



iJRASET

International Journal For Research in
Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** X **Month of publication:** October 2026

DOI: <https://doi.org/10.22214/ijraset.2026.85226>

www.ijraset.com

Call: ☎ 08813907089

E-mail ID: ijraset@gmail.com

Privacy-Preserving Local Retrieval-ALLM

Mohan Mene, Pranav Ingle, Dipanshu Langote, Kunal Bhagat, Om Meherkure, Aditya Singh
Department of Computer Science and Engineering, G H Rasoni University, Amravati, Maharashtra, India

Abstract: *With the growing use of cloud-based artificial intelligence (AI) systems for document analysis, there are mounting concerns regarding user privacy. Certain documents may contain private, academic, organizational, or legally protected information that would need to be redacted or withheld before extracting meaningful information from them. This paper proposes Privacy-Preserving Local Retrieval-ALLM, a local document question-answering system capable of performing information retrieval and generating responses based on the contents of the documents. The system described in the paper uses the following process. First, it ingests user documents, performing optical character recognition if necessary, and segments them into text parts and embeddings. Next, the system stores the embeddings in a local vector database for similarity search. The system then employs a local Qwen 2.5 3B Instruct model to generate responses from the retrieved information. Finally, the system provides an optional permission-based web search if the retrieved document does not have the necessary information. In such cases, the solution does not need to transmit the contents of the documents to a third party but still has access to the necessary information. In conclusion, the prototype presented in the paper is an example of how local document analysis, semantic search, and local large language model (LLM) can be combined into a privacy-preserving question-answering system.*

Keywords— *ChromaDB, Document Question Answering, Local LLM, Optical Character Recognition, Privacy-Preserving AI, Semantic Retrieval, Vector Database*

I. INTRODUCTION

Recent developments in artificial intelligence (AI) have seen document analysis and question-answering technology become sufficiently mature for the majority of users to now be able to easily access. AI technologies can be used to derive data from documents, generate summaries of documents as well as providing explanations in natural language. These currently usually come in the form of cloud-based services, which require your data to be sent to servers away from your device. This poses a critical privacy risk for personal, academic, organizational, financial and other sensitive data.

From running language models locally we can also derive an alternative approach in which data is processed locally on your own computer. Recent breakthroughs in lightweight large language models, semantic text embeddings and vector databases enable building document question answering systems without depending on external AI services. But running a language model locally is not enough to build a document question answering system. The system must be architected such that it can extract information from documents, retrieve relevant passages from the documents and rely on language model abilities to generate an answer.

Privacy-Preserving Local Retrieval-ALLM refers to document analysis (OCR on scans, and tokenization), semantic retrieval, and local LLM inference. This architecture first analyzes the uploaded documents: for scans, it runs the OCR; it reduces the document to smaller meaning-carrying units called tokens. It translates the tokens into semantic embeddings and stores the embeddings in a local vector database. It employs the semantic model to search for the relevant parts of the document; it uses the local LLM, specifically, the LLM known as Qwen 2.5 3B Instruct, to generate the responses. Here, the LLM answers questions in light of the extracted document information generated by the document analysis process implemented by the user.

While, on the other hand, the system can be extended with the addition of a permission-based layer for web search for those on which the information required is not in the documents or the local pipeline. This layer would provide more extensive evidence and in turn more complete answers as now the system would be able to use the information available on the web. At the same time, however, the architecture would still be based on local text document processing.

The aim of the system as outlined in this paper is therefore to develop the set-up of a Privacy-Preserving Local Retrieval-ALLM that can be used to reduce the reliance on external services to execute all the processes involved in the retrieval process and provides the ability to extract relevant information from text documents. It combines the local processing architecture for OCR, text tokenization, semantic embedding creation, vector data storage, similarity retrieval, and large language model inference. The implementation from the team exemplifies the possibility of the approach to process private documents, the components of the proposed system that have been implemented in the team project include MiniLM based embeddings, ChromaDB based local storage and similarity search, Qwen 2.5 3B Instruct, and permission based web access.

II. RELATED WORK

Emerging updates from NLP research demonstrate that language models currently allow for an extensive list of question-answering, knowledge seeking, and information extraction tasks. On the other hand, models that rely heavily on pretrained knowledge may not have access to information contained in private documents or information recently added to a base of knowledge. NLP is responsible for human language understanding and generation, while retrieval-based methods will probably provide contextual information to be used in the answer generation.

Retrieval-Augmented Generation (RAG) retrieval augmentation the technique of RAG incorporates a language model's knowledge in its parameters as well as the retrieval of information from an external knowledge base. Based on studies by Lewis et al, the source, retrieved passage, can also be incorporated into the generation, and be used to perform knowledge intensive NLP tasks by giving the model access to external information not just the pre-trained parameters. [1] This is used in document question-answering systems, where relevant segments are retrieved then answered.

The retrieval will only be successful if a model can instantiate the query and document segments in such a way that their meaningful similarities can be identified. Sentence-BERT (SBERT), a model created by Reimers and Gurevych, computes segment-level embeddings that can be efficiently compare using similarity metrics like cosine similarity. These representations allow the use of a semantic search by transforming the query and document segment to the same vector space.[2]

The document question-answering system also needs to consider the precondition of documents, which contain the needed information solely in the form of images or scans. The document-to-machine translation engine, called optical character recognition (OCR) engine, introduces a solution to this problem. Smith gives an extensive description of the Tesseract OCR engine, including how it is used in document analysis and character recognition. And in the near future, researchers will look into how OCR errors can affect document retrieval-augmented generation pipelines.[3]

Smaller open-weight LLMs have made local inference more accessible Open and highly performant foundational language models recently developed by Touvron et al., such as the LLaMA family of models, have made it more practical to perform local inference [4]. The Qwen2.5 family of models comprise models of various sizes and instruction tuned models for a variety of language-model tasks [5]. These models enable

researching local language-model use cases where computational resources, data control, or privacy concerns would make relying on only cloud-hosted models ill-advised.

Conversations about relevant documents and development of LL model-based applications have already been addressed through existing projects and frameworks PrivateGPT for document conversation with privacy and locality in mind [6]. LangChain for application development focused on language models and retrieval from text [7]. Multiple tools for web scraping that can access web sources if data needed cannot be found in existing local text sources [8].

To summarize, the state of the art already defines the building blocks for retrieval-based document question answering. Semantic embedding for information retrieval, RAG for allowing language models to incorporate retrieved context, OCR for information retrieval from image-based documents and compact language models for local inference are already in use. The proposed Privacy-Preserving Local Retrieval-ALLM brings together these components to form a local-first document question answering flow and additionally offers permissioned web retrieval in cases where required information cannot be retrieved from the local document context.

III. PROPOSED METHODOLOGY

The Privacy-Preserving Local Retrieval-ALLM is designed as a local document question answering system that is designed to operate primarily within the user environment for document processing, retrieval and answering. The approach includes document ingestion, text extraction, text segmentation, semantic embedding vector generation, vector database storage, retrieval by similarity search, and answering by an LLM running locally. If not enough information is available in the current context, then a permission-based web retrieval system may be used.

A. Document Ingestion and Text Extraction

First, the acceptance of a user-supplied document. Machine-readable text documents will be handled directly, but scanned or image-based document formats require optical character recognition (OCR). Text is extracted from document images for documents that are PDF or image-based, making sure that the system can work on such documents without natively selectable text.

B. Text Segmentation

Document and Chunks Once the document has been extracted, the information is separated into chunks or smaller Text segments. Document chunks allow the retrieval system to drill down to the source versus pulling in the entire source. The chunks are stored with the document context and the content can be retrieved when responding to a query.

C. Semantic Embedding and Local Vector Storage

Every bit of text in a document is also transformed into vectors through the MiniLM based embedding model. The conversion captures a semantic representation of the section of a document, which is then stored locally in the ChromaDB vector database. Once the user provides a query, it is also transformed to an embedding and matched to find the most relevant semantic sections of the document.

D. Similarity-Based Retrieval

Next, the query embedding is used to find the most relevant text segments from the documents in context of the user's query. The text segments that are returned to the user serve as context for the model. This retrieval step reduces the model's dependency on pre-trained knowledge and helps the model answer user queries based on files uploaded.

E. Local Response Generation

The document context obtained is passed to the local Qwen 2.5 3B Instruct deployment via prompt augmentation. Local Qwen 2.5 3B Instruct generates a response based on the provided context and user query. The document context provided to the language model for answering the query is kept inside the local QnA pipeline and not transferred to an external AI service.

F. Permission-Based Web Retrieval

If the above needed knowledge cannot be retrieved from local document context, an optional web retrieval component is provided in the system. Unlike conventional QA systems, the web access would be queried on demand and provided only upon user approval. Web access operates as a supplementary knowledge source and retains the local first principle in document processing pipeline. The project prototype is capable of question answering from local document context and web retrieval.

G. Overall Processing Workflow

In brief, this can be summarized as below: User Document Text / OCR Text Extraction Text Segmentation MiniLM Embeddings ChromaDB Query Embedding Similarity Search Retrieved Context Qwen 2.5 3B Instruct Generated Answer If anything is unable to find locally, this can fallback as follows: Query Request for Access Retrieval from Web Extra Context Response. This setup combines document ingestion, semantic search, local vector store and local LLM inference together in a simple privacy-preserving QnA pipeline.

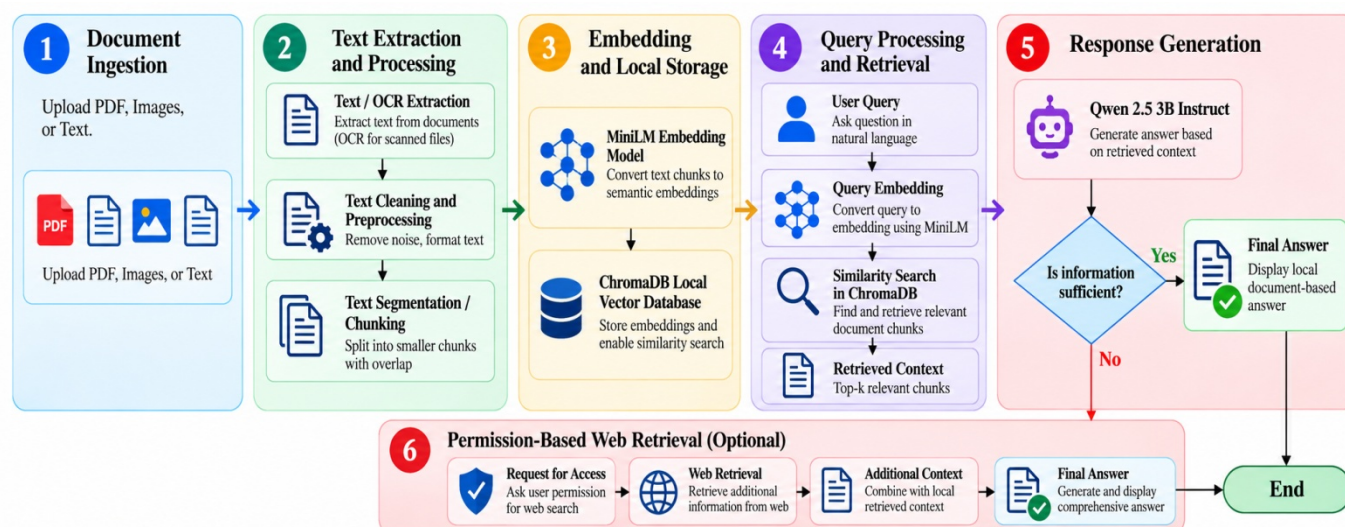


Figure III.1. System Workflow

IV. IMPLEMENTATION

The system was designed as an end-user document question-answering system with sections for document processing, semantic retrieval, local vector storage, and generation of answers from language models. The implementation follows the approach described in the previous section and is intended to execute all of the main document processing and inference functionality on the end-user machine.

A. Document Processing

I have a requirement to process the document as given, and the application or user must pass the document to the component in the form of a document that the component can read the text from. If you have image-based text or scans, run through OCR to get text as I am assuming you will need to be able to handle both types.

B. Embedding and Vector Database

The text that is extracted from documents is split into chunks and converted into semantic vectors using a MiniLM-based embedding model. The embedding vectors are stored locally in ChromaDB. The vector database offers the persistence and similarity-search function to find relevant parts of the documents for a user query.

C. Retrieval and Context Construction

When a user inputs a question, that question will be mapped into a question embedding, and this embedding will be compared with the document embeddings stored in the database. Relevant segments of documents will be retrieved based on the semantic similarity to this question embedding. These segments will then be assembled to form the context for the response-generation model.

D. Local Language Model

Response-generation Qwen 2.5 3B Instruct 3B is the local language model used in the project implementation. The search document context is included in the prompt to the model, Qwen 2.5 3B Instruct, which probably produces the answer by access to search results from user documents.

E. Web Retrieval Fallback

Furthermore, the implementation employs a permission-based web retrieval for when the requested information cannot be retrieved from the available local document context. The system may request permission to retrieve information from the web as a secondary source of information, while following the local-first paradigm.

F. Local Processing Architecture

The design allows for a local process path from ingestion, to retrieval, to answer generation. The central components can work together as a pipeline – the text that has been extracted is embedded, saved locally in a vector database, retrieved in accordance with the query's relation to an embedded chunk, and used as context for local language model. Web retrieval is secondary, not primary.

V. RESULTS AND DISCUSSION

Testing of the prototype using local document question-answering pipeline as well as a web retrieval pipeline of the constructed workflow. The goal of the test was to ensure if the modules could extract insights from the documents that were uploaded and use these insights to find answers from local language models and provide additional insights.

A. Local Document Question Answering

It is shown on the prototype that the system could receive an uploaded pdf file and the knowledge base could be supplied by the content in that pdf file. The data from a pdf file was saved into the local vector database as semantic embeddings. When a user entered the question asked to the uploaded pdf, the answer could be generated from the local deployed language model by feeding the part of the uploaded data back to the language model. The project demo showed that the system could generate answers from the uploaded files without asking any questions to the AI api service.

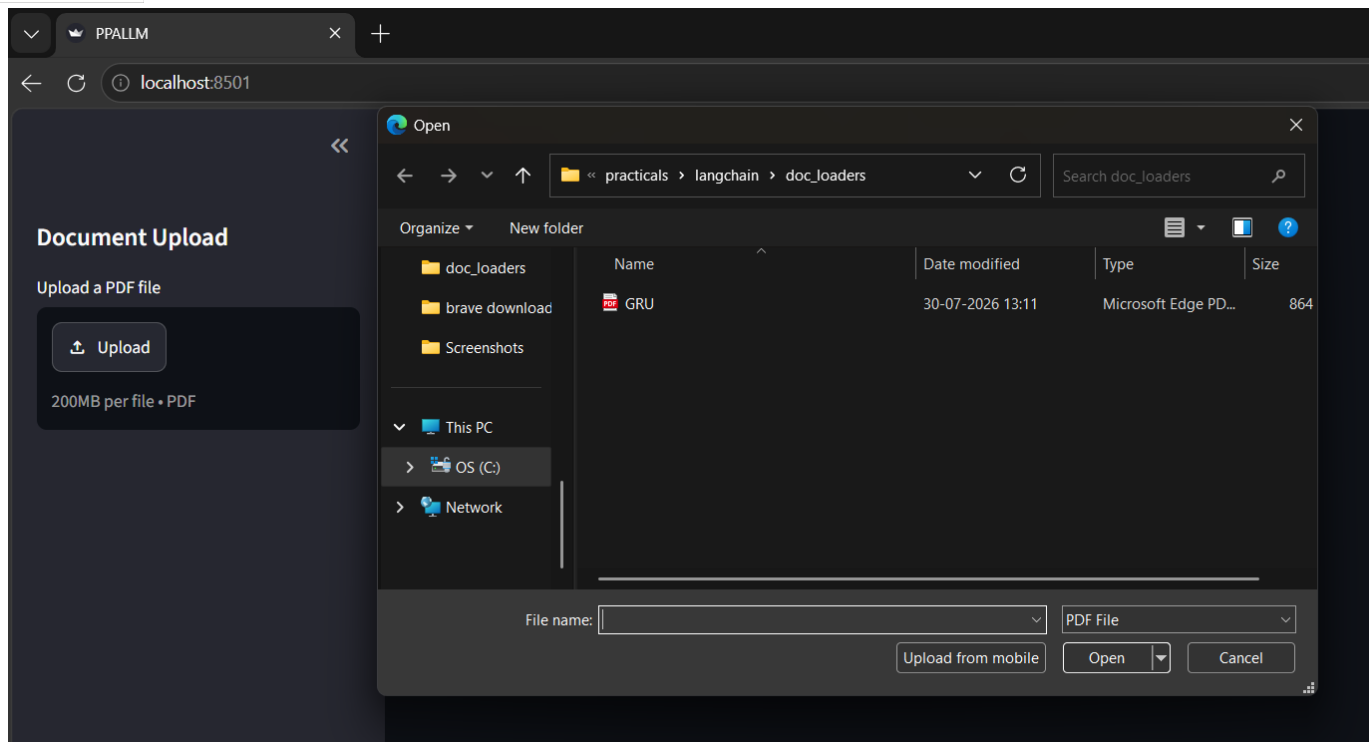


Figure IV.1. Document Upload

B. Semantic Retrieval

The retrieval part of the model is what helps the model find a piece of the document that is related to the question. Instead of taking the entire document and feeding it into the language model, smaller chunks are retrieved from the local representation. This then creates a small context for which the answer can be generated from. This is the retrieval stage.

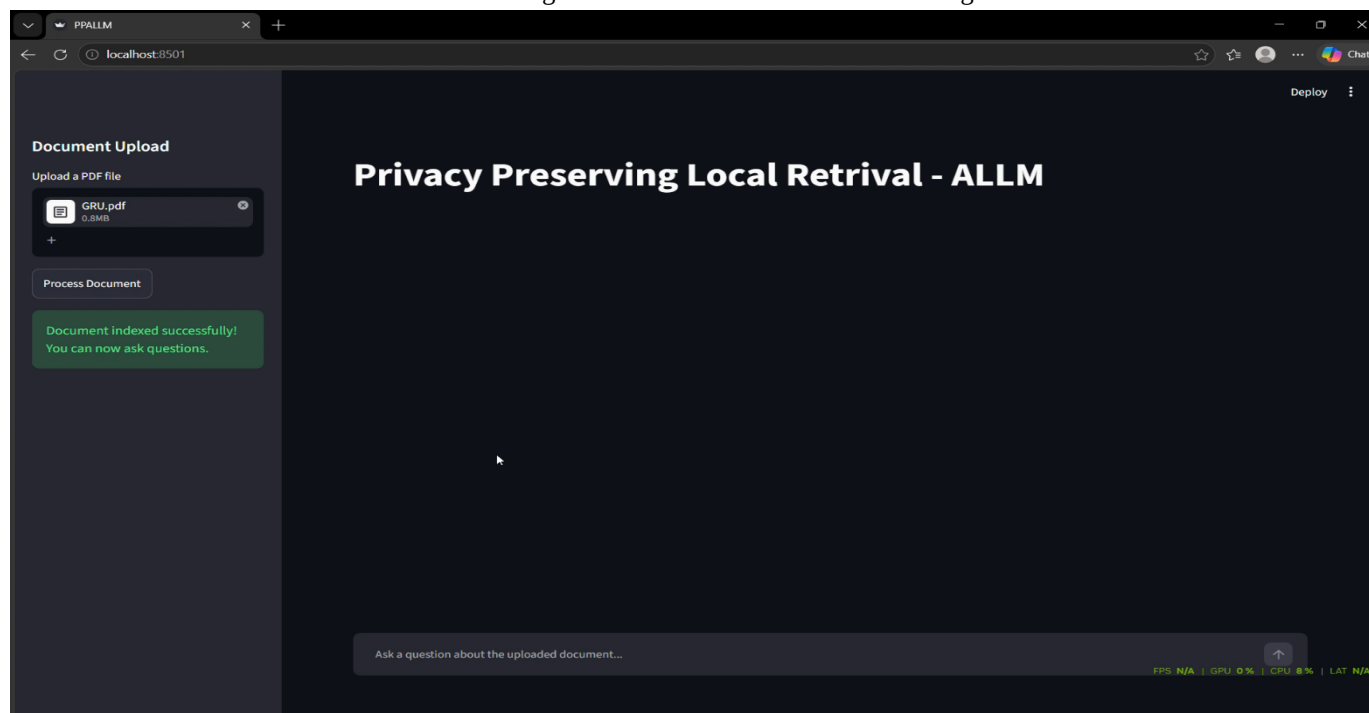


Figure IV.2. Indexing and Semantic Retrieval After Doc Upload

C. Local Response Generation

This prototype takes the Qwen 2.5 3B Instruct model to produce local answers. We re-insert the retrieved document context back into the model input and generate responses based on the local semantic information from the document that you provided. This demonstrates how we combine local semantic retrieval with local LLM inference for our approach.

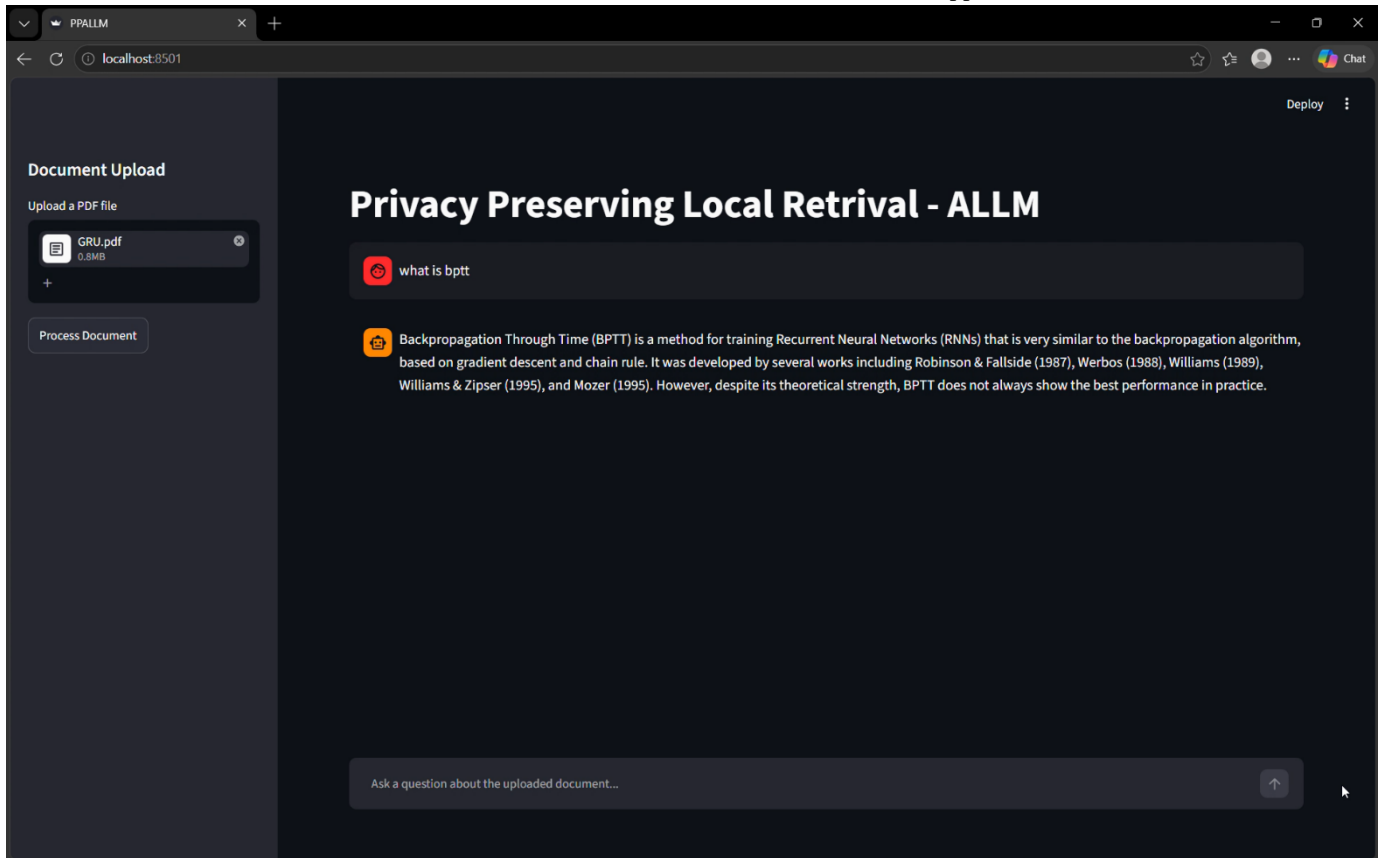


Figure IV.3. Local Response Generation

D. Permission-Based Web Retrieval

The second mode of operation has been illustrated by this prototype. The system will be able to use the permission-based web retrieval to retrieve supplementary data if the query cannot be answered based on the uploaded document. Based on the project demonstration, the system is capable of providing answers for queries whose answers are outside the PDF document.

E. Discussion

The functional results indicate promise for a one-query approach to document question-answering that puts together end-to-end OCR document extraction, document semantic embedding, local vector database, document retrieval and local LLM inference. The local-first approach is presumably going to limit the need to export the document contents to AI services. The permission-enabled web retrieval is another potential information source if a local document context is inadequate.

The evaluation below is purely functionality-based, displaying system functionalities, as it does not provide any quantitative comparison among other question-answering systems. For the current implementation, there is no metric of retrieval accuracy, answer accuracy, speed, memory usage, and cloud model comparison, hence, these metrics are to be tested as future evaluations.

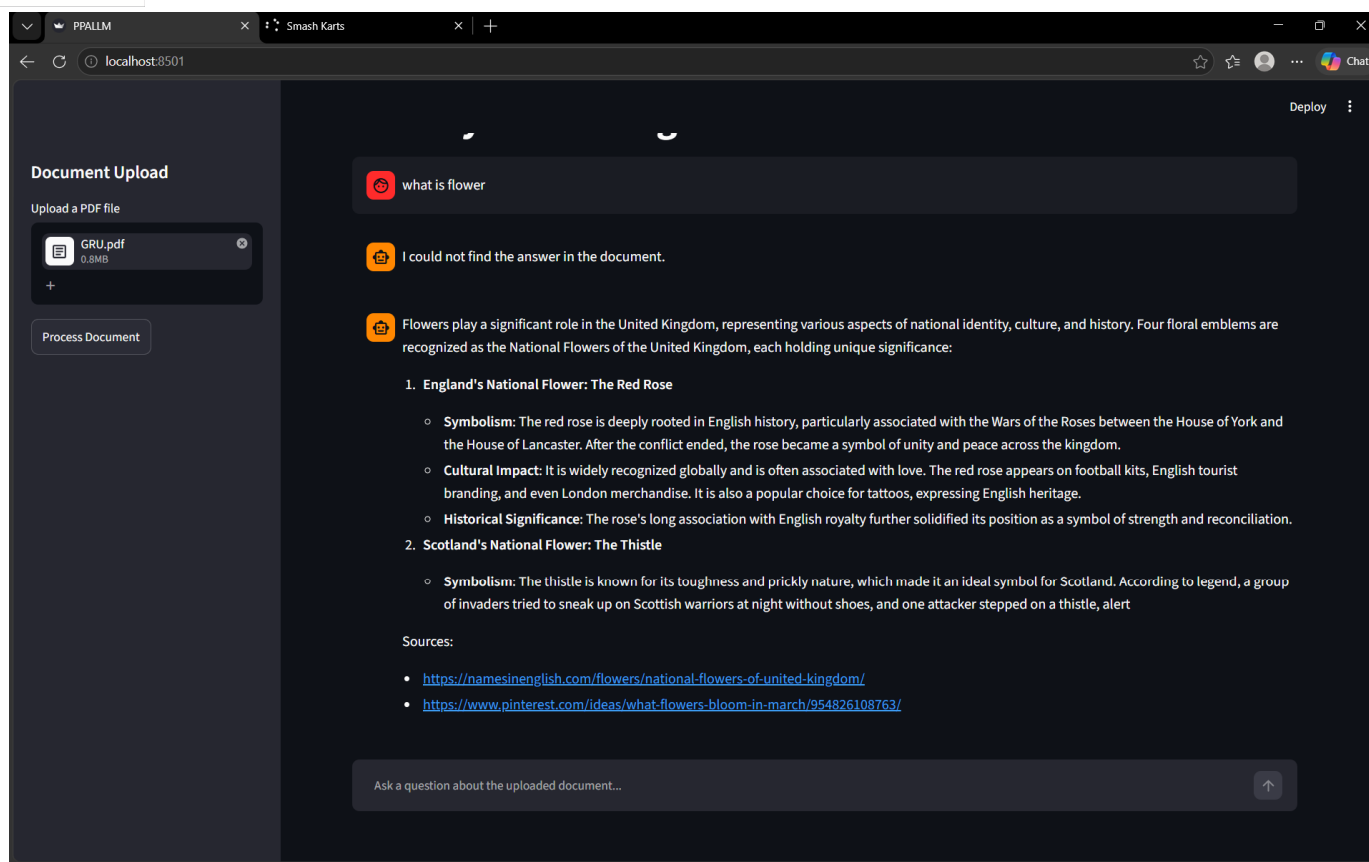


Figure IV.4 Web Retrieval and Response Generation

VI. CONCLUSION

How does this Prototype work? The Qwen 2.5 3B Instruct model above, as illustrated in this demonstration, shows what it's able to do based on the context inside the documents. This answers in a document-based question answering, which uses a local processing pipeline: text extraction with optical character recognition (OCR), document text segmentation, semantic embedding generation, local vector storage, local vector similarity search, and local LLM inference. These stages (...) collectively comprise the operational pipeline for this prototype.

Additionally, the proof of concept uses a permissioned web retrieval engine to enhance the language model with data if the data requested is not available within the context of the document you provided. This adds an additional source of information but this architecture makes local document information retrieval the primary operation.

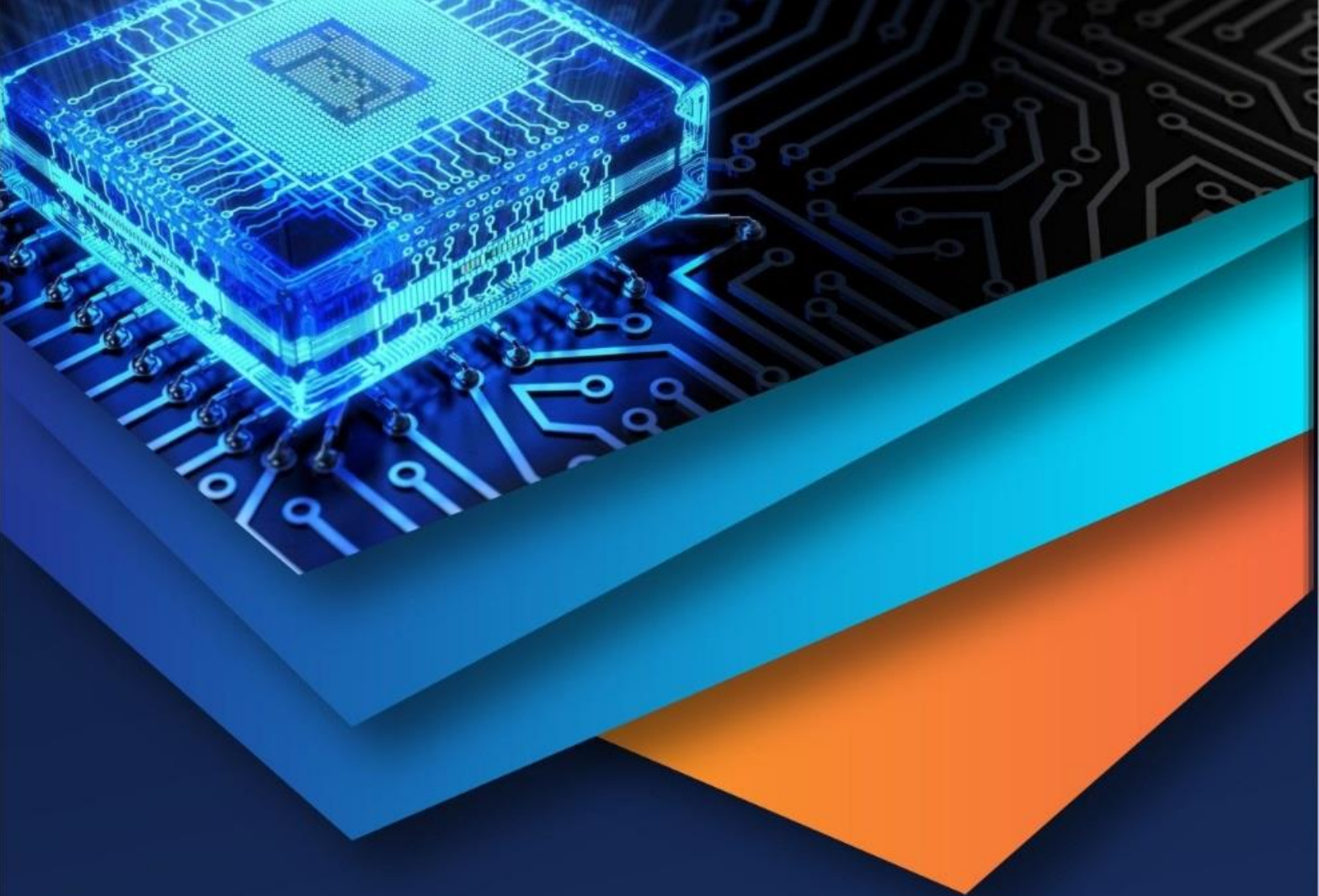
What the existing implementation provides is the core functionality of this method. Evaluations have been only based on function (not quantitative measures, including retrieval accuracy, answer accuracy, latency, resource consumption, etc.); additional efforts in the future are probably the systematic benchmarking, local inference optimization, testing on diverse types of documents and datasets, as well as the comparison of retrieval and response-generation efficiencies.

REFERENCES

- [1] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.
- [2] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in *Proc. 2019 Conf. Empirical Methods in Natural Language Processing and 9th Int. Joint Conf. Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China, 2019, pp. 3982–3992.
- [3] R. Smith, "An Overview of the Tesseract OCR Engine," in *Proc. Ninth Int. Conf. Document Analysis and Recognition (ICDAR)*, vol. 2, 2007, pp. 629–633.
- [4] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "LLaMA: Open and Efficient Foundation Language Models," *arXiv preprint arXiv:2302.13971*, 2023.
- [5] Qwen, A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Lin, J. Yang, J. Tu, J. Zhang, J. Yang, J. Zhou, J. Lin, K. Dang, K. Lu, K. Bao, K. Yang, L. Yu, M. Li, P. Zhang, Q. Zhu, R. Men, R. Lin, T. Li, T. Tang, T. Xia, X. Ren, X. Zhang, Y. Fan, Y. Su, Y. Zhang, Y. Wan, Y. Liu, Y. Cui, Z. Qiu, Z. Zeng, Z. Liu, and Z. Wang, "Qwen2.5 Technical Report," *arXiv preprint arXiv:2412.15115*, 2024.



- [6] I. Martínez Toro, D. Gallego Vico, and P. Orgaz, "PrivateGPT," GitHub repository, 2023.
- [7] H. Chase, "LangChain: Building Applications with LLMs through Composability," GitHub repository, 2023.
- [8] C. Bhatt, A. Bisht, R. Chauhan, A. Vishwakarma, M. Kumar, and S. Sharma, "Web Scraping Techniques and Its Applications: A Review," in Proceedings of the 2023 3rd International Conference on Innovative Sustainable Computational Technologies (CISCT), 2023, pp. 1–8, doi: 10.1109/CISCT57197.2023.10351298.



10.22214/IJRASET



45.98



IMPACT FACTOR:
7.129



IMPACT FACTOR:
7.429



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089  (24*7 Support on Whatsapp)