Information Extraction from Coroner's Reports with OCR, RAG, and LLM's

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. There are several initial considerations regarding the data and the application requirements.

Firstly, the documents are typically in PDF format which present challenges in automated processing due to the disparate nature 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 consitency and correctness.

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 will require an Optical Character Recognition (OCR) module.

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 leading to incorrect or incomplete answers. To mitigate these problems a Retrieval Augmented Generation (RAG) model can be utilized which is able to pull in information from external data sources. For our application we propose 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 will enable 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. Namely:

1. Data Preprocessing

The PDF documents are input to a preprocessor for conversion to a machine-processable format, eg., JSON, markdown, plain text etc. The preprocessor will be required to handle both structured documents and unstructured documents (i.e., scanned PDFs). For the unstructured inputs an OCR model will be necessary. With the data in a structured format such as plain text, synthetic data may be generated to enlarge the data pool.

2. 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.

3. Content Retrieval

Queries must be transformed into the same format as the vector embeddings 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.

4. 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.

5. Response Generation

The context-infused prompt, consisting of the original user query combined with the retrieved relevant content is provided to the LLM so that it is able to produce a response that coheres with the original documents and has a high accuracy due to the alignment with the vector database.

6. Output

The output of the query should provide an accurate representation of the actual contents of the documents in conjuction with metadata that enables an audit of the response, eg., the source of the resulting outputs.

Architecture

RAG LLM architecture

RAG LLM architecture

Technology/Model Specifics

Much of the functionality and components of the document processing pipeline architecture are available in the open source Docling Project. Specifically, the Docling model provides the following components relevant to our project:

- Multiple document input formats
- OCR support for scanned PDFs and images
- Various export formats and options, including Markdown, HTML, DocTags and lossless JSON
- Local execution capabilities for sensitive data and air-gapped environments
- Integrations with LangChain, LlamaIndex, etc (i.e., LLMs with RAG capabilities)

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
4. What Is Docling? Transforming Unstructured Data for RAG and AI