# **Dr. X Research Assistant - Technical Documentation**

## Overview

Dr. X Research Assistant is an AI-powered tool designed to process and analyze documents for summarization, translation, and question answering. It supports multiple file formats, including .docx, .pdf, .xlsx, .csv, and .txt. The system leverages local LLMs and vector databases to ensure offline functionality and efficient document processing.

## **Features**

1. **Summarization**: Generate concise summaries of technical and academic content.
2. **Translation**: Translate documents into multiple languages.
3. **Question Answering:** Answer questions based on the content of uploaded documents.

**­**

## **Methodology**

Built using a modular and scalable architecture that integrates document processing, vectorization, and interaction with a local LLM (Large Language Model). Below is a detailed explanation of the methodology:

**Document Upload and Processing**

* Users upload documents in supported formats through the Streamlit-based web interface.
* Uploaded documents are temporarily stored in the `temp/` directory for processing.
* The system ensures that the uploaded files are handled securely and deleted after processing to maintain user privacy.

**File Reading**

* TextLoader for plain text files.
* PyPDFLoader for PDF files.
* Docx2txtLoader for Word documents.
* UnstructuredExcelLoader for Excel files.
* CSVLoader for CSV files.
* Metadata such as the source and page number is added to each document chunk, which is critical for traceability during question answering and summarization tasks.

**Text Chunking**

* Chunking ensures that the text fits within the token limit of the LLM and improves processing efficiency.
* The chunk size and overlap are configurable, with default values of 1000 tokens per chunk and 300 tokens overlap. Overlapping ensures that context is preserved across chunks, which is particularly important for tasks like summarization and question answering.

**Vectorization**

* The `Qdrant` database is initialized in-memory for fast access during runtime. This approach is optimal for real-time applications where low latency is critical.
* The embedding model (`nomic-embed-text:latest`) was chosen for its ability to generate high-dimensional vector representations that capture semantic meaning effectively, enabling accurate retrieval of relevant chunks.

**LLM Integration**

* The project uses the `llama3.2:3b` model, accessed via the Ollama server, for tasks such as summarization, translation, and question answering.
* Temperature: Set to 0.6 to balance creativity and determinism in responses.
* Context Size: Limited to 512 tokens to ensure compatibility with the model's architecture.

**Multi-threading**:

* Utilizes up to 90% of available CPU cores for efficient processing, making it suitable for local deployment on high-performance machines.The LLM was chosen for its ability to handle diverse NLP tasks with high accuracy and its compatibility with local deployment, ensuring data privacy.

**Intent Detection:**

* Intent detection ensures that the system routes the query to the appropriate processing pipeline, optimizing resource utilization.

**Task Execution:**

Based on the detected intent, the appropriate processing pipeline is triggered:

* Summarization

Each chunk is summarized individually, and the final summary is a concatenation of all chunk summaries. This approach ensures that the entire document is covered while maintaining coherence.

* Translation

Supported file types include `.docx`, `.pdf`, `.xlsx`, `.csv`, and `.txt`. The translation process preserves the structure of the original document, ensuring that the output is user-friendly.

* Question Answering

Relevant document chunks are retrieved from the vector database using similarity-based search. The similarity search is powered by the `Qdrant` vector database, which uses cosine similarity to rank chunks based on their relevance to the query.

The `answer\_question` function in [`main.py`](main.py) generates answers based on the retrieved context and user query. The LLM is provided with both the query and the retrieved context, enabling it to generate accurate and contextually relevant answers.

**Evaluation**

* Calculates ROUGE scores to evaluate the quality of generated summaries. ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a widely used metric for summarization tasks, comparing the overlap between the generated summary and a reference summary.

**User Interaction**

* The Streamlit interface provides a chat-like experience for users to interact with the assistant.
* Users can upload documents, ask questions, request summaries, or translate documents through the interface. The interface is designed to be intuitive, making it accessible to users with minimal technical expertise.

The modular architecture allows for easy extension and scalability, making the system adaptable to future requirements.