Coroner Project System Design Overview

Adam Holt 21689348

Project

Design, implement, and evaluate a data application for information extraction from coroner's reports using:

- **Optical Character Recognition (OCR)**
- **Pretrained Large Language Models (LLM)**
- **Retrieval Augmented Generation (RAG)**
- **Statistical/Semantic Level Evaluation (BERTScore)**

Requirements

We are required to design and implement a self contained document processing application for the extraction of information related to the conditions and potential causes of road fatalities contained in official coroner's reports for the The Western Australian Centre for Road Safety Research (WACRSR). There are several initial considerations regarding the data and the application requirements.

Firstly, the documents are PDF format which present challenges in automated processing due to the disparate nature and quality of the documents.

Secondly, given the sensitive nature of the data, the application architecture needs to meet specific security requirements. Namely, that all components of the data processing pipeline are able to be run on a local machine with commodity hardware, and that these components are open source; do not rely on proprietary software; and cannot make network calls to any non local API functionality.

Finally, in order to provide useful information to the client, it is necessary that the generated outputs of the document processing application can be evaluated for correctness and relevance.

Data

Manipulating PDFs programmatically is difficult due to the format's inherent complexity and lack of a standardized, machine-readable semantic structure. Specifically, PDFs are not just simple text documents; they can contain a wide array of elements such as text, images, vector graphics, tables, graphs, complex fonts, and interactive forms, each requiring specific handling and adding to the overall complexity of programmatic processing.

Initial analysis of the document data indicates that there are primarily two forms: strucured PDF documents and unstructured PDF documents. The unstructured PDF documents are typically scanned versions of the orignal paper reports. In order to process the unstructured scanned documents the application utilizes an Optical Character Recognition (OCR) module as part of the preprocessor architecture.

Application

Large language models (LLMs) are able to generate human like responses to queries but due to probabilistic inference and reliance on static training data, this can lead to incorrect, incomplete, and/or irrelevant answers. To mitigate these problems a Retrieval Augmented Generation (RAG) model can be utilized. The RAG structure supplies contexutally relevant information from external data sources (i.e., the coroner's reports).

For our application we have designed a data processing pipeline which incorporates a document preprocessor to extract structured data for conversion into a machine-processable format which is then used as a data store for a RAG LLM model. The model can then be queried for specific information contained in the coroner's reports. Utilising a RAG system in conjuction with an LLM enables query outputs to be contextually relevant, factually accurate, and linked with source information directly from the documents.

Process & Application Architecture

The data processing pipeline can be broken down into discrete units of functionality. Each unit of functionality resides in one of three architecture components:

- **Data Preprocessing**
- **RAG**
- **Evaluation**

The units of functionality encapsulated within each architecture component are as follows:

Data Preprocessing

Data Preprocesser

The PDF documents are input to a preprocessor for conversion to a machine-processable format (i.e., JSON). The preprocessor extracts textual information from the unstructured inputs via an OCR module.

Vector Embeddings

The machine-processable format output from the preprocessor can then be serialized and chunked in order to be transformed into vector representations which will be the source of a vector database used for RAG.

RAG

Query

User queries are projected into the vector embedding space in order to determine the correct context for retreival of information from the document that is most relevant to the query. The retriever component of the pipeline must perform a search within the vector database to identify this context.

Context Augmentation

The pretrained LLM component of the pipeline has a fixed capacity for generating responses that are based on the training data. The information from the retriever component is used as an additional layer of

context to augment the LLM query on demand.

Response Generation

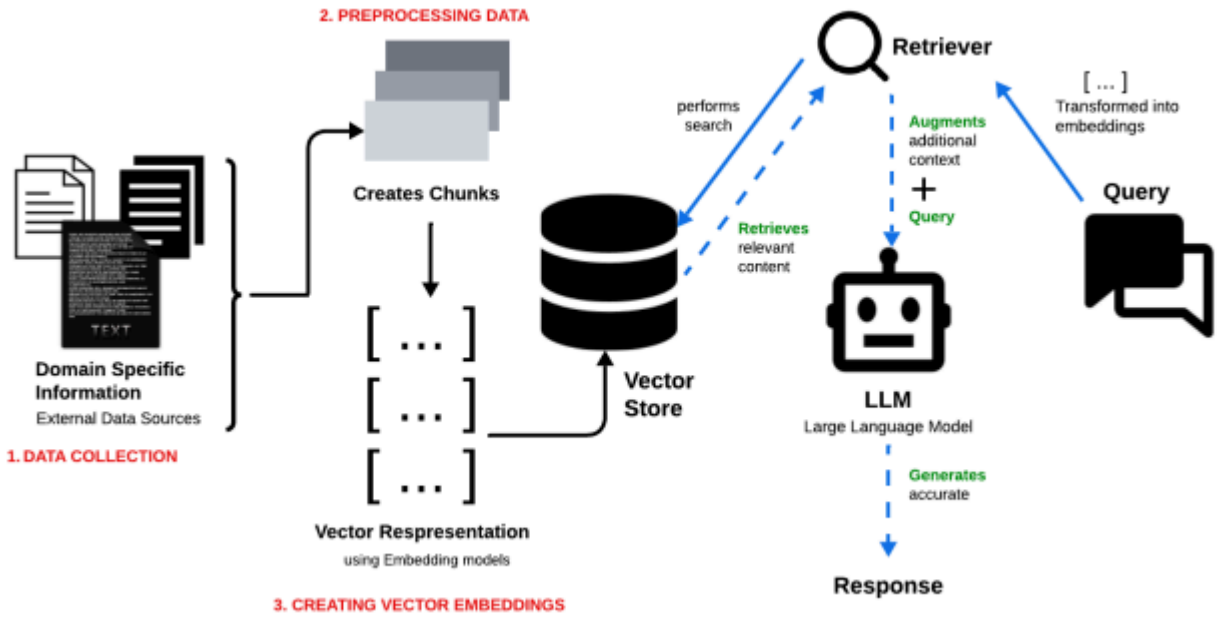
The context-augmented prompt, consisting of the original user query combined with the contextually relevant content is provided to the LLM so that it is able to produce a response that coheres with the original document and has a high accuracy due to the alignment with the vector database. The appropriate metadata consisting of the page number of the original PDF document which the text content was taken, and the name of source document is displayed in the response in order for the user to analyse and cross reference for correctness and relevance.

Evaluation

The output of the query (i.e., the *response*) should provide an accurate representation of the actual contents of the document in conjuction with metadata that enables an audit of the response. An evaluation is performed on these outputs with a statistical semantic similarity *score* (i.e., the BERTScore).

Architecture

Below, is an *example* illustration of the application architecture from Khan & Hasan.



Implemetation & Model Specifics

Data Preprocessing/Extraction

The Docling Project was chosen for the following reasons:

- simple modular interface
- open source
- locally executable

OCR

Various PDF processing OCR modules were tested:

- pdf2image (high memory usage)
- tesseract (installation/configuration difficulties)
- EasyOCR (easily configured with Docling; good balance of speed and output quality)

The chosen OCR module was EasyOCR.

Vector Database

Various vector databases were considered:

- Milvus (designed for very large scale applications)
- Chroma (security concerns, i.e., telemetry)
- Local in-memory implementation

The approach take for vector database implementation was to use a local in memory vector database. The benefit being, local execution coheres with strict security requirements. The tradeoff being, memory constraints of local hardware and limited to cosine similarity search.

Pretrained LLMs

For local hosting of pretrained LLM models we utilized Ollama due to being open source and highly configurable.

The chosen pretrained models as follows:

- llama3.2 (3.2 Billion parameters; 2.0 GB)
- phi4-mini (3.8 Billion parameters; 2.5 GB)
- Gemma3 (4.3 Billion parameters; 3.3 GB)

Implemention API

LangChain was chosen as the main implementation API due to it's modularity and extensibility. Furthermore, the main components are strictly open source and there exist a wide variety of community extensions.

Sources

1. Retrieval-Augmented Generation for Large Language Models: A Survey
2. Developing Retrieval Augmented Generation (RAG) based LLM Systems from PDFs: An Experience Report
3. Docling Technical Report