Document-based Question Answering System

1. Introduction

The Document-based Question Answering (QnA) System is an AI-powered application designed to allow users to upload documents in various formats (PDF, Word, text) and ask natural language questions. The system responds with relevant answers based on the content of the documents. It uses open-source tools and a lightweight language model to ensure fast inference, cost efficiency, and ease of deployment.

2. System Process Flow and Architecture

The core workflow of the Document QnA system involves the following steps:

1. User uploads one or more documents via a simple user interface.

2. The documents are processed and converted to plain text using format-specific parsers.

3. Text data is split into chunks and vectorized using pre-trained sentence transformers.

4. The chunk embeddings are stored in a FAISS vector database for fast similarity search.

5. When a user submits a question, it is embedded and compared against the indexed data.

6. The most relevant chunks are retrieved and passed along with the question to a local LLM.

7. The language model generates an answer using the retrieved context.

8. The answer is presented to the user in the web interface.

System Architecture Overview:

[User Interface]
    |
[Document Upload]
    |
[Document Loader (PDF, DOCX, TXT)]
    |

[Text Splitter and Embedder]
|
[FAISS Vector Store] <--- Embedding Generation (Sentence Transformers)
|
[Question Input] ---> [Embedding] ---> [Similarity Search]
|
[Relevant Chunks + Question]
|
[Local Language Model (e.g., Tiny LLM)]
|
[Answer Generation]
|
[Response to User]

3. Benefits of the System

1. No dependency on proprietary APIs; fully open-source stack.
2. Fast inference using lightweight local LLMs, ideal for on-premise deployment.
3. User-friendly interface built with Streamlit for ease of use.
4. Handles multiple document formats including PDF and DOCX.
5. Avoids cloud cost overhead by running entirely offline.
6. Scalable design: supports additional document types and larger models if needed.
7. Encourages privacy and security by keeping document data local.

4. Technology Stack Used

1. Python 3
2. Streamlit (for the web interface)
3. LangChain (document loading, QnA pipeline)
4. FAISS (vector similarity search)
5. Sentence Transformers (for text embeddings)
6. PyPDF and docx2txt (for document parsing)

7. HuggingFace Transformers (for local language model)

5. Future Work

1. Integrate image-based QnA capabilities by leveraging OCR tools like PaddleOCR to extract text from images (e.g., scanned documents, ID cards).

2. Add support for additional file types like Excel and HTML.

3. Incorporate feedback loops to improve model accuracy over time.

4. Implement document classification and tagging features.

5. Deploy the solution using containerization (Docker) and CI/CD pipelines for scalable production readiness.

6. Introduce user authentication and document access control.