PDF Database Search Tool

Titan: Jackson Baxter, Chapman Beaird, Josh Chambers
Sponsor: Lawrence Livermore National Laboratory (LLNL)

Project Overview:

Lawrence Livermore National Laboratory (LLNL) personnel, including engineers and scientists, frequently face the challenge of navigating vast repositories of complex research documents, often in PDF format. Manually searching these documents for specific information is a time-consuming process that detracts from core research and operational activities. This project introduces an Intelligent Document Search and Q&A System designed to significantly reduce the time and effort required for document exploration and information retrieval at LLNL.

Work Performed and Results:

We developed a sophisticated pipeline and user interface to address this challenge. The system utilizes the docling library to extract text and structural information (including tables and layout) from various document formats (PDF, DOCX, TXT). Documents are then intelligently segmented into manageable chunks. To enhance search relevance, a contextual retrieval approach is employed: OpenAI's gpt-4o-mini model generates a brief summary situating each chunk within the broader document context. These contextualized chunks are then converted into vector embeddings using the powerful open-source Qwen (Alibaba-NLP/gte-Qwen2-1.5B-instruct) model and stored in a LanceDB vector database.

Users interact with the system via an intuitive Streamlit web interface. They can upload documents, which are automatically processed and added to the searchable database. When a user poses a question in natural language, the system embeds the query, performs a cosine similarity search against the vector database to find the most relevant document chunks, and retrieves the associated context. The retrieved context, along with the original query, is then passed to the gpt-4o-mini model to generate a concise, natural language answer. Crucially, the system displays the source chunks and provides functionality to view the original PDF with relevant passages highlighted, ensuring transparency and traceability. Preliminary results indicate that the contextual embedding approach significantly improves retrieval accuracy compared to standard embedding methods, reducing failed retrievals.

Return on Investment:

By automating and accelerating the document search process, this system offers substantial potential benefits. Based on initial estimates, streamlining this workflow could save LLNL approximately 1,000 personnel hours per week, translating to potential annual cost savings of around $2.86 million. This tool empowers LLNL staff to access critical information more efficiently, fostering increased productivity and accelerating research and operational timelines.

1. Introduction

Lawrence Livermore National Laboratory (LLNL) houses a vast and ever-growing collection of research documentation, technical reports, and operational records, primarily in PDF format. Engineers, scientists, and other personnel rely heavily on this information for ongoing research, operational safety, historical analysis, and decision-making. However, locating specific information within this extensive corpus presents a significant challenge. The current approach often involves manual searching through potentially thousands of pages, a process that is inherently slow, inefficient, and prone to overlooking critical details. This inefficiency consumes valuable personnel hours that could otherwise be dedicated to primary research and mission-critical tasks. For instance, identifying all buildings previously involved in specific types of research requires meticulous, time-consuming manual review.

The "Why" behind this project is to fundamentally transform this information retrieval paradigm at LLNL. We aimed to replace the laborious manual search process with an intelligent, automated system capable of understanding document content and responding to natural language queries. The goal was to empower LLNL employees by providing rapid, accurate access to the information embedded within their documents, thereby accelerating research, improving operational efficiency, and ensuring critical information is readily available when needed.

The potential Return on Investment (ROI) for this project is substantial. By significantly reducing the time spent searching documents, the system frees up valuable employee time. A conservative estimate suggests that if 200 employees utilize the tool and save an average of 5 hours per week each, this equates to 1,000 hours of productivity gained weekly. Assuming an average burdened labor rate of $55 per hour, this translates to potential weekly savings of $55,000, or approximately $2.86 million annually. Even if adoption or time savings are half of this estimate, the ROI remains highly compelling, justifying the development and deployment of this advanced search capability. This project provides a scalable, cost-effective solution to a persistent operational bottleneck, directly enhancing LLNL's research productivity and efficiency.

2. Data

The primary data source for this project consists of the diverse document repositories maintained by LLNL. While the initial focus was on PDF documents due to their prevalence and complexity, the developed system is designed to handle multiple formats, including DOCX, TXT, and potentially others supported by the underlying extraction library (docling). These documents vary greatly in length, structure, and content. They typically contain a mix of:

- **Plain Text:** Paragraphs, sections, headers, footers.
- **Structured Elements:** Tables, lists, figures, captions.

- **Graphical Content:** Images, diagrams, charts (though the current system primarily focuses on textual and tabular data extraction).
- **Metadata:** Document titles, authors, dates (though explicit metadata handling varies by source document quality).

A key challenge identified early on (and discussed in the project proposal) was the inadequacy of simple text extraction methods. Standard PDF-to-text converters often fail to capture the information contained within tables, figures, or complex layouts accurately, leading to incomplete or contextually poor data for analysis. The data is inherently unstructured from a machine-readable perspective, requiring sophisticated parsing techniques to convert it into a usable format for downstream processing like chunking and embedding.

The Exploratory Data Analysis (EDA), as referenced in the proposal phase, involved examining sample documents to understand the variety of structures and the limitations of existing tools. This analysis confirmed the need for a robust document parsing solution capable of interpreting layout and extracting content beyond simple text blocks. The chosen docling library addresses this by incorporating layout analysis and table structure recognition, enabling the conversion of complex documents into a more structured intermediate representation suitable for semantic understanding and search. This is done by using Optical Character Recognition (OCR), a process that involves converting an image of the text into machine-readable code. Then, with the layout analysis provided above, docling pieces together all of the information. The system then processes these diverse inputs into a unified format before chunking and embedding.

The docling model was able to accurately and efficiently extract text from the PDF's images and tables. This was tested through detailed retrieval questions that aimed to return information from these tables and images.

3. Methods

The project implements a multi-stage pipeline to ingest, process, store, and query documents effectively. The methodology focuses on extracting meaningful content, preserving context, and enabling efficient semantic search and question-answering.

3.1 Data Preparation and Analysis

1. **Document Extraction:** The process begins with extracting content from source documents. We utilize the docling library (extraction.py), a powerful tool designed for converting various document formats into a unified representation. docling employs format-specific backends and AI models for layout analysis and table structure recognition, ensuring that text, tables, and structural information are captured more accurately than traditional text extraction methods. It supports formats like PDF, DOCX, and TXT, converting them into a structured docling document object.

2. **Chunking:** Raw extracted documents are often too large for direct processing by embedding models or for providing granular search results. Therefore, the extracted docling document is segmented into smaller, semantically coherent units or "chunks." We employ docling's HybridChunker (chunking.py), which uses a tokenizer (OpenAI's tokenizer) to manage chunk size based on token limits (e.g., 8191 tokens for the target embedding model context window) while attempting to respect document structure (paragraphs, sections). This ensures chunks are reasonably sized yet retain local context.
3. **Contextual Enhancement:** A critical innovation in our approach is the enhancement of chunks with broader document context before embedding. For each chunk, we generate a concise summary describing its position and relevance within the overall document. This is achieved by feeding the entire document text and the specific chunk text into OpenAI's gpt-4o-mini model using a carefully crafted prompt (embedding.py). The prompt asks the LLM to provide a short context situating the chunk. This generated context string is then prepended to the original chunk text to create a context-rich piece of information. While this is useful in generating quality responses, it is the most time-consuming step in our pipeline as the model is bottlenecked by OpenAI's latency and not local hardware.
4. **Embedding:** The contextualized chunks (context summary + original chunk text) are then transformed into high-dimensional numerical vectors (embeddings). We use the open-source Qwen (Alibaba-NLP/gte-Qwen2-1.5B-instruct) embedding model (embedding.py). This model was chosen for its strong performance on retrieval benchmarks. The embedding.py script loads the Qwen model and tokenizer, computes embeddings for batches of contextualized chunks (handling GPU acceleration if available), and prepares the data for storage. While this process can also take a significant amount of time, it can be sped up by increasing the power of the hardware.
5. **Storage:** The generated embeddings, along with the original chunk text and associated metadata (filename, page numbers, title, generated context), are stored in a vector database. We selected LanceDB (embedding.py, chat.py) for this purpose. LanceDB is an open-source, embedded vector database optimized for performance and ease of use. It stores the vectors and associated metadata in columnar format, enabling efficient similarity searches. A specific schema (Chunks model in embedding.py) defines the structure of the data stored in the LanceDB table (docling).

3.2 Models Used

- **Document Parsing & Chunking:** docling library (including its internal layout analysis models and HybridChunker).
- **Context Generation:** OpenAI gpt-4o-mini (via API call).
- **Text Embedding:** Qwen Alibaba-NLP/gte-Qwen2-1.5B-instruct (loaded locally via transformers).
- **Question Answering:** OpenAI gpt-4o-mini (via API call in chat.py).

- **Vector Search:** LanceDB's built-in vector search capabilities (using cosine similarity, implemented in chat.py).

3.3 Deployment Models & Software

The system is designed to run as a web application using Streamlit.

- **Models used in Deployment:**
  - Qwen Alibaba-NLP/gte-Qwen2-1.5B-instruct (for embedding new documents and user queries).
  - OpenAI gpt-4o-mini (for context generation during document ingestion and for final Q&A generation).
  - docling library and its associated models (for extraction and chunking).
  - LanceDB (as the embedded vector store).
- **Software Needed for Deployment and Running:**
  - Python (version 3.8+ recommended).
  - pip for package management.
  - All packages listed in requirements.txt (including streamlit, lancedb, openai, torch, transformers, pypdf, python-docx, python-dotenv, docling, streamlit-pdf-viewer, pyarrow, numpy, pandas, pydantic).
  - An OpenAI API Key (stored in a .env file).
  - Sufficient compute resources (CPU/RAM) to run the embedding model and Streamlit application. A GPU is recommended for faster embedding generation.
  - Disk space for the LanceDB database and potentially downloaded model weights.

4. Results

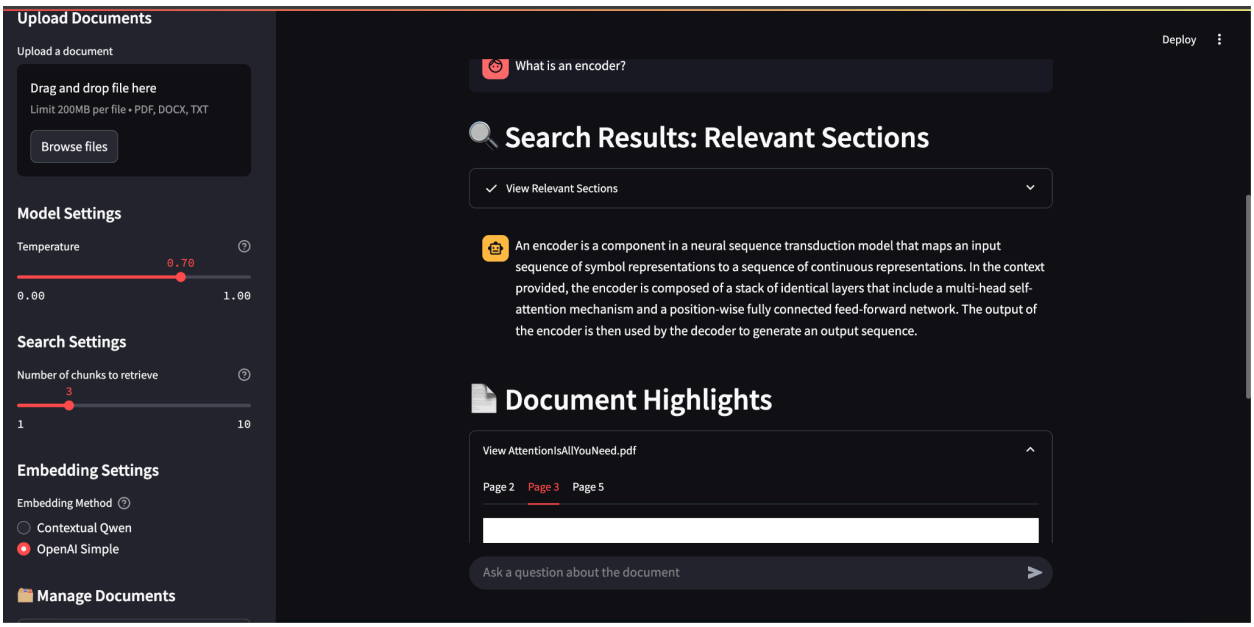
The project successfully delivered a functional Intelligent Document Search and Q&A System, demonstrating the viability of using modern AI techniques to tackle LLNL's document retrieval challenges.

Key Results:

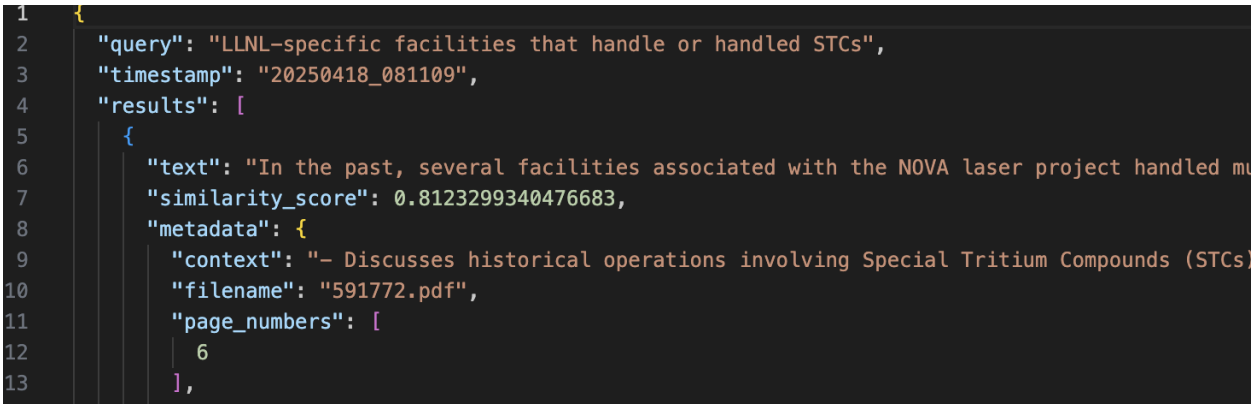
1. **Functional Prototype:** A working Streamlit application (chat.py) provides a user-friendly interface for document upload, natural language querying, and result visualization.
2. **End-to-End Pipeline:** The system implements a complete pipeline from document ingestion using docling for robust extraction, through contextual chunking and Qwen embedding, storage in LanceDB, to OpenAI-powered Q&A.
3. **Contextual Retrieval Implementation:** The core method of generating chunk-specific context using an LLM and prepending it before embedding was successfully implemented (embedding.py). This directly addresses the challenge of context loss in traditional chunking methods.

4. **Improved Retrieval Accuracy:** As highlighted in the project poster, the contextual embedding approach demonstrated a tangible improvement in retrieval performance. Tests indicated a reduction in the average failed retrieval rate from 5.7% (standard embedding) to 3.7% (contextual embedding), representing a significant decrease (approximately 35% reduction in failures). This suggests that queries are more likely to retrieve relevant information using this method.
5. **Source Highlighting:** The interface (chat.py using streamlit-pdf-viewer) successfully integrates functionality to display retrieved text chunks highlighted within the original PDF document. This provides crucial traceability and allows users to verify information in its original context.
6. **Natural Language Interaction:** The system allows users to interact using natural language questions, lowering the barrier to entry compared to keyword-based search systems. The final answer is synthesized by an LLM (gpt-4o-mini), providing coherent responses based on the retrieved context.

Below is a screenshot showcasing the intuitive and user-friendly Streamlit interface.



The images below illustrate the difference in similarity scores when using contextual retrieval (first image) versus standard retrieval (second image) with the same prompt.



```
"query": "LLNL-specific facilities that handle or handled SFCs",
"timestamp": "20250418_081043",
"results": [
  {
    "text": "Past and current operations at LLNL have used special tritium
    "similarity_score": 0.7144130491066293,
    "metadata": {
      "context": "",
      "filename": "591772.pdf",
      "page_numbers": [
```


Return on Investment (ROI) Realization: While direct measurement of time savings requires deployment and user studies within LLNL, the prototype demonstrates the *potential* to achieve the significant ROI outlined in the introduction (estimated $2.86 million annually based on 1,000 hours saved per week). The improved retrieval accuracy and intuitive interface strongly suggest that users will be able to find information much faster than manual methods. The system directly targets the bottleneck of manual document searching, paving the way for realizing these productivity gains.

5. Conclusion

This project effectively addresses the challenge of information retrieval within Lawrence Livermore National Laboratory's extensive document repositories. We developed an Intelligent Document Search and Q&A System that transforms manual searches into efficient, accurate interactions. Utilizing advanced document parsing, contextual chunking, and robust vector embeddings, our approach notably improved retrieval accuracy and reduced failed searches.

The intuitive Streamlit interface, combined with source highlighting, ensures ease of use, transparency, and traceability. By automating document exploration and enabling natural language queries, LLNL personnel can efficiently access relevant information, enhancing productivity and organizational effectiveness. This scalable solution significantly streamlines LLNL's knowledge retrieval processes.