Project: The Enigmatic Research of Dr. X

Author: Muhammed Yaseen TP

Date: 21-04-2025

1. Project Overview

This project focuses on developing an end-to-end NLP pipeline designed to process, analyze, and extract valuable insights from the research materials left behind by Dr. X. The primary goals of the system include:

- **Content Extraction**: Accurately extract both text and tables from various document formats, including .docx, .pdf, .csv, and .xlsx.
- **Token-Aware Chunking**: Segment the extracted content using the cl100k_base tokenizer to ensure context-preserving and model-compatible input chunks.
- **Embedding Generation**: Generate high-quality vector embeddings using the nomic model to represent semantic content.
- **Vector Store Integration**: Store the embeddings along with relevant metadata in a local vector database for efficient retrieval.
- **RAG-Based Question Answering**: Enable intelligent querying through a Retrieval-Augmented Generation (RAG) framework powered by a local large language model (LLM).
- **Translation and Summarization Support**: Incorporate features for translating and summarizing extracted content, with mechanisms in place for performance evaluation.2. System Architecture

2.1 Architecture Diagram

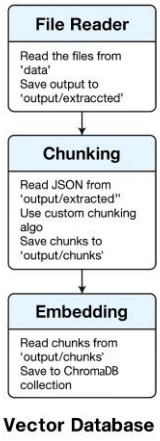


Fig: Data Processing Pipeline

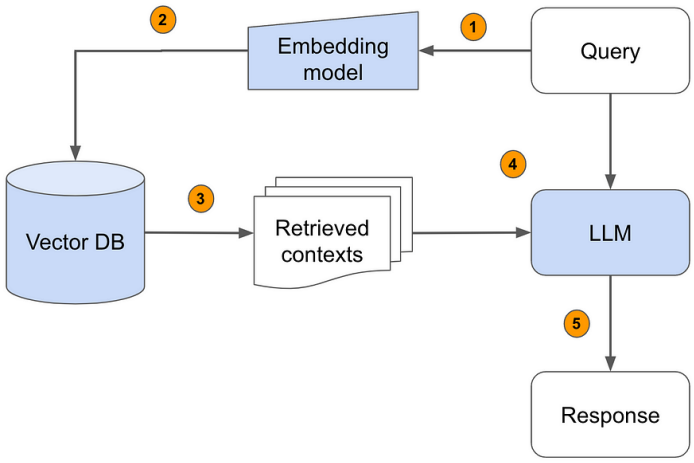


Fig: RAG Architecture

2.2 Component Descriptions

1. **File Reader:**

A custom document reader was developed to extract structured content from .pdf and .docx files. Rather than treating documents as flat text, the reader intelligently identifies and preserves the hierarchical layout—including sections, headers, paragraphs, and tables. Each logical unit is stored in a structured JSON format, which is critical for enabling downstream processes (such as context-preserving chunking, targeted retrieval, and translation). This structure ensures that no contextual information is lost in early stages of the pipeline.

2. Chunking:

To manage large documents efficiently, I designed a **custom token-aware chunking algorithm** that utilizes the cl100k_base tokenizer. Instead of naïvely splitting by fixed length, the algorithm traverses the JSON structure hierarchically and chunks the content while respecting section boundaries. This approach includes a fallback mechanism that avoids cutting across meaningful semantic units (like paragraphs or section headers),

thereby **preserving context** crucial factors for both summarization and accurate Q&A responses. Each chunk includes detailed metadata: file name, page number, and chunk ID, which ensures robust traceability throughout the system.

3. Vector Database:

implemented a custom ChromaDB manager to efficiently handle embeddings and metadata. Text chunks are vectorized using the nomic-ai/nomic-embed-text-v1 model, chosen for its high performance on long-text embeddings. The metadata-enhanced entries in the vector database support **fast, accurate, and traceable retrieval**, forming the foundation of a high-performance RAG system.

RAG System:

A graph-based RAG system was developed to connect vector search, memory, and prompting modules, enabling modularity and scalability. The user query is embedded using the same embedding model and passed through the retrieval module, which fetches the most relevant chunks from ChromaDB. These are then injected into the LLM prompt as context. I used **gemma:3-4b-it-qat**, hosted via **Ollama** on my local system, and wrapped it using Langchain's OllamaChat class to enable efficient, offline LLM usage. The RAG pipeline supports memory, allowing it to maintain conversation history and deliver a chatbot-like experience.

3. Tools and Methodology

Component	Tools/Models Used
File Reader	Python_docx, PDFplumber
Chunking	Cl100k_base
Embedding	nomic-embed-text-v1
VectorDB	Chroma
RAG QA system	Langchain, LangGraph, Ollama, Gemma3
Translation	Gemma3 LLM ,

4. Question-Answering (RAG)

The Retrieval-Augmented Generation (RAG) system supports both single-turn and multi-turn conversational queries using a graph-based approach. The system is designed to simulate chatbot-like interactions with memory and contextual awareness.

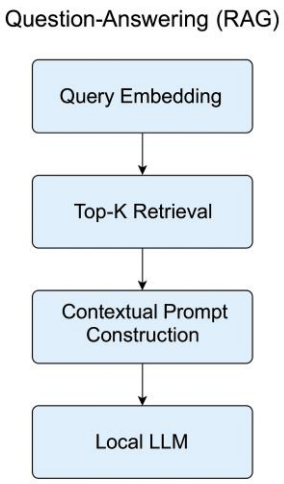
Key Steps:

1. **Query Embedding**
   The user's input query is embedded using the same nomic-ai/nomic-embed-text-v1 model used during the chunk embedding phase. This ensures alignment in semantic space for accurate retrieval.

2. **Top-K Retrieval from Vector Store**
   The query embedding is used to perform a similarity search on the ChromaDB vector store. The top-k most relevant chunks (configurable, e.g., k=5) are retrieved. Each chunk includes metadata like file name, page number, and section, improving traceability.

3. **Contextual Prompt Construction**
   Retrieved chunks are formatted into a prompt with clear boundaries and metadata headers. The system dynamically includes previous user queries and responses for multi-turn conversations. This is achieved through a **graph-based memory module** that maintains conversational flow and context.

4. **Local LLM for Answer Generation**
   The constructed prompt is passed to a local instance of the gemma:3b-it-qat LLM using the Ollama backend. I integrated OllamaChat from LangChain to interact with the model. The LLM generates responses that are context-aware, coherent, and grounded in retrieved knowledge.



5. Translation and Summarization

5.1 Translation

Our system supports multilingual translation with structure preservation and evaluation. The translation workflow is designed to work on structured, token-aware chunked data and includes the following steps:

1. **Input Data**:
   The translation module reads JSON files generated from earlier chunking steps. Each JSON file contains hierarchical chunks (paragraphs, tables, headers) extracted from documents.
2. **Language Detection**:
   Each text chunk is automatically analyzed to detect its source language. If the chunk is already in **English**, it is passed through unchanged. If the chunk is in a **non-English language**, it is first translated to English using the appropriate translation model.
3. **LLM-based Translation**:
   After normalization to English, the chunk is passed to a **local LLM (Gemma3:4b, served via Ollama)** for translation into the target language (e.g., Arabic). This model ensures that the semantic meaning is preserved and adapts the content naturally to the target language.
4. **Structure Preservation**:
   The output translation retains the original format and hierarchy, including headers, paragraphs, lists, and tables, so it can be seamlessly used in downstream applications like document regeneration or localization.
5. **Chunk-Based Processing**:S
   Translation is performed **per chunk**, allowing better memory management and structural context handling. This method also enables localized rollback and caching if needed.
6. **Evaluation: Round-trip Translation + ROUGE**:
   For quality assessment, implement **round-trip translation**, i.e.:
   a. Translate source → target language
   b. Translate target → source language
   c. Compare original and back-translated texts using **ROUGE scores** (ROUGE-1, ROUGE-2, ROUGE-L) to evaluate fidelity and fluency.
   d. This approach helps validate the effectiveness of the translation model and detect semantic drifts or loss of information.

5.2 Summarization

The summarization module is currently under development. The approach involves summarizing individual chunks of text and recursively combining them to generate a coherent overall summary. For evaluation, we plan to use a combination of abstractive and extractive summarization methods to create reference summaries, which will be compared with the model-generated output using ROUGE scores.

6. Performance & Innovation

- **Token-Aware Chunking**: Implements intelligent chunking strategies that respect token limits while preserving contextual coherence, enabling more accurate downstream processing.
- **Table-Aware Extraction**: Extracts tabular data from both PDFs and DOCX files with structure retention, improving information fidelity and usability.
- **Multi-Turn Contextual Q&A**: Supports conversational querying with multi-turn context retention, allowing for more natural and intelligent interactions.

7. Limitations and Future Work

- **Expand File Format Support**: Extend compatibility to additional file types such as .xlsx and .csv to enhance the system's versatility.
- **Advanced Summarization**: Integrate a more sophisticated summarization module to generate accurate and context-aware summaries.
- **Improved PDF Extraction**: PDF files are particularly challenging to extract due to their unstructured and inconsistent formatting. Developing a robust and reliable PDF parsing mechanism is a key area for future improvement.
- **Performance Optimization**: Enhance the efficiency and speed of individual modules to support faster processing and scalability.

-