Information Extraction from Coroner’s Reports

Client: Dr. Matthew Albrecht

Group: 18

1. **INTRODUCTION:**
   1. **Project Objectives and Data Description:**

This project focuses on building a self-contained system that can securely process the Coroner’s reports to extract critical insights regarding the causes and impacts of a particular road fatality in Western Australia. These reports comprise disparate, unstructured data on the accident related factors such as medical conditions, socio-economic factors, accident circumstances and road conditions, and alcohol or substance presence. These insights are critically valuable for the WA Centre fir Road Safety Research (WACRSR) to identify and imply preventive measures against surging road transportation fatalities.

Though these reports are typically PDF-based, they are inconsistently formatted, often scanned, including text, tables, graphs, and images. The sensitive nature of the data strictly prohibits the cloud-based processing solutions.   
The manual review process currently employed by the group of researchers like our client is time intensive and prone to human error. Analysing the dynamics, this system has been built that automates the data extraction ensuring local execution to address data security concerns. This scalable system is easy to handle, and operates fast generating critical results within a few seconds.

* 1. **Summary of Approach and Contribution:**

We implemented a modular Retrieval- Augmented Generation (RAG) pipeline that runs entirely on a local machine to meet the project’s objectives. It leverages the power of Large Language Models (LLMs) for natural language understanding, while anchoring their responses in the factual content of the source documents. This grounding mechanism reduces the risk of LLM hallucination.

The pipeline ingests PDF reports and applies OCR when necessary, segments the extracted text into manageable chunks, and stores them in a local vector database. When a query is submitted by a user, the system retrieves the most relevant text segments and provides them as context to the LLM, which then generates a response.

The system is designed to be modular, allowing for the evaluatioin of different LLMs like Llama 3.2, Gemma3, Phi4-mini to assess the trade-offs between performance and computational cost. The entire workflow is orchestrated locally using OLLAMA, ensuring full data privacy and security.

The Contributions include:

i. Modular Python-based system using LangChain for RAG

ii. Ollama for local LLMs (gemma3. Llama3.2, phi4-mini

iii. BERTScore for evaluation

iv. Pre-vectorized JSONL files for efficiency

v. Chat interface for querying

vi. Evaluation Scripts

1. **METHODS AND RESULTS:**

This section details the architectural decisions, implementation specifics, and the dual-pronged evaluation methodology employed in this project. Every choice made in this project is inspired by the unique constraints and the objectives of the project.

* 1. **Reasons for Selecting the Architectural Approach**

Locally deployed Retrieval-Augmented Generation (RAG) pipeline is the core of the project. The approach was chosen over alternatives like LLM fine-tuning or direct querying of a base LLM for various critical reasons grounded in recent NLP research:

1. **Mitigation of Factual Hallucination**: Standard LLMs are very much prone to generate factually incorrect information, known as hallucination. For a domain like Coroner’s report which requires high accuracy, hallucination is an unacceptable risk. The RAG grounds the LLM’s response in explicit evidence retrieved from the source document which forces the model to synthesize answers based on provided text over its parametric knowledge. This technique mitigates the chances of hallucination eventually enhancing factual consistency (Lewis, Perez and Piktus).
2. **Adherence to Security and Privacy Constraints**: Data security and privacy is a non-negotiable requirement of the project due to its sensitive nature. This precluded the use of cloud-based AI services and inspired a self-contained system. The selection of Ollama for local model serving, combined with open-source LLMs and a local in-memory vector database, ensures that sensitive data from the Coroner’s reports never leave the user’s machine (JustDataBuddy).
3. **Enhancing Explainability and Auditability**: Keeping the LLM hallucination in mind, an ability to verify the system generated output was a mutually agreed upon requirement. Where a standard LLM acts as a black box making it difficult to trace the evidence of its claims, RAG implementation provides source attribution for every generated answer, linking the information back to the specific page and text chunk in the original PDF. This doesn’t only help to trace the evidence but also help to re-verify the context and the results significantly enhancing the accuracy and the trust on the system.
4. **Flexibility and Scalability**: A substantial, manually curated dataset of question-answer pairs from Coroner’s reports are required by Fine-tuning LLM . Which wasn’t feasible. RAG offers a more flexible approach, as the knowledge base can be updated simply by adding new documents to the vector store without any model retraining (Gao, Xiong and Jia).
   1. **System Implementation and Experimental Methodology**

The methodology is broken down into the distinct phases of the RAG pipeline, followed by a robust, hybrid evaluation framework.

1. **Data Preprocessing (OCR and Chunking)**

The Coroner’s reports are processed using a preprocessor to extract text via OCR for scanned files, converting to plain text or JSONL. The documents are chunked into ~500- token segments with metadata (page, source) for vectorization. Synthetic data augmentation (paraphrasing chunks) enlarges the pool if needed, but pre-vectorized JSONL files were relied upon for speed.

1. **Vector Embeddings and RAG Integration**

Chunks are embedded using mxbai-embed-large (Ollama) into an inMemoryVectoreStore (Lanchain). Queries are embedded similarly, retrieving top-k (k=3) similar chunks via cosine similarity. Retrieved contexts augment props: “Given {context}, answer {query}/” This ensures factual grounding.

1. **LLM Query System and Response Generation**

Local LLMs generate responses from augmented prompts. ChatPromptTemplate with context, query, and instructions for no prior knowledge outputs response + sources,

1. Evaluation and Visualization

# References

Gao, Yunfan, et al. "Retrieval-Augmented Generation for Large." *arXiv* (2024): 3-12. Document.

JustDataBuddy. *Using Local LLMs for Sensitive Data: A Guide for Professionals*. 14 March 2025. Website. 13 October 2025.

Lewis, Patrick, et al. "Retrieval-Augmented Generation for." *arXiv* (2021): 5-19. Document.