Dr. X Research Q&A System

OSOS AI Technical Test – Documentation

# Introduction & Objective

The objective of this project is to analyze the mysteriously abandoned publications of Dr. X and build an AI-driven system capable of:

* Reading and processing multi-format documents (.pdf, .docx, .xlsx, etc.)
* Extracting and chunking content intelligently
* Embedding and storing the content in a vector database
* Enabling question answering via a local Retrieval-Augmented Generation (RAG) system
* Translating multilingual content to English/Arabic
* Summarizing content and evaluating output quality

All tasks are performed entirely offline using local models and vector databases.

# File Extraction & Chunking

## File Types Handled

* .pdf: Extracted using PyMuPDF (fitz)
* .docx: Handled with python-docx
* .csv, .xlsx, .xls, .xlsm: Parsed using pandas

## Table Handling

* Tables were converted into Markdown format using the tabulate library, improving readability and model comprehension.

## Tokenization & Chunking

* Tokenizer: cl100k\_base via tiktoken
* Smart sentence-based chunking using NLTK
* Chunk Size: 500 tokens, with 50-token overlap
* Metadata recorded: filename, page number, chunk number, and text

# Embedding & Vector Storage

## Embedding Model

* Used nomic-ai/nomic-embed-text-v1 for generating dense vector embeddings.

## Vector Database

* Used FAISS (faiss-cpu) for offline vector similarity search.

## Stored Metadata

Each chunk stored with:

* Source file
* Page number
* Chunk ID
* Full text

## Performance Logging

* Token count and embedding time tracked
* Avg: ~2300 tokens/sec embedding speed

# RAG Q&A System with LLaMA

## LLM Used

* Model: llama-2-7b.Q4\_K\_M.gguf (via llama-cpp-python)
* Max context: 4096 tokens

## How RAG Works

1. User question is embedded
2. Top-k relevant chunks retrieved from FAISS
3. Prompt constructed with context + question
4. LLaMA generates a grounded response

## CONVERSATIONAL MEMORY

* Previous 1–2 Q&A pairs are injected into the prompt for follow-up support
* All user interactions logged in qna\_history.log for traceability

## Performance

* Prompt tokens: ~800–1000 tokens
* Response time: ~3–6 seconds
* Tokens/sec: ~200–300

## Extras

* Tokens/sec logged in performance.log
* Can answer based on follow-up questions if integrated with conversation memory

# Translation & Summarization

## Translation Model

* Used: facebook/nllb-200-distilled-600M
* Languages supported: 200+
* Target: English and Arabic
* Auto-detects language using langdetect

## Summarization Model

* Used: facebook/bart-large-cnn
* Input: chunks of 500 tokens
* Output: 40–150 word summaries

## ROUGE Evaluation

* Used rouge-score to evaluate summaries.

|  |  |
| --- | --- |
| **Metric** | **Average** |
| ROUGE-1 | 0.48 |
| ROUGE-L | 0.44 |

# Performance Metrics & Innovations

## Tokens/sec Summary

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Avg Tokens** | **Time (s)** | **Tokens/sec** |
| Embedding | ~8500 | ~3.5 | ~2400 |
| Translation | ~10200 | ~6.2 | ~1600 |
| Summarization | ~8900 | ~4.2 | ~2100 |
| RAG QA | ~950 | ~3.5 | ~270 |

## VISUALIZATION

* Created 7\_visualize\_performance.py to generate individual line charts per task.
* Output: PNG graphs of tokens/sec for Embedding, Translation, Summarization, and RAG QA.

## Innovations

* Smart sentence-based chunking using NLTK
* Markdown-formatted tables using tabulate
* Conversation-aware RAG with memory injection
* Q&A traceability with log-based monitoring
* Tokens/sec logging for every step
* Full offline capability without any external API dependency

# CACHING AND DEMO UI

## MODEL CACHING

* All models use cache\_dir=./cache to avoid repeated downloads
* Greatly improves cold start time and ensures offline repeatability

## STREAMLIT DEMO UI

* app.py provides a web-based frontend for the RAG system
* Real-time Q&A interface
* Displays context chunks and generated answers
* Maintains conversation memory across turns
* Logs all Q&A to qna\_history.log

## HOW TO RUN

streamlit run app.py

## Conclusion

The solution successfully fulfills all parts of the OSOS AI Technical Test using robust local NLP pipelines. The RAG system not only uncovers insights into Dr. X’s research but forms the foundation for future offline document analysis systems.