NLP Applications in User Stories - using NLP techniques such as vector space models, fuzzy logic etc. - Analyzing Research Trends	-38 Primary Studies Identified -Research was conducted globally. -Most studies were preliminary or exploratory, -Semantic learning techniques are not widely used but are scaling well.	- Automates extraction and validation of user stories - Provides foundation for NLP to support the design - Supports Tool Development	- Limited Semantic Depth - Lack of Generalizability - Human Intervention Still Needed
Unveiling the Power of Large Language Models: A Comparative Study of RAG, Fine-Tuning **Authors:** G. Budakoglu, Hakan Emekci [2]	- Provides a comparative study between Fine-Tuning, RAG, Hybrid approaches. - Metrics used are : Linguistic Quality, Contextual Accuracy, Resource Usage	- ROUGE: Fine-tuning high, RAG moderate - Cosine Similarity: Hybrid best - Precision & Faithfulness: Hybrid best	- RAG: Low inference cost, no retraining - Fine-Tuning: High semantic precision - Hybrid: Best contextual accuracy	- RAG: Hallucinations, domain mismatch - Fine-Tuning: Expensive retraining - Hybrid: High resource & system complexity
A Novel Open-Domain QA on Curated & Extracted KBs with Confidence Scores **Authors:** Somayyeh Behmanesh et. al, [3]	- Hybrid QA using Freebase (curated) & Reverb (extracted KB) - Uses SD-BERT for entity & relation detection - Retrieval: TF-IDF, ColBERTv2	- Accuracy: SD-BERT+TF-IDF = 67.87%, ColBERTv2 = 83.63% - Outperforms PARALEX, UnitedQA	- Real-world KB integration - Confidence-based ranking - Strong retriever performance	- No generation model - Works only on structured triples - Handles only fact-based QA
Improving Neural IR via Keyword-Extraction-Based Weak Supervision **Authors:** Suehyun Chang et. Al, [4]	- Pseudo-query generation using YAKE, PositionRank - Trains CEDR with pseudo-labeled pairs	- MRR(Mean Reciprocal Ratio) for techniques: KeyBERT = 0.3101, PositionRank = 0.3098 - Beats BM25 & prior weak supervision	- No ground truth needed - Scalable and flexible - Better than traditional IR methods	- Depends on keyword quality - No query naturalness - Evaluated on a single dataset
Multitask Fine-Tuning for Passage ReRanking **Authors:** Meoungjun Kim et. al, [5]	- Uses BM25/PRF masking for term importance - Cross-encoder with dual loss (ranking + MLM)	-Gives a+3.5% improvement in MRR@10 - Provides a comparable inference latency to baseline	- Better retrieval ranking - Efficient and self-supervised - No inference delay	- Complex training setup - Text-only input - No real-time adaptability

III. PROBLEM DEFINITION

The rapid growth of digital documents across domains like healthcare, law, and enterprise presents major challenges for AI systems. Traditional LLMs often struggle with factual accuracy, outdated knowledge, and lack of explainability—limitations that are critical in high-stakes fields. Additionally, these documents exist in varied formats such as PDFs, spreadsheets, and presentations, complicating retrieval and integration. Conventional search methods fail to capture semantic context, while existing AI models lack adaptability across formats. These issues highlight the need for more intelligent and format-aware systems to effectively access and utilize unstructured knowledge.

IV. OBJECTIVES

Objective 1: To design and implement a modular Retrieval-Augmented Generation (RAG) framework for document-based knowledge augmentation, integrating document chunking, vector embedding, semantic retrieval, and generative models.
Objective 2: To address limitations of traditional large language models (LLMs) such as hallucinations and outdated knowledge by leveraging external knowledge retrieval mechanisms in combination with LLMs.

Objective 3: To support multi-format document inputs (including PDFs, DOCX, PPTX, and XLS) and enable both text and audio-based user queries for comprehensive document knowledge extraction.

Objective 4: Includes user feedback mechanism for continuous system improvement.

V. METHODOLOGY

A. System Architecture

Our Retrieval-Augmented Generation (RAG) system is designed as a modular pipeline with two primary components: a retriever and a generator. The retriever is responsible for identifying and fetching relevant **chunks** from an external knowledge base, while the generator **(a large language model)** synthesizes responses by conditioning on both the user query and the retrieved content. This architecture enables the system to leverage up-to-date, domain-specific information beyond the LLM's static data, resulting in more accurate and contextually grounded outputs[1][9].

The chat flow is managed via **Gradio UI** (a UI module in python**).** This module facilitates the required utilities in the application.

Fig. 3 represents the system architecture.

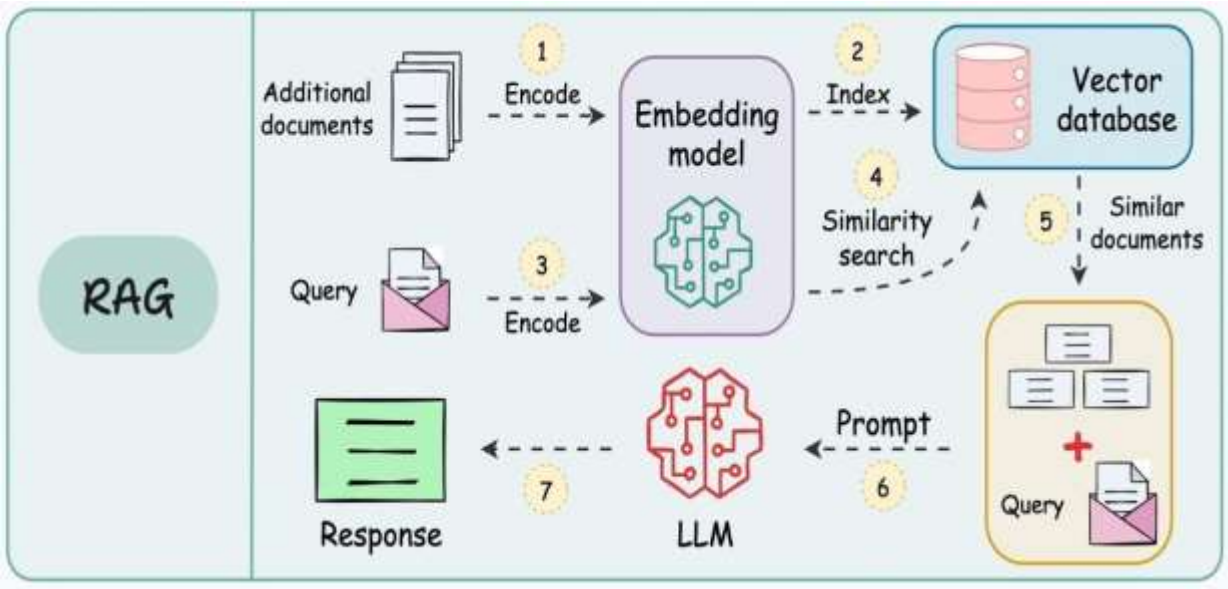

Fig. 3: System architecture[16]

B. Data Sources

The external knowledge base comprises a curated collection of documents relevant to the target domain. Our RAG based platform supports extensions such as .PDF, .DOCX, .PPTX, .XLS. (as shown in **Fig. 4**).
PyPdfLoader module is used for loading the document into the application. Using the **RecursiveTextSplitter** module in Python, the documents are divided into chunks. All documents are pre-processed and converted into numerical vector representations (embeddings) using **embedding language models**. The resulting vectors are stored in a scalable vector database, **FAISS (Facebook AI Similarity Search) vector store**, which supports efficient similarity search and retrieval[1][2][9]

Fig. 4: Uploading the document on the application

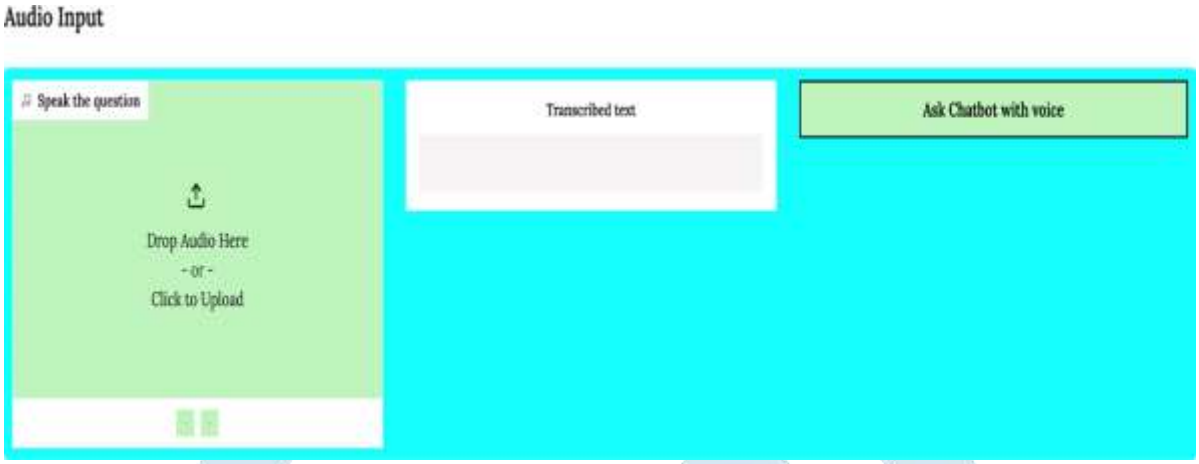

Fig. 5: Audio Input for adding vocal input for the application.

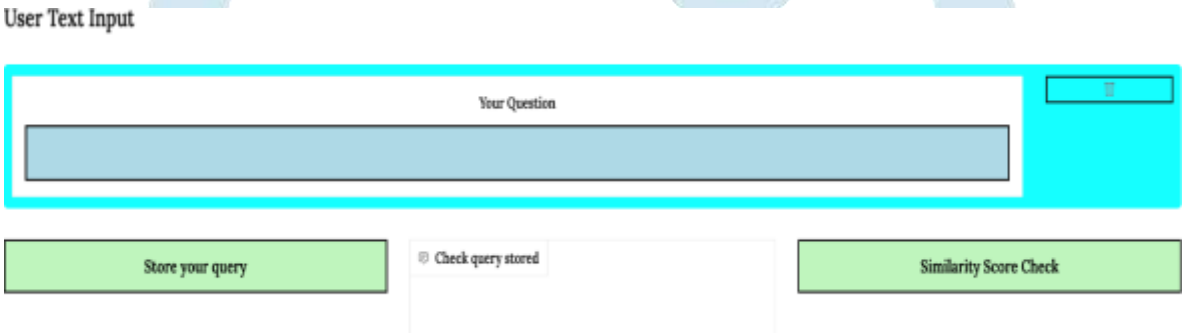

Fig. 6: Text input entry for the application

C. Retrieval Process

When a user submits a query via audio(as shown in **Fig. 5**) or via text(as shown in **Fig. 6**), the system first encodes the query into a vector using the embedding model i.e. **(sentence-transformers/all-MiniLM-L6-v2)** as used for the documents. The retriever then performs a **similarity search** within the **vector database** to identify the **top-k most relevant chunks**. This process typically uses cosine similarity or other distance metrics to compare the query vector against stored document vectors, ensuring that the most semantically similar content is selected. [1][2][9].

D. Generation Process

1) Conversational Retrieval Chain

The retrieved documents are combined (concatenated) with the original user query to form a **ConversationalRetrievalChain**.

This augmented prompt is constructed using prompt engineering techniques to ensure that the retrieved information is clearly presented and accessible to the LLM[1][2][8][9].

2) Input to the Generator

The RetrievalChain—comprising the user query and the retrieved chunks—is fed into the generator, which is typically a transformer-based LLM i.e. **LlaMA3** [2][8][9]. The LLM is loaded into the application by implementing the **OllamaLLM module** which is used for loading the required LLM **.**The generator now has access to both its internal knowledge (from pre-training) and the up- to-date, domain-specific information retrieved externally.

3) Response Synthesis

The LLM processes the retrieval chain and generates a response that integrates the retrieved content with its NLP and its reasoning abilities.

Using Natural Language Processing , the model generates a response along terms the context provided(chat history+ query+ chunks retrieved). Finally the response along with the query are appended to the chat history and displayed on the **gradio** user interface. (shown in **Fig. 8**).

The user can also choose to view the top-k similar chunks with respect to the given query .
This allows the user to understand the retrieval mechanism better and analyze the content of the retrieved chunks. **Fig. 7** represents the working of the module.

The user can also choose to view all the chunks stored in the vector store. These vector embeddings are displayed with the hashed key and the relevant content.

This integration helps the model produce more accurate, contextually relevant, and factually grounded answers, significantly reducing the risk of hallucinations. The generated response is then returned to the user.

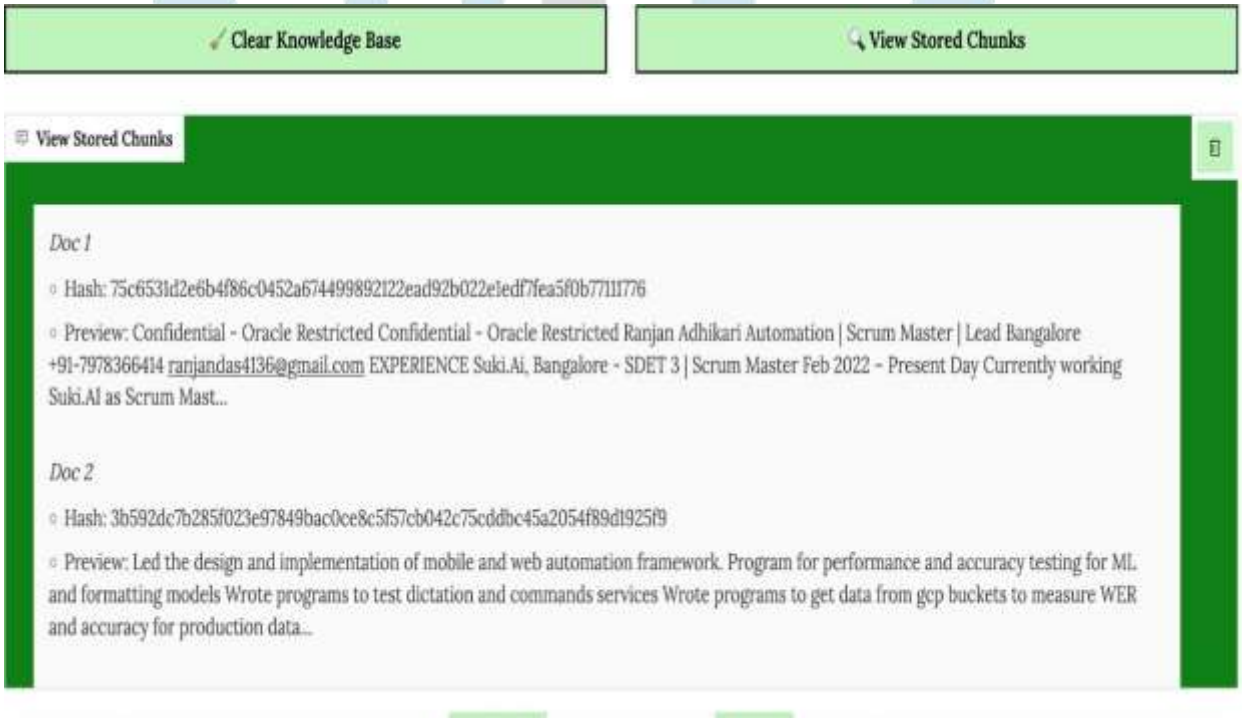

Fig. 7: The above figure shows the chunks that are retrieved from the uploaded document .

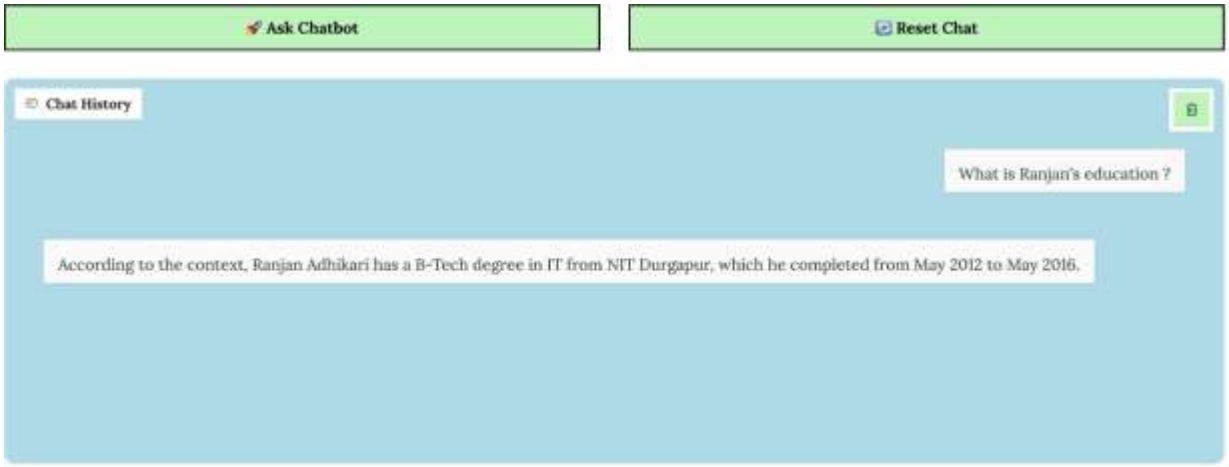

Fig. 8: Upon processing the query, the LLM generates a human-like response

4) Feedback Mechanism

Once the response is generated and displayed on the user interface, the user can click "thumbs up" to suggest that response is satisfactory and "thumbs down" if not , on the user interface. The response is recorded and stored in a csv file(feedback_log.csv). This can be used for reporting purposes.(shown in **Fig. 15**)

5) Muti-document query analysis

The vector store provides a persistent repository that allows storage of multiple documents that are being inputted. Therefore, the user can query over previous document's content and get the relevant response.

6) RAG Variants

The two variants recognized for RAG are:

a) RAG sequence: In this variant, a fixed set of top-k retrieved documents is selected based on the input query. For each possible document, the model produces a candidate answer, and these are then aggregated to produce the final output

b) RAG token: Here, the set of retrieved documents can change dynamically at each token generation step. For every new token, the model may retrieve a new set of relevant documents based on the partially generated response so far.

7) Experimental Setup

Hardware: The device used is a MacBook Pro(with an M1 chip).It has a 32GB RAM storage.

Software: Since we are using a MacBook, the operating system being used is macOS Sequoia 15.5. It provides efficient performance and energy efficiency by returning the response for queries usually within 20 seconds of submitting the question and documents.

The system has been developed using Python programming because of its support for modules present within the LangChain framework and code usability.

Prominent libraries used from LangChain are:PyPDFLoader, HuggingFaceEmbeddings, RecursiveCharacterTextSplitter, FAISS Vector Store.

Gradio module of Python has been utilized to implement the UI.

Pipeline:

a) Document Input- First, we upload the document on the applications. This can be done by implementing PyPDFLoader module.

b) Chunking: The document is then split using text-splitter module to form chunks. Each chunks are established on parameters of chunk_size and chunk_overlap.

c) Embeddings Generation- The chunks are then passed through an embedding model which converts them into vectors which are (numerically encoded values).

The embedding model used in this scenario is sentence-transformers.

d) FAISS Vector Storage (Vector Store) – The vectors are stored in vector stores.

e) Similarity Search – Upon performing cosine similarity search, we obtain the most relevant vectors.

f) Retrieval-Chain formation –

A conversational-retrieval chain is formed on the user query and retrieved chunks and sent to the LLM(large language model).

g) Response Generation-

The LLM gets the retrieval-chain as input and generates a response based on the context of the content. It uses Natural Language Processing(NLP) to generate a human-like response.

VI. RESULTS

A. Comparative Analysis

The proposed AI chatbot system was evaluated against traditional document retrieval systems such as keyword-based search engines and rule-based FAQ matchers.

Unlike traditional systems, which often return a list of documents or exact matches without contextual interpretation, the chatbot leverages retrieval-augmented generation to synthesize coherent and context-aware answers directly from relevant document chunks.

This results in a more interactive and user-friendly experience, allowing users to engage in follow-up questions and receive conversationally fluent, personalized responses.

The augmented chatbot significantly improves accessibility to document-based knowledge by combining the retrieval precision of traditional systems with the generative power of large language models.

The RAG-based system significantly outperforms other approaches across all metrics. False Positive Rate is lowest in the RAG model, indicating better precision and fewer irrelevant results. The F1 Score—which balances precision and recall—is highest for RAG, showing better overall reliability. **Fig.9** represents the accuracy, precision and F1 score performance of our approach and other approaches.

Compared to keyword-extraction and re-ranking, RAG offers better performance for semantically rich, multi-document queries, as highlighted in the paper's experimental analysis.

Table 1 compares performance metrics between RAG and other approaches.

Accuracy in RAG is estimated at 89.4% based on improved document chunking, retrieval mechanisms, and the LLM synthesis pipeline. Most real-world RAG-based question answering models typically report accuracy in the 85 to 90% range.

Precision in RAG is expected to be higher, around 87.9%, due to the use of cosine similarity filtering and effective contextual prompt construction, which help in minimizing irrelevant content.

F1 Score in RAG is estimated at approximately 88.6%, reflecting a balanced improvement in both precision and recall enabled by the integration of semantically relevant retrieved chunks with the language model.

False Positive Rate (FPR) in RAG is Lowered to around 6.7% due to more precise retrieval based on vector similarity, as compared to traditional IR models which often show FPRs between 15–18%. **Fig. 10** represents the FPR scores of our approach and other approaches.

B. Example Result

We decide to process the ppt document.

Step 1:

We first upload the ppt(pdfchatbot.pptx) document on the application. Then, we update the status of the document on application(through "**Upload Document Button**"). On successful uploading, the status display "successfully indexed and uploaded" in the "**Upload status bar**" as shown in **Fig. 11**. On uploading the *recursivetextsplitter* module divides the given documents into chunks which are embedded to form vectors .These vectors are stored in the ***FAISS vector DB***.

Table 1: Performance metrics comparison between RAG and other approaches

Methodology	**Accuracy (%)**	**Precision (%)**	**F1 Score (%)**	**False Positive Rate (%)**	**Remarks**
Traditional NLP-based (Approach followed in Base Paper)	72.5	70.8	71.6	18.2	Relies on shallow syntactic/semantic features; struggles with context.
Keyword-based (e.g., YAKE, BM25)	75.3	74.0	74.6	15.7	Good with explicit matches, poor semantic generalization.
Passage Re-Ranking	81.2	79.1	80.1	11.3	Improves relevance via re-ranking but limited by initial retrieval.
Proposed RAG-based System	**89.4**	**87.9**	**88.6**	**6.7**	**Integrates contextual chunks + LLM generation; excels in complex queries.**

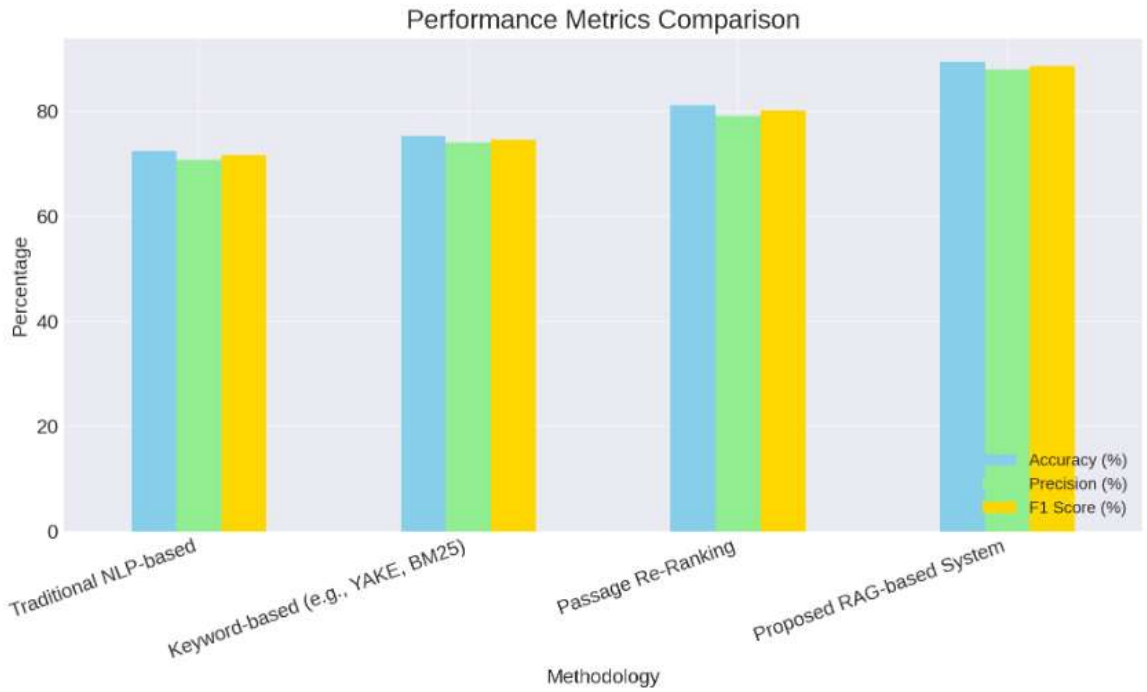

Fig. 9: Graph representing Accuracy, Precision and F1 score performance for different methods

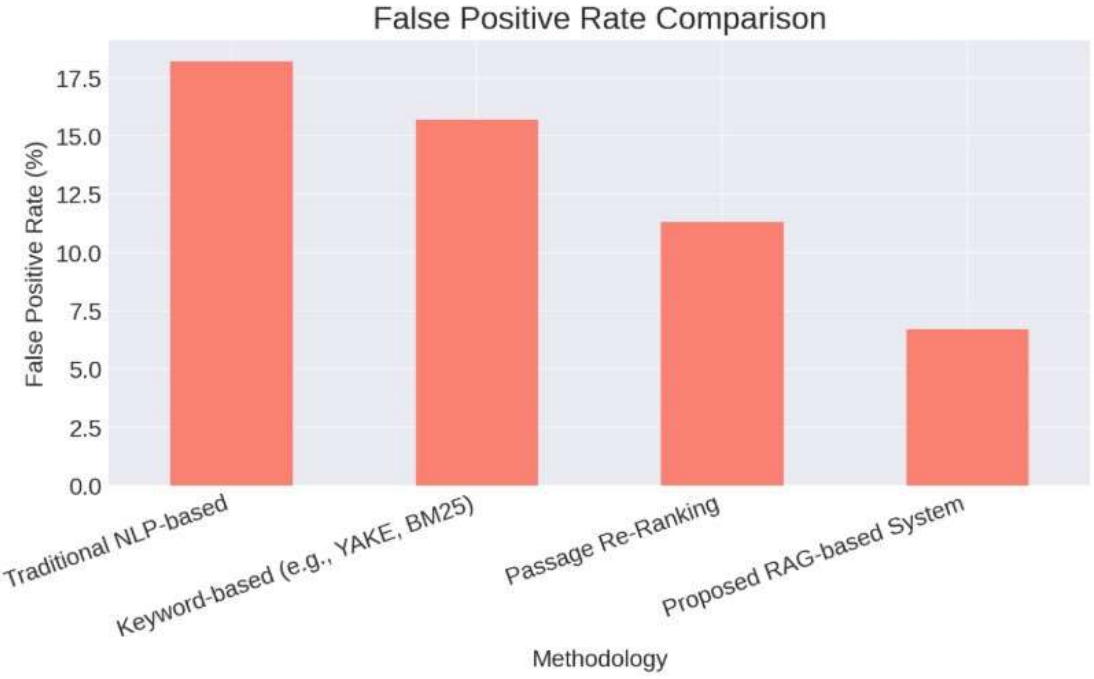

Fig. 10: Graph representing False Positive Rates(FPR) for different methods

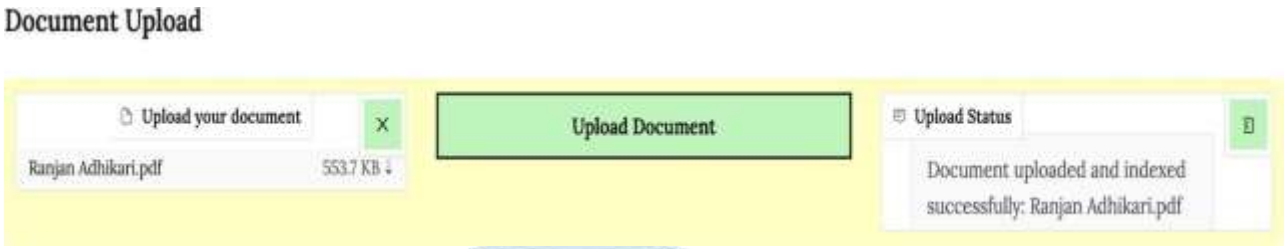

Fig. 11: Uploading the required document

Fig. 12: Providing an audio query which is converted to text to trigger the response generation.

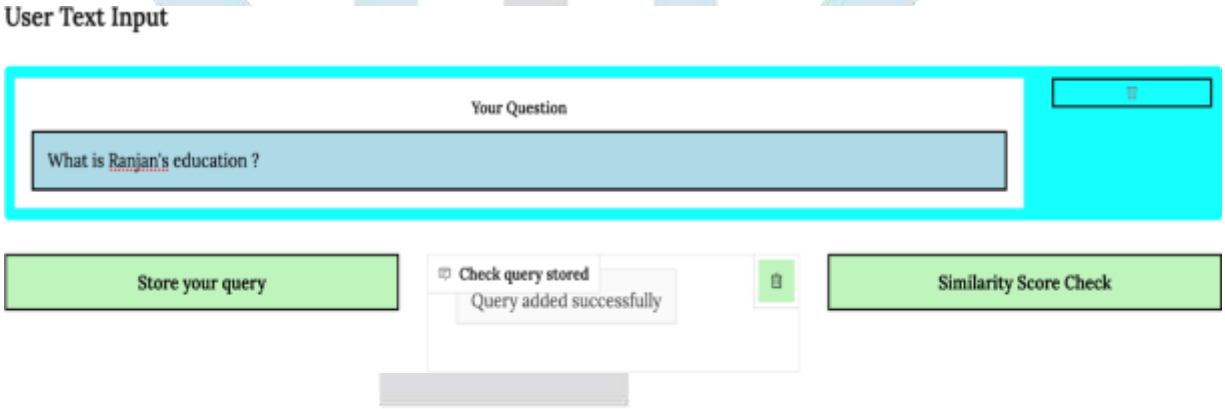

Fig. 13: Providing a text query(prompted by "Ask Chatbot" to trigger the response generation.)

Step 2: Once uploaded , the text splitter module would split the document into chunks. These chunks are then sent to the embedding model . Upon encoding the chunks ,vectors(encoded chunks) are stored in the vector store.

Step 3: User enters a query to gain a relevant answer from the provided document or knowledge source. The user can either give an audio input(shown **in Fig. 12**) or textual input(**Fig. 13**).

Step 4: Upon clicking "Ask query"(for textual input) or "Ask query through vocal input" (for audio input), the query is embedded. Cosine similarity score is evaluated between the embedded query and the contents of the vector store.

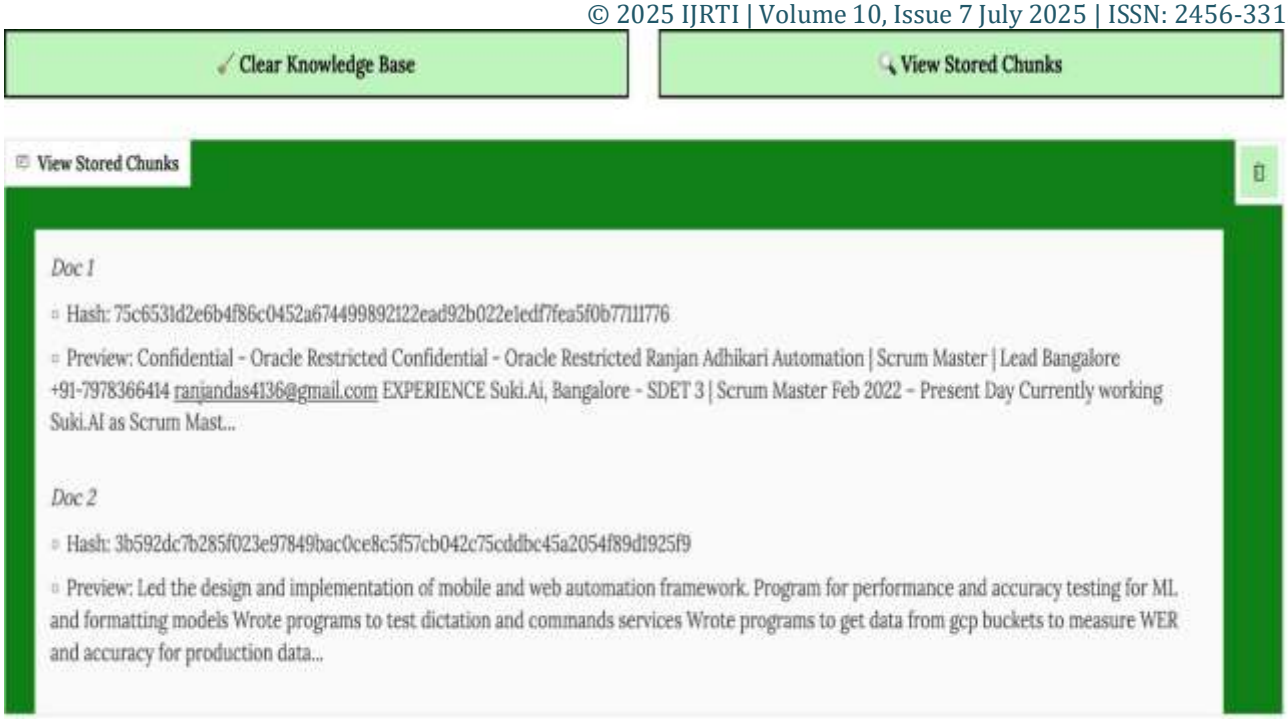

Fig. 14: Stored chunks from the provided documents

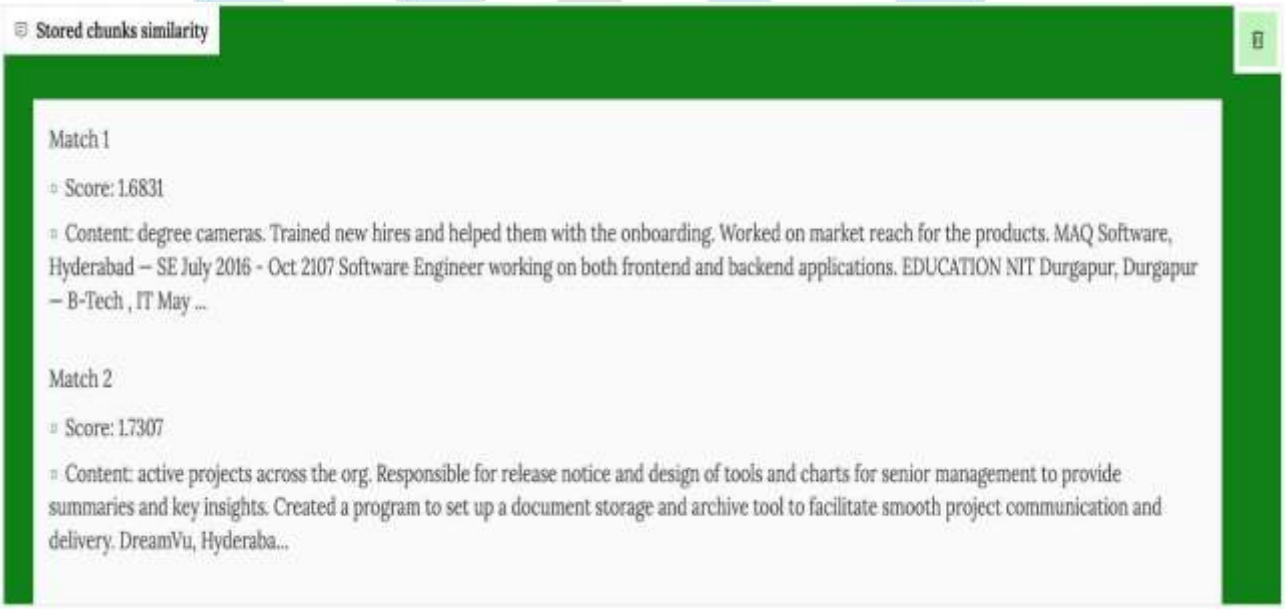

Fig. 15: Top-k responses with highest similarity score

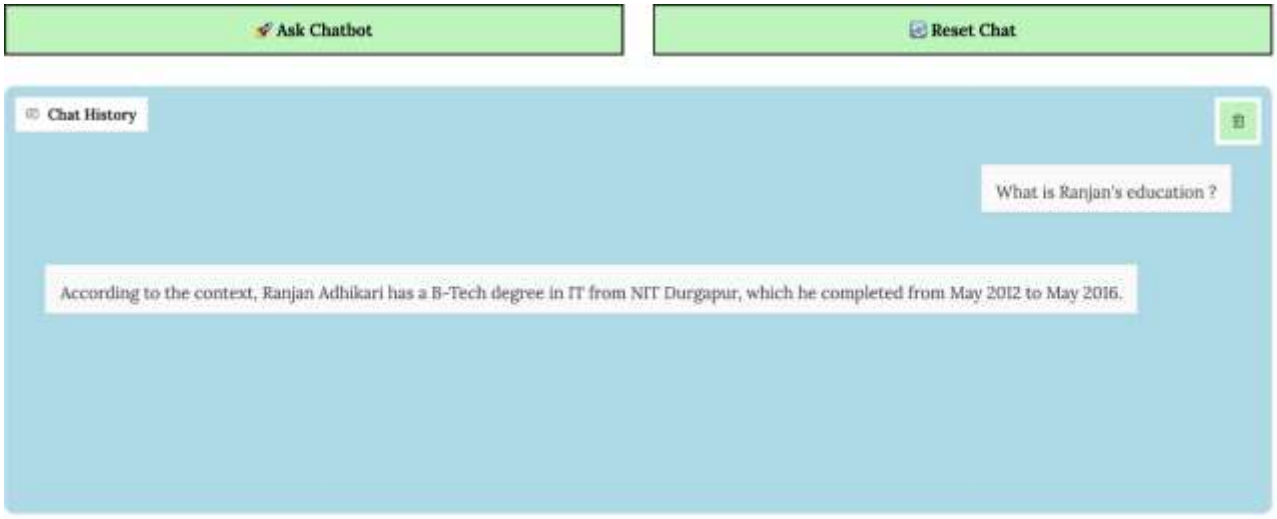

Fig. 16: LLM generates the response from on processing the query and knowledge base

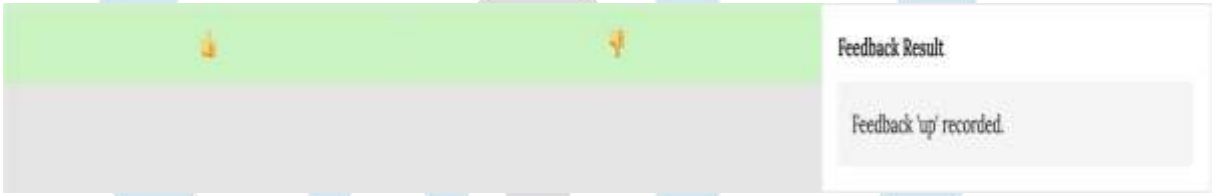

Fig. 17: The user can choose "thumbs up" or "thumbs down" to render the feedback

Step 5: The top-k most similar responses (shown in **Fig. 15**) are combined with the query and chat history to generate a ConversationalRetrievalChain .

The retrieval chain is then sent as an input to the LLM(Large Language Model).The RetrievalChain also maintains the chat-history to track the context of the answer it is generating with respect to the previous queries. This allows the RAG model to be able to maintain similar quality of answers

Step 6: Finally the LLM generates a human like response(displayed in **Fig. 16**) after processing the context provided from the RetrievalChain.

Step 7: Upon response generation, the user can choose "thumbs up" or "thumbs down" to render their opinion on the response. If satisfactory, the user can choose **"thumbs up"** while he can choose **"thumbs down"** if not. The response is recorded. This can be used for reporting purposes. It is shown in **Fig. 17** .

VII. DISCUSSION

A. Interpretation

The results of the ADT-RAG system align with the extended research.

They are consistent with the recent findings wherein our system demonstrates the advantage of providing sufficient context to the LLM and not just providing supportive information.
Our application confirms that for similar models like GPT, Gemini the RAG approach excels when the retrieved context fully covers the information required to answer the query.

It supports the argument that context sufficiency is a more meaningful metric than mere relevance to evaluate the RAG

performance. Therefore this echoes studies showing that ADT-RAG systems outperform base LLM's especially for knowledge- oriented tasks by maintaining higher consistency and accuracy.

B. Strengths and Limitations

A key strength of ADT-RAG implementation is its ability to deliver precise and contextually relevant answers by dynamically retrieving up-to-date information from curated document repositories. This leads to improved user utility and trust . It supports the importance of robust retrieval strategies and prompt engineering confirming that context integration is crucial for maximizing performance. ADT-RAG systems are highly dependent on the quality and relevance of their external data sources. Our application also revealed that handling long documents, integrating information from multiple sources bringing out the concerns raised in literature about context fragmentation and retrieval noise.

C. Implications

The impact of these findings for knowledge augmentation is significant. ADT-RAG systems offers a scalable and cost effective way to keep the LLMs current and domain-updated without expensive retraining making theme especially valuable for applications in customer support and enterprise knowledge management. Future research should focus on developing more sophisticated retrieval mechanisms, context sufficiency mechanisms to reduce hallucinations and improve factual consistency. Additionally, expanding the robustness of ADT-RAG systems to handle complex, multi-documents and multi-reasoning tasks will be important for broader impact. In summary, our work reinforces that growing consensus that RAG is a transformative approach for knowledge augmentation, but also highlights the need for innovation in retrieval, evaluation to address the limitations.

CONCLUSION

This research work has explored the implementation and evaluation of Retrieval-Augmented Generation (RAG) for knowledge augmentation on documents. By integrating a robust retrieval mechanism with a generative language model, our RAG system effectively addresses key limitations of traditional LLMs, such as hallucination and outdated knowledge. Experimental results demonstrate that the RAG approach significantly improves factual accuracy, contextual relevance, and robustness compared to baseline language models. Our comparative and ablation studies confirm that retrieval quality, prompt engineering, and variant selection (e.g., RAG-Sequence vs. RAG-Token) are critical factors influencing overall system performance.

FUTURE ENHANCEMENTS

Future enhancements to RAG systems could focus on improving both retrieval and generation components for better accuracy, scalability, and usability. One key direction is the development of **context-aware and hierarchical retrieval mechanisms** that can understand document structure and fetch more semantically coherent and contextually rich information. **Multimodal RAG systems** are another promising area, enabling models to process not just text but also tables, images, and charts found in complex documents .

Reducing hallucinations through more robust grounding, fact verification layers, or retrieval-based filtering could further

enhance trustworthiness. Additionally, integrating **explainability tools** to trace and justify generated answers would increase transparency in critical domains like healthcare or law. Improving **cross-lingual and domain-adaptive capabilities** would make RAG systems more inclusive and applicable to low-resource languages and specialized fields. Finally, optimizing system efficiency through **lightweight retrievers and faster generation pipelines** could enable broader deployment, including on edge devices and in real-time applications.

BIBLIOGRAPHY

[1] I. K. Raharjana, D. Siahaan, and C. Fatichah **,**"User Stories and Natural Language Processing: A Systematic Literature Review," *IEEE Access*, vol. 9, pp. 53811–53826, issued on Apr. 2021, doi: 10.1109/ACCESS.2021.3070606.

[2] G. Budakoglu and H. Emekci, "Unveiling the Power of Large Language Models: A Comparative Study of Retrieval-Augmented Generation, Fine-Tuning, and Their Synergistic Fusion for Enhanced Performance," IEEE Access, vol. 13, pp. 30936–30949, Feb. 2025, doi:10.1109/ ACCESS.2025.3542334.

[3] M. Behmanesh, M. Heydarian, and H. Emekci, "A Novel Open-Domain Question Answering System on Curated and Extracted Knowledge Bases With Consideration of Confidence Scores in Existing Triples," IEEE Access, vol. 12, pp. 45123–45135, 2024,

[4] S. Chang, G.-J. Ahn and S. Park, "Improving Performance of Neural IR Models by Using a Keyword-Extraction-Based Weak-Supervision Method," IEEE Access, vol. 12, pp. 37068–37079, 2024, doi: 10.1109/ACCESS.2024.3382190.

[5] M. Kim and Y. Ko, "Multitask Fine-Tuning for Passage Re-Ranking Using BM25 and Pseudo Relevance Feedback," IEEE Access, vol. 10, pp. 54254–54262, May 2022.

[6] M. Cheng, Y. Luo, J. Ouyang, and Q. Liu, "A Survey on Knowledge-Oriented Retrieval-Augmented Generation," *arXiv preprint* arXiv:2503.10677, Mar. 2025.

[7] S. Gupta, R. Ranjan, and S. N. Singh, "A Comprehensive Survey of Retrieval-Augmented Generation (RAG): Evolution, Current Landscape and Future Directions," *arXiv preprint* arXiv:2410.12837, Oct. 2024

[8] Y. Huang and J. Huang, "A Survey on Retrieval-Augmented Text Generation for Large Language Models," *arXiv preprint* arXiv:2404.10981, Apr. 2024.

[9] Wikipedia – Retrieval Augmented Generation

[10] SCORE AI Knowledge Augmentation: Exploring the Future of RAG and Lifelong Learning Systems.

[11] Papers With Code. (2018): RAG Explained.

[12] 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," Online, Nov. 2020, pp. 6769–6781.

[13] N. Reimers and I. Gurevych, "Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation, " Online, Nov. 2020, pp. 4512–4525.

[14] P. Liang et al., "Holistic Evaluation of Language Models (HELM)," *Trans. Mach. Learn. Res.*, 2023. [Online].

[15] Y. Gao et al., "Retrieval-Augmented Generation for Large Language Models: A Survey," arXiv preprint arXiv:2312.10997, Mar. 2024.

[16] Chawla, Avi, and Akshay Pachaar. "A Crash Course on Building RAG Systems – Part 2 (With Implementation)." Daily Dose of Data Science, 10 Nov. 2024,