International Journal of Current Research and Techniques (IJCRT)
E-ISSN: 2349-3194 (Online) | P-ISSN: 2348-4446 (Print)
UGC Approved Journal No: 47722 [No: 2616]
DOI: https://doi.org/10.5281/ZENODO.15450517

RAG Bot for Random Documents

D. Anjaneyulu, *Assistant Professor, KLM college of engineering for women, Kadapa, India*.

P. Bhaskar, *Assistant Professor, KLM college of engineering for women, Kadapa, India*.

S. Harini Yadav, *Assistant Professor, KLM college of engineering for women, Kadapa, India*.

Manuscript Received: May 14, 2025; Revised: May 16, 2025; Published: May 17, 2025

Abstract: The "RAG Bot for Random Documents (Offline)" is an intelligent AI-powered system designed to efficiently extract, understand, and respond to queries based on offline documents. It utilizes the power of Retrieval-Augmented Generation (RAG) to combine the strengths of document retrieval with natural language generation, offering accurate, context-aware answers without the need for an internet connection. The bot supports multiple file formats such as PDF and TXT, making it highly flexible and practical for various user needs, including researchers, students, legal professionals, and business analysts.

Unlike traditional document search tools that rely solely on keyword matching, the RAG Bot uses sentence embeddings and semantic search to understand the true meaning of queries. This is achieved using transformer models like Sentence-BERT for generating embeddings and Flan-T5 for generating natural language answers. The backend incorporates modules for extracting document content, embedding textual data, and indexing with FAISS for efficient similarity-based retrieval. The frontend, built using Flask and HTML/CSS, provides a user-friendly interface for uploading documents and asking questions.

Keywords: RAG, offline, document processing, embeddings, semantic search, Flask, file formats, transformer models.

1. Introduction:

In today's digital age, the exponential growth of data and documents has created a need for smarter, faster, and more reliable systems to manage and extract useful information. Traditional document handling methods are often time-consuming, require manual reading, and fail to provide meaningful insights without significant effort from the user. This issue becomes even more pronounced in environments where internet connectivity is limited or unavailable, making online AI services impractical or insecure. To address these challenges, "RAG Bot for Random Documents (Offline)" presents an innovative solution built on advanced Artificial Intelligence techniques. The RAG (Retrieval-Augmented Generation) Bot combines the best of both retrieval and generative AI models to process offline documents and generate human-like, context-aware responses. It offers significant advancement over existing systems by enabling local AI-powered document understanding without the need for online services.

The bot is designed to support various document types, including PDFs and TXT files. It processes these documents by first extracting text content, converting the content into numerical vectors (embeddings), and storing them in an efficient similarity search index using FAISS. Upon receiving a user query, the system retrieves the most relevant segments of the document and feeds them into a transformer-based language model like Flan-T5 to produce concise, informative answers in natural language. The backend of the system is modular, with each module handling a specific function—ranging from text extraction, embeddings generation, and semantic retrieval to user interface management. This modular structure not only improves the maintainability and scalability of the system but also allows developers to easily upgrade or replace components without affecting the entire system.

The user interface is implemented using Flask and HTML/CSS, allowing users to upload documents and submit queries in a streamlined, web-based environment. The bot displays answers directly on the page, making it convenient for users to interact with and understand complex documents effortlessly. In contrast to online AI bots that depend on cloud servers and remote APIs, this RAG Bot operates completely offline, preserving data privacy and ensuring availability in low-connectivity environments. This makes it highly suitable for use in secure organizations, academic institutions, and remote areas.

International Journal of Current Research and Techniques (IJCRT)
E-ISSN: 2349-3194 (Online) | P-ISSN: 2348-4446 (Print)
UGC Approved Journal No: 47722 [No: 2616]
DOI: https://doi.org/10.5281/ZENODO.15450517

The RAG Bot is not just a tool—it represents a shift in how we interact with documents. By using advanced NLP models and smart retrieval mechanisms, it brings intelligence and ease to document analysis. As data continues to grow, such tools will play a crucial role in improving productivity and decision-making.

2. Literature Survey

This literature survey highlights the existing research, tools, and frameworks that have influenced and inspired the development of the RAG Bot for Random Documents.

1. **Traditional Document Retrieval Systems:** Conventional document retrieval techniques are largely dependent on keyword-based search mechanisms. Tools like Apache Lucene and Elasticsearch have long been used for full-text search capabilities, which allow users to retrieve documents based on the occurrence of specific terms.
2. **Introduction of Semantic Search:** To address the limitations of keyword-based systems, researchers have explored semantic search, which attempts to understand the meaning behind user queries. Technologies such as TF-IDF (Term Frequency-Inverse Document Frequency), Latent Semantic Analysis (LSA), and Latent Dirichlet Allocation (LDA) were early efforts.
3. **Transformer Models and BERT (Bidirectional Encoder Representations from Transformers):** Which can understand natural language process. BERT uses attention mechanisms to consider the full context of a word by looking at the words before and after it, making it exceptionally good at tasks like question answering and semantic similarity.

3. System Analysis

Proposed System

The proposed system introduces an AI-powered solution called the RAG Bot for Random Documents (Offline). It is designed to address the limitations of existing systems by enabling intelligent, context-aware document understanding without the need for internet connectivity. The system leverages advanced Natural Language Processing (NLP) techniques and a Retrieval-Augmented Generation (RAG) framework to extract meaningful responses from documents stored locally. The RAG Bot combines semantic search and AI-generated responses to ensure users receive relevant, human-like answers to their queries. At its core, it uses Sentence-BERT to convert document content into numerical vectors (embeddings), and FAISS (Facebook AI Similarity Search) for fast and accurate retrieval of relevant information. The system architecture is modular and consists of the following components:

- Embeddings Module: Converts document text into embeddings for similarity matching.
- Extract Module: Reads and extracts raw text from PDF and TXT files.
- RAG Module: Retrieves the most relevant information and generates answers.
- Templates Module: Provides the user interface design using HTML.
- App Module: Implements the web server using Flask to handle file uploads and query requests.
- Key Benefits of the Proposed System:
- Provides context-aware answers rather than keyword-based matches.
- Works offline, ensuring data privacy and independence from cloud APIs.
- Supports multiple formats like PDF and TXT.
- Includes an easy-to-use web interface built with Flask and HTML/CSS.
- Scalable and modular for future improvements such as voice input, multilingual support, and real-time chat.

International Journal of Current Research and Techniques (IJCRT)
E-ISSN: 2349-3194 (Online) | P-ISSN: 2348-4446 (Print)
UGC Approved Journal No: 47722 [No: 2616]
DOI: https://doi.org/10.5281/ZENODO.15450517

Feasibility Study

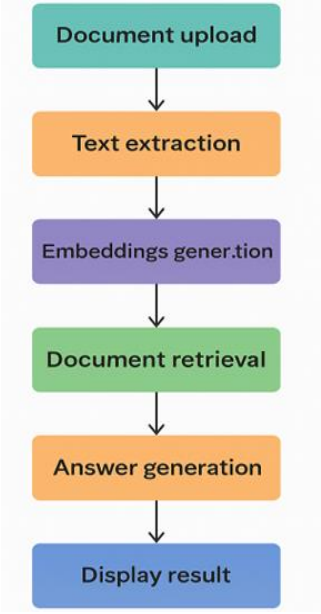
Feasibility study is an essential part of the system analysis process. It helps in evaluating whether the proposed system is practically achievable and worth implementing in terms of cost, technology, and user acceptance. The RAG Bot project was evaluated under the following feasibility dimensions: Economic Feasibility: The RAG Bot system is economically viable as it requires minimal investment in terms of software and hardware. The project is built using open-source libraries and tools such as Flask, FAISS, PyTorch, and Hugging Face Transformers, which eliminate the need for expensive software licenses.

Technical Feasibility

Technically, the system is highly feasible. The Text extraction is handled using reliable Python libraries such as PyPDF2, while embeddings are generated using Sentence-BERT. Document retrieval is powered by FAISS, a high-performance similarity search engine, and the answer generation uses the Flan-T5 model for natural language output. The system's architecture is modular, enabling smooth integration and updates. Furthermore, the offline capability enhances its robustness and reduces dependency on third-party services.

Social Feasibility

From a social perspective, the RAG Bot addresses a real and growing need for intelligent document understanding, especially in environments where internet access is limited or data privacy is a concern. By eliminating the need for internet connectivity, the system is more inclusive and can be used in rural or restricted environments.



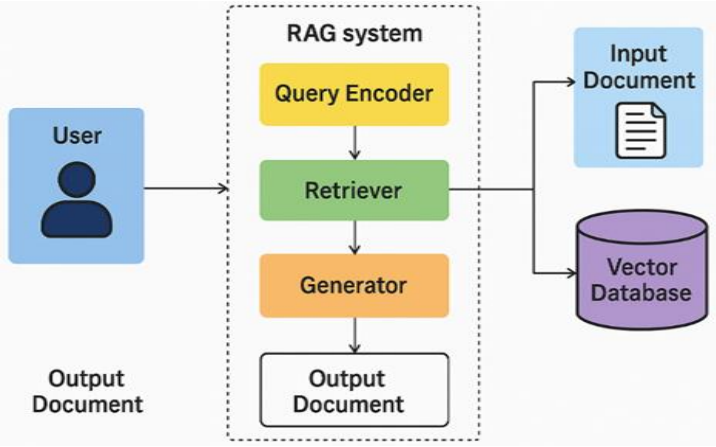


Figure-1 Flow Chart for Proposed System

Figure-2 System Architecture

4. Software and Hardware Requirements

Software Specifications

The RAG Bot system is built using a robust set of open-source tools and frameworks designed for natural language processing, web development, and efficient retrieval. The choice of software ensures scalability, platform independence, and offline operability.

Operating System: Windows 10 or higher

International Journal of Current Research and Techniques (IJCRT)
E-ISSN: 2349-3194 (Online) | P-ISSN: 2348-4446 (Print)
UGC Approved Journal No: 47722 [No: 2616]
DOI: https://doi.org/10.5281/ZENODO.15450517

Programming Language: Python 3.8 or above Python is used for backend development due to its simplicity and rich ecosystem of AI/ML libraries.

Libraries and Frameworks

- Flask – Lightweight Python web framework for building the web interface
- PyTorch – Deep learning framework used for model inference (Flan-T5)
- Transformers (by Hugging Face) – Provides access to pre-trained language models like FlanT5
- Sentence-Transformers – Used to generate vector embeddings (semantic representation of sentences)
- FAISS – Facebook AI Similarity Search for fast and scalable vector search
- PyPDF2 – Library for extracting text from PDF documents
- docx – Library to read text from DOCX files (optional extension)
- NumPy – For efficient numerical operations and matrix handling
- Regex – For basic text cleaning and preprocessing
- HTML/CSS with Bootstrap 5 – For creating a responsive and user-friendly frontend

Development Tools

- Visual Studio Code / PyCharm (for coding and debugging)
- Postman or browser tools (for API testing)
- Git for version control Execution Environment
- Localhost (running completely offline)
- No cloud or internet dependencies required for basic usage.

Hardware Specifications

To ensure efficient execution of the RAG Bot system and support for processing large documents and AI model inference, the following hardware configuration is recommended:

5. System Design

Input Design: The input design in RAG Bot system defines how users interact with the system and how data is collected for further processing. The primary inputs to the system include document files and user queries.

Document Upload

- Format Supported: PDF, TXT
- Purpose: To allow users to upload documents from which text will be extracted and processed.
- Input Method: File selection via web interface (upload form)
- Validation: The system checks file format and size before accepting uploads.

User Query

- Format: Natural language question (typed in a textbox)
- Purpose: To retrieve specific information or summaries from the uploaded document.
- Input Method: Text field on the interface labelled "Ask a Question"
- Validation: Empty queries are rejected with a warning; basic sanitization is applied.

UML Diagrams

Unified Modelling Language (UML) diagrams are essential in system design as they visually represent the structure, behaviour, and interactions between different components of the system. In the RAG Bot, UML diagrams help to understand how data flows through the modules, how different classes interact, and how users interact with the system. They also aid developers in visualizing both high-level workflows and low-level operations, ensuring modular development and clear communication between team members.

International Journal of Current Research and Techniques (IJCRT)
E-ISSN: 2349-3194 (Online) | P-ISSN: 2348-4446 (Print)
UGC Approved Journal No: 47722 [No: 2616]
DOI: https://doi.org/10.5281/ZENODO.15450517

Class Diagram highlights the structure of the key modules: Embedding Model, RAG Model, Extractor, and App, along with their attributes and methods. This modular view ensures each class has a specific role—e.g., handling text extraction, indexing, or query processing.

Sequence Diagram and Activity Diagram: Which illustrate the step-by-step interaction between objects and user actions. The sequence begins with the user uploading a file, continues through the document processing pipeline, and ends with answer generation.

Deployment Diagram shows how the components are distributed across hardware environments, particularly demonstrating that everything runs locally without internet dependency. These UML diagrams together provide a complete technical view of the system's architecture and interactions.

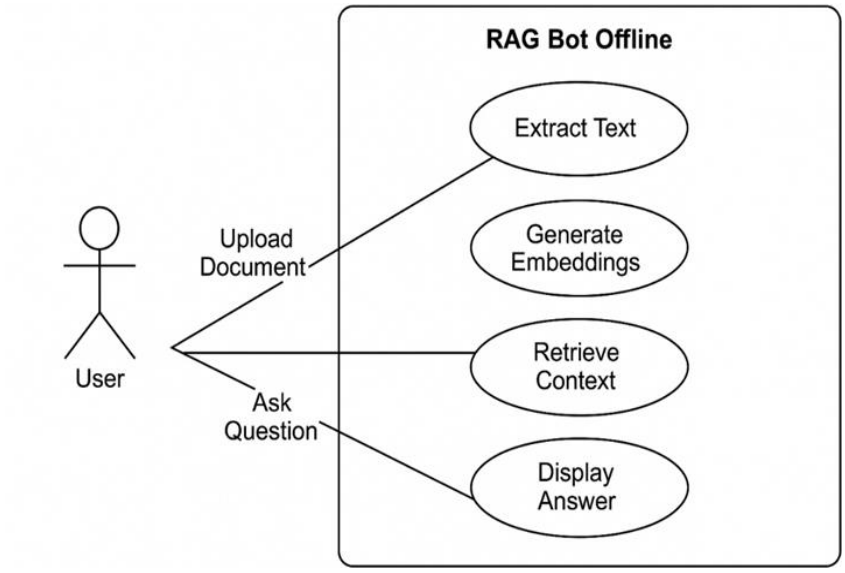


Figure-3 Use Case Diagram for System & User

6. Module Implementation

The system is divided into multiple modules, including:

- App Interface for document upload and question input,
- Text Extraction Module for reading document content,
- Embedding Module for generating semantic vectors,
- Retrieval Module using FAISS,
- and Answer Generator Module powered by a language model.

All these modules are orchestrated using a Flask backend, ensuring smooth communication between the frontend and processing layers. The system also ensures data integrity and privacy by keeping all processes local, making it suitable for educational, legal, and healthcare environments where sensitive documents are handled.

7. Unit Testing

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product.

1. Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application.

International Journal of Current Research and Techniques (IJCRT)
E-ISSN: 2349-3194 (Online) | P-ISSN: 2348-4446 (Print)
UGC Approved Journal No: 47722 [No: 2616]
DOI: https://doi.org/10.5281/ZENODO.15450517

2. Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields.

3. Functional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

4. System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. Ex: white box testing, Black box testing.

8. Output Screens

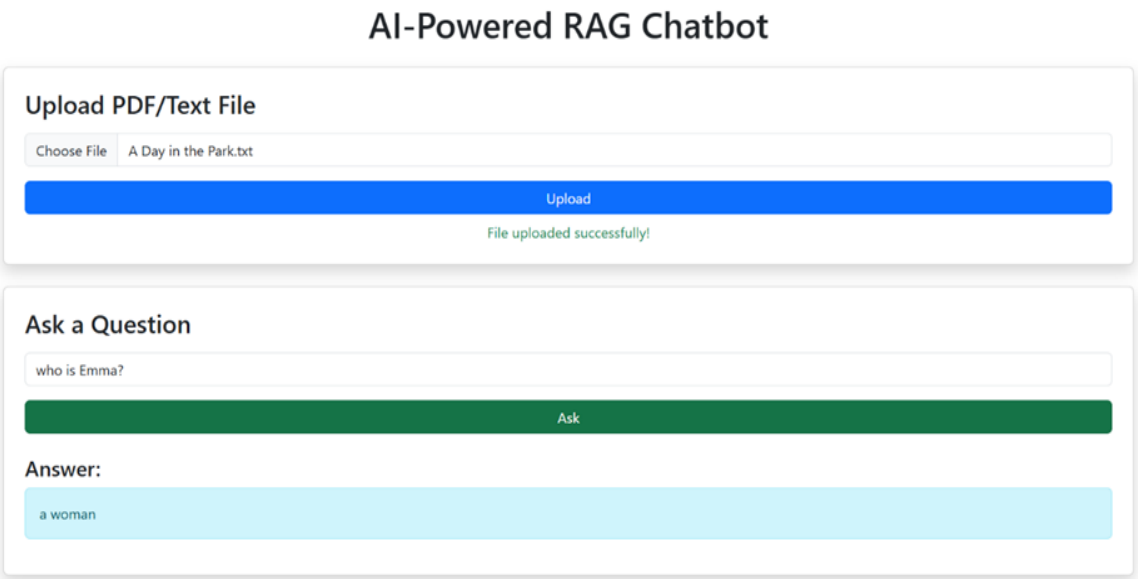


Figure-4 Output

9. Conclusion

The RAG Bot for Random Documents presents a significant advancement in offline document processing using modern artificial intelligence techniques. By combining the power of semantic retrieval through FAISS and context-aware natural language generation using models like Flan-T5, the system efficiently extracts relevant information and delivers accurate, human-like responses to user queries. Unlike conventional search methods that rely on keywords or internet access, this bot functions entirely offline, making it a practical solution for environments that demand data privacy, reliability, and speed.

The system's modular architecture, which includes components for text extraction, embedding generation, retrieval, and answer generation, ensures that it is scalable, maintainable, and flexible for future upgrades. Through a simple yet effective web interface built with Flask, users can easily upload documents and interact with the system without needing technical expertise. Overall, the RAG Bot successfully bridges the gap between static document storage and intelligent document understanding, paving the way for smarter, offline-ready AI solutions.

10. Problem Statement

The modern retail industry is characterized by increasing complexity, demanding consumers, and fierce competition. Retailers grapple with the challenge of managing intricate supply chains that span numerous suppliers, distribution channels, and logistical processes, all while striving to meet fluctuating customer demand and optimize sales performance. Often, valuable data pertaining to supply chain operations and sales activities

International Journal of Current Research and Techniques (IJCRT)
E-ISSN: 2349-3194 (Online) | P-ISSN: 2348-4446 (Print)
UGC Approved Journal No: 47722 [No: 2616]
DOI: https://doi.org/10.5281/ZENODO.15450517

resides in disparate systems and formats, including spreadsheets like Microsoft Excel. This fragmented data landscape makes it exceedingly difficult for retail businesses to gain a holistic and real-time understanding of their overall performance.

This is to bridge the gap between fragmented data and actionable insights by providing a centralized and dynamic platform for analyzing key performance indicators related to both the supply chain and sales functions. By visualizing these interconnected datasets, the report will enable retailers to gain a deeper understanding of the relationships between operational efficiency and revenue generation, leading to more informed strategic and tactical decisions.

11. Scope

The scope of this "E-Commerce Order Processing Dashboard" project is specifically focused on the analysis of retail supply chain and sales data utilizing Microsoft Power BI as the reporting and visualization tool, with Microsoft Excel serving as the primary source of data. This will involve the creation of a data model within Power BI that establishes relationships between the various datasets extracted from Excel. This data model will facilitate the calculation of relevant KPIs and enable the creation of insightful visualizations that demonstrate the interplay between supply chain activities and sales outcomes.

The sales analysis component will encompass the examination of sales transactions, including sales volume, sales value, revenue trends over time, product-level sales performance, and potentially customer-related sales data such as purchase frequency and average order value, again, depending on the data available in the Excel files. The analysis may also explore sales performance across different geographical regions or sales channels if this information is present in the data.

12. Microsoft Power BI

Power BI Desktop integrates seamlessly with the Power Query Editor, a robust data transformation.

13. References

[1] Bird, S., Klein, E., & Loper, E. (n.d.). Natural Language Processing with Python.
[2] Russell, S., & Norvig, P. (n.d.). Artificial Intelligence: A Modern Approach.
[3] Ng, A. (n.d.). Machine Learning Yearning.
[4] LangChain Documentation. (n.d.). https://python.langchain.com
[5] OpenAI API Documentation. (n.d.). https://platform.openai.com
[6] Facebook AI Similarity Search (FAISS). (n.d.). https://faiss.ai
[7] GeeksforGeeks. (n.d.). Python Programming Tutorials. https://www.geeksforgeeks.org
[8] Medium. (n.d.). Articles on Retrieval-Augmented Generation (RAG). https://medium.com.