Project Title

DocuMind: Intelligent Document Analysis and Question-Answering System

Proposed Cluster

- AIML
- BD
- ANLP

Team Members

- Sachita Aryal (079BCT0)
- Sujal Pandit (079BCT084)
- Sujan Bhattarai (079BCT086)

Project Overview

In the digital world we live in today, vast amounts of information whether reports, research papers, or policy files are stored and accessed in the form of documents such as PDFs, DOCX, and other computer-based text formats. Although these documents contain valuable knowledge, extracting relevant information efficiently remains a significant challenge. Traditional search methods rely on keyword matching and lack contextual understanding, making extraction time-consuming and inefficient for large document collections. This problem affects everyone who deals with numerous documents frequently, from students and researchers to professionals and organizations that need quick and accurate access to information without reading documents end-to-end.

The proposed system, DocuMind: Intelligent Document Analysis and Question-Answering System, addresses this challenge by developing a cross-platform desktop application. Users can load a single document or a collection of documents and interact with them using natural language queries. The system retrieves relevant information from loaded documents and generates context-aware answers with source citations. All document processing and embedding generation occur locally, ensuring user documents remain private and secure, while answer generation leverages cloud-based inference for optimal performance on standard hardware.

DocuMind employs pre-trained transformer models and Retrieval-Augmented Generation (RAG) to combine semantic document retrieval with answer generation. A vector-based database stores document embeddings locally, enabling efficient retrieval from hundreds of documents if required, while precomputing embeddings allows a CPU-only system to operate effectively.

The project focuses on building a working prototype of an intelligent document question-answering system. The primary emphasis is on understanding and applying the concepts of NLP, embedding-based retrieval, and AI-driven generation, rather than achieving state-of-the-art performance. The system serves as a learning tool to explore AI/ML workflows, experiment with RAG pipelines, and develop software engineering skills, providing practical insights into building real-world AI applications.

Motivation

As students, we spend a significant amount of time reading and searching through large volumes of reports, unstructured text, and research papers to find specific information. This process is inefficient, time-consuming, and often mentally exhausting, especially when the documents are very large or complex. Important details are frequently buried deep inside pages of text, making simple keyword search insufficient and forcing repeated manual reading.

This creates a clear need for an intelligent system that can understand documents and provide precise, context-aware interaction to reduce information overload and improve productivity. By building an intelligent document question-answering prototype, we aim to explore how modern NLP techniques and retrieval-based AI systems can transform static documents into more accessible knowledge sources, while gaining practical experience in real-world AI application development.

Implementation Plan

Phase 1: Project Setup and Environment Configuration

- Set up development environment and project structure
- Install required dependencies and packages
- Project structurization and Git configuration

Phase 2: Desktop Application UI and PDF Viewer Setup

- Build the main application interface
- Implement functional PDF viewer with basic navigation functionalities
- Create document management system
- Testing and validation of the application

Phase 3: Document Processing Pipeline

- Extract text from uploaded documents
- Clean and preprocess extracted text
- Split text into manageable chunks with metadata
- Create processing pipeline
- Test and validate for various document types

Phase 4: Embedding Generation and Vector Database

- Generate vector embeddings for document chunks
- Set up local vector database
- Implement similarity search
- Optimize and test database operations

Phase 5: Query Processing and Retrieval Module

- Build query processing module
- Implement retrieval mechanism to perform similarity search in vector database and retrieve relevant chunks
- Develop ranking and filtering
- Context preparation and implementation of retrieval evaluation
- Testing and validation for various edge cases

Phase 6: Answer Generation with LLM Integration

- Integrate an LLM via API call for answer generation
- Design effective prompts for context-aware answers
- Implementation of source citation mechanism
- Handle conversation history for follow-up questions
- Testing and optimization of response quality

Phase 7: Full System Integration and Chat Interface

- Connect all modules into a cohesive system
- Complete chat interface implementation
- Implement end-to-end Q&A workflows
- Final UI polish and features as needed

Phase 8: Testing, Optimization and Documentation

- End-to-end testing of all features
- Performance optimization
- Complete documentation
- Preparation for demonstration

Use Cases

Application areas & scope of the proposed project: The system is targeted to be applicable where dealing with large volumes of documents is required.

1. **Students and Researchers:**
   - Quickly query textbooks, lecture notes, and research papers for the required information without needing to read end-to-end.
2. **Legal Professionals:**
   - Analyze contracts, agreements, or case files efficiently.
   - Identify specific clauses or compare terms across multiple documents.
3. **Healthcare Professionals:**
   - Query medical literature and guidelines efficiently.
   - Extract relevant information from patient reports and research papers.
   - Compare procedure documentation and clinical protocols.
4. **Technical Teams:**
   - Search through product manuals or troubleshooting guides.
   - Reduce time to resolve support tickets by retrieving precise information.

Resources Required

Computing resources and hardware requirements:

- **Personal computer:** Minimum 8 GB RAM and multicore CPU
- **Local storage:** 20-30 GB for documents, embeddings, and application data
- **Vector database:** Local lightweight database to store document embeddings
- **NLP frameworks:** Pre-trained transformer models (CPU compatible)
- **LLM API:** Groq API (free tier) for answer generation
- **Internet connection:** Required for answer generation via API
  While answer generation requires internet connectivity, all document processing, text extraction, embedding generation, and retrieval operations are performed locally.
- **Development Tools:** Python, IDE (VS Code, PyCharm), Git
- **Optional GPU:** For faster embedding generation and model inference
- **References:** Relevant research papers, official framework documentation, Hugging Face guides, and reputable online tutorials