A Tailored AI Framework for Retrieval and Interpretation of Legal Case Documents: Enhancing Information Access Using Hybrid Retrieval Techniques

S. Mary Rosaline[1], C. Naveen[2], Azaria. J. Wesley[2], K. Athiganesan[2], G. Karuppusamy[2]
[1]*Assistant Professor, Computer Science and Engineering, Tamilnadu College of Engineering, Coimbatore, India*
[2]*UG Students, Computer Science and Engineering, Tamilnadu College of Engineering, Coimbatore, India*

Abstract: Invocation and interpretation information from unstructured legal documents such as case files and judgments remain a complex task due to different formats and domain-specific languages. This article proposes a TaylorMade KI framework specifically developed for the legal realm. The system combines discussion of vector-based call methods and knowledge graph control, improved by multi-document analysis and metadata extraction. This hybrid approach ensures accurate context-related results and maintains traceability and reliability of the source. With quick and commentary responses in several legal documents, this frame is suitable for supporting legal investigations, fall comparisons, and judicial reviews.

Keywords: document processing, unstructured data analysis, large-scale language models (LLM), vector decoration, knowledge graphics, semantic search, hybrid calls, metadata extraction.

I. INTRODUCTION

In modern legal ecosystems, large amounts of information are embedded in unstructured forms such as court reasons, legal known, and court files. In contrast to structured databases, these documents lack a consistent scheme, making it difficult to efficiently access accurate and meaningful findings. Extracting relevant information from such complex and unstructured legal data is extremely important for legal practitioners, researchers and political decision makers to make appropriate decisions. The proposed system integrates a hybrid call method that combines semantic performance of vector adultization with structured thinking in knowledge graphs. It also includes multi-document analysis for comparative findings and metadata extraction, ensuring source traceability and context-related reliability. This approach not only improves information accessibility, but also supports deeper legal analysis. This means that the systems used in legal research, documentation and case preparation workflows are extremely important.

II. LITERATURE REVIEW

[1] Further development of large-scale language models (LLMS) has had a significant impact on the way organizations process unstructured data. The paper, entitled "Designing an Unstructured Analytics System Operated by LLM," provides Aryn, a framework for automating document analysis. Sycamore and Luna, distributed engines for document processing, show two core components that convert natural language queries into executable tasks. Although this study focuses on accident reporting, the methods can be adapted to legal settings. This setting requires scalable and intelligent processing of tasks such as summaries and entity extraction in long case files. A search for "Looked Generation of Large-scale Language Models" investigation examines the Generation of Re-Articles (RAG) as a potential solution.

[2] By including relevant external data during the creation of the reaction, RAG improves the consistency and accuracy of the generated content. This paper discusses different types of claims and valuation strategies, highlighting their value in areas such as the law that the current information is inherently important to legal interpretation. PDFS: Development of LLM Systems (Call Off Deduction -

Elevator Generation) from Experience Report provides insight into establishing a system that uses PDF documents as a dynamic knowledge base. Learn about important processes such as text extraction, document indexing, and access optimization.

[3] This study compares tools such as the OpenAI's Assistant API with open-source models such as Lama. These methods provide accurate and query-specific answers that arise directly from the case file. A study titled "From Local to Global: Graph Rag Approach to Query-focused Summary" covers the introduction of Graph Rag. This is how you can use knowledge graphs to gain relational importance throughout the document. Create structured graphics from the document corpus and generate overviews at the group level. This technique is particularly effective in legal research that is important for pursuing legal, parties and trap decisions. "Hybrid rag: Avoyel Generation for Integrated Knowledge Graphs and Vectors for Efficient Information Extraction" illustrates a hybrid framework that combines Graph Rag and Vector rag technology. This two-layer approach allows for both structured relationship extraction and flexible semantic calls. This method was initially tested on financial documents, showing promising results in domain-specific document analysis and can be applied to legal files to retrieve and improve information.

III. PROPOSED METHOD

The suggested system aims to aid in the analysis of unstructured legal documents through Retrieval Augmented Generation (RAG). By combining semantic vector search with the reasoning abilities of knowledge graphs, this system guarantees that users receive responses that are both meaningful and contextually relevant when they submit queries.

A. Document Processing

The initial stage involves handling the raw unstructured documents. This is achieved by extracting text from each file and running two processes simultaneously to prepare the data for retrieval.

Vector embedding generation: The extracted text is divided into smaller segments, referred to as "chunks," to fit the input size limit of the embedding model. Each chunk is transformed into a vector using a sentence embedding model like all-MiniLM-L6-v2. These vectors encapsulate the semantic essence of the text, thus enabling the system to execute similarity-based searches later on.

Knowledge graph construction: Concurrently, the system employs a language model to pinpoint key entities within the text, such as names, places, or organizations. It then determines the connections among these entities, creating relationships like "Person A → works at → Company B." This interconnected data is stored in a structured format known as a knowledge graph, facilitating relational reasoning during retrieval.

B. Hybrid Retrieval Engine

The system features a retrieval engine that operates two distinct methods in parallel. This hybrid strategy enhances both the accuracy and relevance of the search results.

Vector-based semantic search: When a user inputs a query, it is converted into a vector that is then compared to the document vectors using cosine similarity. This method allows the system to uncover the most semantically related segments of the documents, even when there is no direct match of keywords.

Knowledge graph querying: Simultaneously, the system analyzes the entities present in the user query and explores the knowledge graph for all associated relationships. This process enables a deeper understanding of the content, revealing connections not immediately visible from the text itself.

The outputs from both the vector search and knowledge graph query are integrated and ranked according to relevance. This ensures that the response is thorough, precise, and well-supported by the original content.

C. Multi-Document Analysis

The system is adept at processing multiple documents simultaneously. When a user selects several documents for querying, a separate hybrid retrieval engine is established for each file. The results from all these engines are then consolidated using a technique called Reciprocal Rank Fusion, which provides a balanced and relevant output from every source. This arrangement permits the system to deliver both high-level summaries and in-depth comparisons among documents.

D. Metadata Integration

To assist users in tracing the origin of each piece of information, the system gathers metadata during document processing. Information such as the file name and page number is recorded and later incorporated into the generated response. This feature is particularly beneficial in legal and academic contexts, where validating the original source is crucial.

E. Response Generation

The system utilizes a tree-structured summarization method to craft final responses. This approach systematically merges related segments of retrieved content and summarizes them incrementally. As a result, the final output is organized, succinct, and logically consistent, avoiding unnecessary redundancy while providing users with a clear and accessible summary of essential information.

F. Justification for Hybrid Approach

This strategy leverages the complementary strengths of the two retrieval methods:

Semantic understanding: Vector embeddings enable the system to grasp the user's query meaning and identify relevant content, even when different terminology is employed.

Logical structure: The knowledge graph maintains the entity relationships that might be lost when text is fragmented into chunks. This feature allows the system to offer responses that are logically coherent based on the document's structure.

By integrating both methods, the system attains a more comprehensive understanding of the document content and delivers responses that are contextually rich and reliable.

IV. SYSTEM ARCHITECTURE

The system brings together multiple components to process documents and respond accurately to user queries. This end-to-end flow is illustrated in Fig. 1, which shows how the different parts of the system work together.

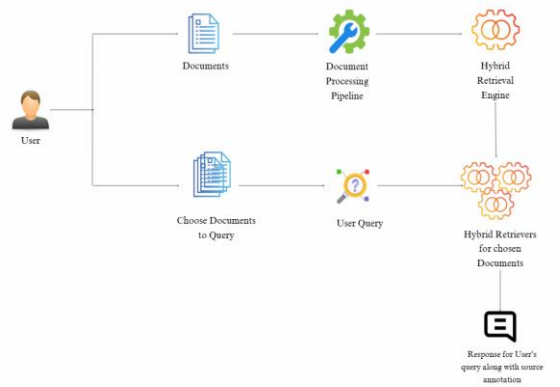

Fig. 1 System architecture

As shown, the entire process begins with the user uploading unstructured documents. These documents go through the document processing pipeline, which is explained in more detail in Fig. 2.

When documents are uploaded, the system extracts the text from them and splits the content into smaller segments, also called chunks. These chunks make it easier to handle token size limitations during processing.

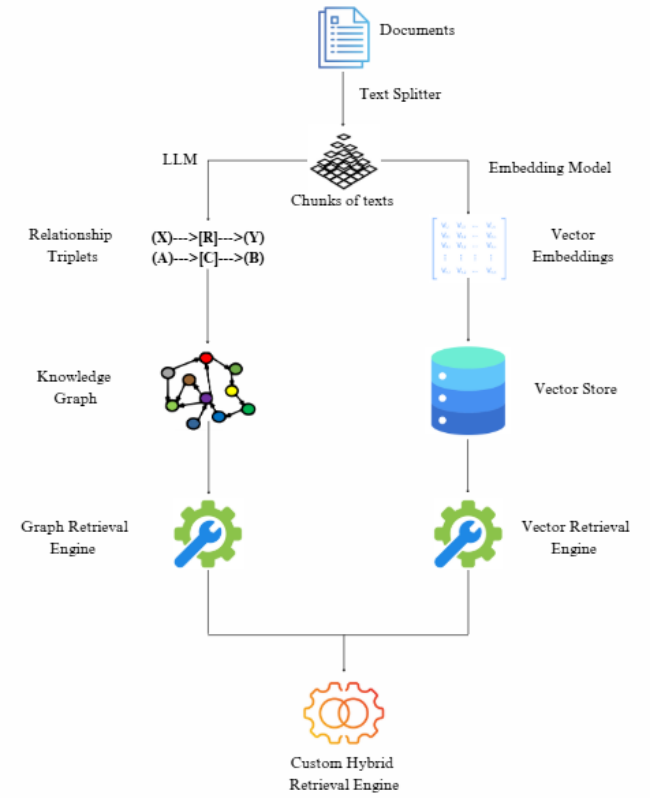

Fig. 2 Document Processing Pipeline

Each chunk is then processed in two ways, both happening at the same time. One process uses an embedding model to convert the text chunks into semantic vector embeddings. The other process uses a language model to identify entities and their relationships, which are converted into triplets like "Entity A → relates to → Entity B."

The generated vector embeddings are stored in a vector database, while the extracted triplets are used to build a knowledge graph, which is stored in a graph database. In addition to this, metadata like document title and page number is also extracted and saved to enable traceability in the final response.

For each document, a hybrid retrieval engine is created by combining both the semantic search and the knowledge graph search capabilities.

Once the user enters a query, the system follows the steps shown in Fig. 3, which outlines the query processing pipeline.

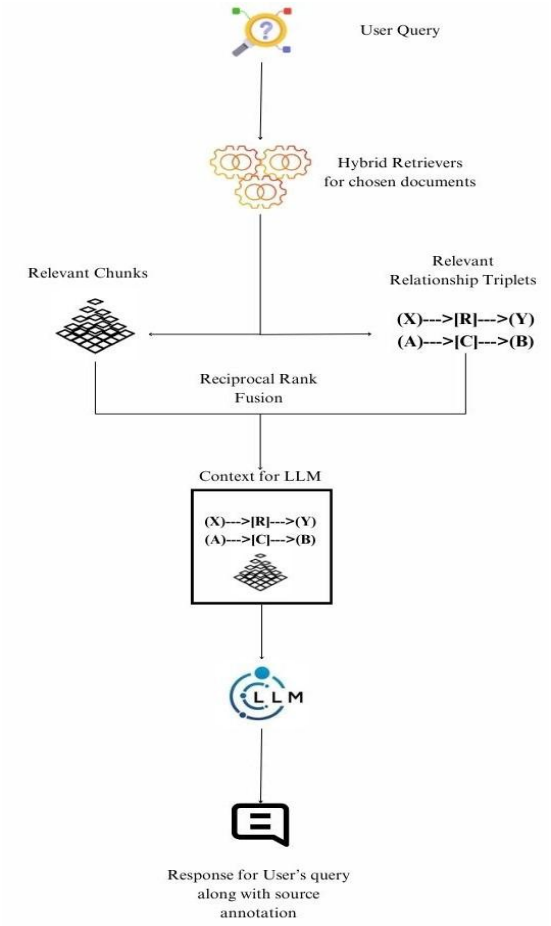

Fig. 3 Query Processing Pipeline

The query is processed by the hybrid retrieval engine, which fetches relevant chunks of text and related graph entities. If the query refers to more than one document, the system combines the context gathered from all selected documents. This combined context is then ranked based on relevance before it is used for response generation.

This layered design ensures that the system not only retrieves accurate and meaningful information but also presents it with traceable sources, making it particularly useful for legal or research-based applications.

V. IMPLEMENTATION

A. Document Processing

1)Content and Metadata Extraction: The system offers an easy-to-use interface for users to upload legal case documents. A document parser, such as LlamaParse, is used to extract the textual content. In addition to the main text, important metadata like the document title and page numbers are also gathered, giving essential context to the legal information.

2)Text Chunking: To manage extensive legal texts, the document is split into smaller, more manageable sections with the Token Text Splitter from LlamaIndex. This approach guarantees that the textual material aligns with the context limits of the models used in the system.

3)Vector Embedding Generation: Next, each text segment is transformed into vector embeddings using a pre-trained model like all-MiniLM-L6-v2. This process converts the textual information into numerical formats, which can be swiftly processed for searching and retrieval purposes.

4)Knowledge Graph Construction: At the same time, the text segments are examined using the Llama 3.3 LLM, which identifies significant entities and relationships within the legal text. The relationship triplets obtained are utilized to build a Knowledge Graph using the Property Graph Index tool in LlamaIndex. This allows for an in-depth understanding of legal concepts and their connections.

B. Vector and Graph Database

*1)Vector Storage (ChromaDB):*ChromaDB acts as the vector storage solution, indexing and storing the created vector embeddings. Along with the embeddings, metadata pertaining to each legal case document is included, ensuring that the document's context is preserved.

*2)Knowledge Graph Storage (NebulaGraph):*For the graph storage, NebulaGraph, a distributed graph database, is used. To avoid data overlap between legal cases, NebulaGraph's Spaces feature is implemented. Each legal case document is stored in its unique isolated space, which is organized through an SQLite database. This technique keeps data distinct, while metadata annotations help retain pertinent information for each document's context.

C. Hybrid Retrieval Engine

A tailored hybrid retrieval engine is designed to extract information from both vector and graph databases. When processing a query, cosine similarity is applied to retrieve vector embeddings, while entity matching is used to pull relevant relationships from the knowledge graph. The gathered information is ranked based on its connection to the query, taking into account factors such as the proximity of response vectors to the query vector and the quantity of relevant relationships.

D. Multi-Document Support

To effectively process several legal case documents, the Reciprocal Rank Fusion (RRF) method is used. This technique integrates results from multiple documents, assigning a score to each based on its rank across various documents. The final ranking prioritizes the most pertinent results from all documents, thus providing a thorough response to the query.

E. Response Generation

The Tree Summarize Method is utilized to create concise responses. This method merges and summarizes the text chunks recursively while adhering to the context window limits of the LLM. Each section is summarized, including its metadata for source attribution. The recursive summarization continues until only a single summary remains, which is then provided as the final response.

VI. CONCLUSION

The proposed AI-driven framework is specifically designed for the retrieval and interpretation of legal case documents, utilizing hybrid retrieval techniques to enhance information access. By integrating vector embeddings and knowledge graphs, the solution overcomes the limitations of traditional information retrieval systems, providing accurate and contextually relevant answers. The system ensures the authenticity of the information by annotating the sources from which it is derived, making it ideal for legal contexts where source verification is crucial.

Additionally, the framework supports multi-document analysis, allowing users to derive comprehensive insights from multiple legal case documents. This capability aids legal professionals in making informed decisions based on detailed analysis from a variety of sources. The system's scalable and adaptable architecture ensures it can be used across a wide range of legal documents, making it suitable for legal research, case study analysis, and courtroom preparation.

As legal data continues to grow, this solution offers an effective way to manage and utilize legal information, improving productivity and enabling data-driven decision-making within the legal industry.

REFERENCE

[1] E. Anderson *et al.*, "The Design of an LLM-Powered Unstructured Analytics System," *arXiv preprint arXiv:2409.00847v2*, Sep. 4, 2024. [Online]. Available: https://arxiv.org/abs/2409.00847.

[2] Y. Gao *et al.*, "Retrieval-Augmented Generation for Large Language Models: A Survey," *arXiv preprint arXiv:2312.10997v5*, Mar. 27, 2024. [Online]. Available: https://arxiv.org/abs/2312.10997.

[3] A. A. Khan, M. T. Hasan, K. K. Kemell, J. Rasku, and P. Abrahamsson, "Developing Retrieval Augmented Generation (RAG) based LLM Systems from PDFs: An Experience Report," *arXiv preprint arXiv:2410.15944v1*, Oct. 21, 2024. [Online]. Available: https://arxiv.org/abs/2410.15944.

[4] D. Edge *et al.*, "From Local to Global: A Graph RAG Approach to Query-Focused Summarization," *arXiv preprint arXiv:2404.16130v1*, Apr. 24, 2024. [Online]. Available: https://arxiv.org/abs/2404.16130.

[5] B. Sarmah, B. Hall, R. Rao, S. Patel, S. Pasquali, and D. Mehta**,** "HybridRAG: Integrating Knowledge Graphs and Vector Retrieval Augmented Generation for Efficient Information Extraction," *arXiv preprint* arXiv:2408.04948v1, Aug. 9, 2024. [Online]. Available: https://arxiv.org/abs/2408.04948.