Dynamic Agentic RAG with Pathway

Team 67

Abstract

In this work, we aim to develop an End-to-End Agentic RAG system utilizing Pathway's real-time data processing capabilities. We identify areas within the legal domain where current RAG systems could be improved upon. Our application should have intelligent decision-making beyond basic retrieval.

1. Introduction

1.1. Problem Statement

This problem statement focuses on the development of an **Agent-based Retrieval-Augmented Generation (RAG)** system using the Pathway framework. The challenge involves introducing novel solutions to the domain and technical issues to produce a highly accurate **question-answering model**. Overall, the target is the creation of robust, efficient, and flexible AI agents, which are capable of **decision-making** and adaptive learning from real-time data streams.

1.2. Use Case Selection and Novelty

Our chosen domain is the legal sector, selected for its complexity and the high accuracy and contextual understanding required to interpret legal texts effectively. Key challenges in this domain include handling specific terminology, interpreting precedents, supporting legal reasoning, and generating precise citations. Additionally, all RAG systems face the fundamental challenge of low accuracy, which becomes particularly critical in legal applications.

While Legal AI tools like Contract Lifecycle Management (CLM) systems and legal research tools are available, a gap remains in their capabilities. These tools are effective for generating standard, high-volume contracts but fall short in crafting unique, research-intensive, and bespoke contracts. Furthermore, none of these solutions are tailored to the Indian legal context and judiciary system, which has its own regulatory standards and complexities.

Our solution addresses these gaps by combining CLM and legal research functionalities into a unified tool specifically designed for drafting custom contracts that align with up-to-date Indian regulations. With features like real-time policy updates, in-depth case research, and automated contract generation, this platform enables legal professionals to craft contracts tailored to unique legal scenarios. Leveraging an Agentic RAG system, our solution is optimized for accuracy and adaptability in this demanding legal domain.

1.3. Solution Overview

Our solution focuses on two primary aspects of current issues in legal question answering. The first is **poor retrieval**, arising from **poor embedding quality** due to legal jargon, along with the **complexity of legal texts**. This complexity also leads to the second issue of **poor multi-hop reasoning** abilities. While this is an issue that affects LLM based question answering in all domains, it is especially pronounced in the legal space, due to the highly interconnected nature of most case docs.

We deal with the former by using improved legal-specific legal embedding models, such as **voyage-law-2**. We also experiment with reranking software for the same. The latter issue is dealt with by creating sophisticated agentic RAG systems that deal with multi-hop reasoning tasks in a step-by-step manner. Additionally, latency-related improvements are introduced by using **LLMCompiler**.

2. Literature Review

2.1. Parsing and Preprocessing

Parsing legal PDFs for RAG workflows is challenging due to complex layouts with nested headers, footnotes, and embedded tables. Traditional parsers like **PyPDF2** and **PDFMiner** can extract text but lack structural sensitivity, struggling with layout-heavy elements such as nested headers and irregular tables. **Camelot** performs moderately well on simple tables but is less effective for complex layouts. Vision-based language models (VLLMs) offer some promise but face challenges in production-level accuracy and efficiency.

To address these limitations, we developed a hybrid pipeline using **Unstructured** and **Docling**. **Unstructured** parses text into structured elements (headers, titles, etc.) in JSON format but underperforms on table extraction, so we skip tables in its parsing. For detailed table extraction, we use **Docling**, running it in parallel and iterating over Unstructured's output to replace or add table data from Docling's Markdown

output. In cases where Unstructured misses tables, we use **regex** to identify table-like structures and substitute them with Docling data.

Finally, we experiment with **FAIR's Nougat**, which enhances document structure preservation by converting PDFs into a specialized markdown format, maintaining layout accuracy, and improving table extraction. This hybrid method balances accuracy with efficiency, addressing the unique complexities of legal PDFs.

2.2. Chunking

Chunking within the RAG framework serves as a vital technique for enhancing the efficiency of data retrieval, especially in complex and highly structured domains such as legal documentation. Earlier methods focused on fundamental approaches to divide texts based on straightforward segmentation criteria, which were effective but not sufficient for handling the intricacies of legal language and references.

Initial trials demonstrated that simplistic chunking strategies often resulted in **fragmented information retrieval**, leading to a **loss of contextual integrity**. This compromised the quality of the generated responses, especially when addressing multifaceted legal queries that demanded nuanced understanding and precision. (Li et al. (2024))

Recognizing these challenges, several chunking techniques were tried to address the limitations identified during preliminary testing. Existing chunking techniques are tried to test their performance for this task in the legal domain. **Semantic chunking** helps in maintaining relevance between the sentences of the documents, which helps maintain **contextual reference in legal documents**, but, this only tackles one of the existing problems, which is determining an accurate breakpoint to segment the document.

The other problem that exists is the **loss of context** since legal documents require a lot of previous context as well due to the long length of the documents.

The results are shown in *Table 1* :

Chunking Techniques	Score
Semantic	0.5184648034046766
Hierarchical	0.46874114629227864

Table 1. Different chunking techniques performance

Legal documents often have a complex structure, including sections, subsections, clauses, and annexes. **Hierarchical chunking** mirrors this structure by processing text at multiple levels. Initially, larger sections are chunked and analyzed, and then these can be further broken down into subsections and clauses. This approach allows for maintaining the **integrity of legal arguments and references** throughout the document. This can also help in maintaining contextual relevance.

Table 2 represents the average scores of semantic chunking for different breakpoint thresholds for several queries. This infers that the scores reduce with the reducing thresholds due to less relevance in the chunks. Thus, we aim to perform the scores of the model at lesser thresholds.

Threshold(in percent)	Factual Correctness	Semantic Similarity*
95	0.48	0.880
90	0.46	0.878
85	0.42	0.872
80	0.33	0.866

Table 2. Semantic Chunking granularity

Dealing with the second issue is actually a challenging task. The simple RAG pipeline of **chunking-embedding-retrieving-generating** can destroy the long-distance contextual dependencies.

2.3. Retrievers

Retrieval-augmented generation (RAG) relies on sophisticated retrieval techniques to select candidate contexts that enhance user queries. Foundational retrievers, such as **TF-IDF** and **BM25** Kalra et al. (2024) use lexical similarities for query-document matching. For baseline comparisons, we evaluated **rank-bm25**, a widely used implementation integrated into libraries like LangChain and Llama-Index.

Dense Passage Retrieval Karpukhin et al. (2020) employs dense vector embeddings for efficient retrieval and is popular in RAG frameworks. However, DPR and similar methods are limited in **multi-hop** scenarios, often trained independently for each hop, which can undermine performance in complex multi-hop queries. To address multi-hop retrieval, we explored **Beam Retrieval** Zhang et al. (2024a), which uses an encoder and dual classification heads to score hypotheses for each passage. We also examined **SURE** Kim et al. (2024a) **,** a model that generates and validates summaries for multi-answer candidate sets, improving retrieval accuracy for intricate texts, like **case law** documents, by summarizing content before passing context to the generator.

In **legal RAG** frameworks, **HyPA RAG** Kalra et al. (2024) stands out with a query complexity classifier for adaptive parameter tuning, combining dense, sparse, and knowledge graph retrieval. Initial experiments included auto-merging techniques, BM25, and simple fusion retrievers combining BM25 and DPR.

Retriever	Score*
DPR	0.743
ColBERT	0.689
FAISS	0.636
BM25	0.614
T5	0.751

Table 3. Retriever Performance Scores

Embedding Model	Score*
vstackai-law-1	66.16
voyage-law-2	65.39
e5-mistral-7b-instruct	59.77
text-embedding-3-large	59.22

Table 4. Embedding-based Retrieval Scores (MTEB-Law)

2.4. Postprocessing

Rerankers can significantly improve the search quality because they operate at a sub-document and sub-query level, meaning they look at the individual words and phrases, their meanings, and how they relate to each other within the query and the documents. This results in a more precise and contextually relevant set of search results.

Most of the post-retrieval processing includes the usage of rerankers. We experimented using various rankers. **Cohere**'s reranking tool Cohere (2024) uses advanced language models to reorder lists based on contextual relevance, enhancing search accuracy and content discovery. **ColBERT** Santhanam et al. (2022) encodes them into matrices of token-level embeddings. At search time, it computes fine-grained similarities between query and document tokens, allowing for more precise relevance scoring. **FlagEmbedding**'s reranking models operate as cross-encoders, evaluating the relationship between a query and each retrieved document to assign a relevance score. The **Jina** Reranker v2 is a transformer-based model that has been fine-tuned for text reranking tasks. **LongContextReorder** ensures that the highest-scored (most relevant) nodes are positioned at the beginning and end of the list, with lower-scored nodes in the middle. A **meta-reranker** is a system that refines and reorders search results by aggregating and analyzing outputs from multiple search engines. **MixedBreadAIReranker** enhances search systems by reordering initial search results based on semantic relevance. **RankGPTRerank** is a reranking module that utilizes large language models (LLMs) to reorder search results. **SentenceTransformerRerank** utilizes cross-encoders to re-evaluate and reorder a list of candidate nodes.**TimeWeightedNodeProcessor** assigns higher relevance scores to newer documents or nodes, ensuring that search results reflect the most up-to-date information.

We used a **BM25 retriever** in the pipeline to evaluate the rerankers, as they rely on the retriever's output. Ideally, a poor retriever would weaken the impact on the final result, leaving the reranker to determine the outcome on its own. Instead of using the original query, we employed a **decomposed query** for evaluation.

Rerankers	Score*
Cohere	0.518
Colbert	0.468
Jina	0.480

Table 5. Rerankers Performance

2.5. Agentic RAG

Agentic RAG refers to an AI agent-bolstered form of RAG. Specifically, it incorporates AI agents into the RAG pipeline to orchestrate its components and perform additional actions beyond simple information retrieval and generation to overcome the limitations of the basic RAG pipelines.

Our literature review encompassed a wide variety of agentic architectures, starting with basic **ReAct(Reason+Act) agents**. These are single-agent systems equipped with query planning, tool use, and routing abilities. While these agents function well for simpler tasks, they plan for 1 sub-problem at a time. this leads to sub-optimal trajectories since it isn't forced to reason about the whole task.

This led us to explore more complex multi-agentic systems, starting with **ReWOO**, which follows a Planner-Worker-Solver framework. The planner generates a list of tasks (a plan) to follow step by step, which is then executed by Worker agents, who update the state with task results. These results are then used by a Solver to generate a complete answer. This basic framework is often used by other systems as well, serving as a good baseline architecture. One such system is **ReSP**, which involves summarization to supplement the LLm knowledge base for sub-question answering. A major drawback is the sequential execution of tasks, leading to longer runtimes.

Current methods for function calling often require sequential reasoning and acting for each function which can result in high latency, cost, and sometimes inaccurate behavior.Drawing inspiration from the principles of classical compilers, **LLMCompiler** enables parallel function calling with three components: (i) a Function Calling Planner, formulating execution plans for function calling; (ii) a Task Fetching Unit, dispatching function calling tasks; and (iii) an Executor, executing these tasks in parallel.

Planner the planner accepts the input question and generates a task list to execute. If it is provided with a previous plan, it is instructed to **re-plan**, which is useful if, upon completion of the first batch of tasks, the agent must take

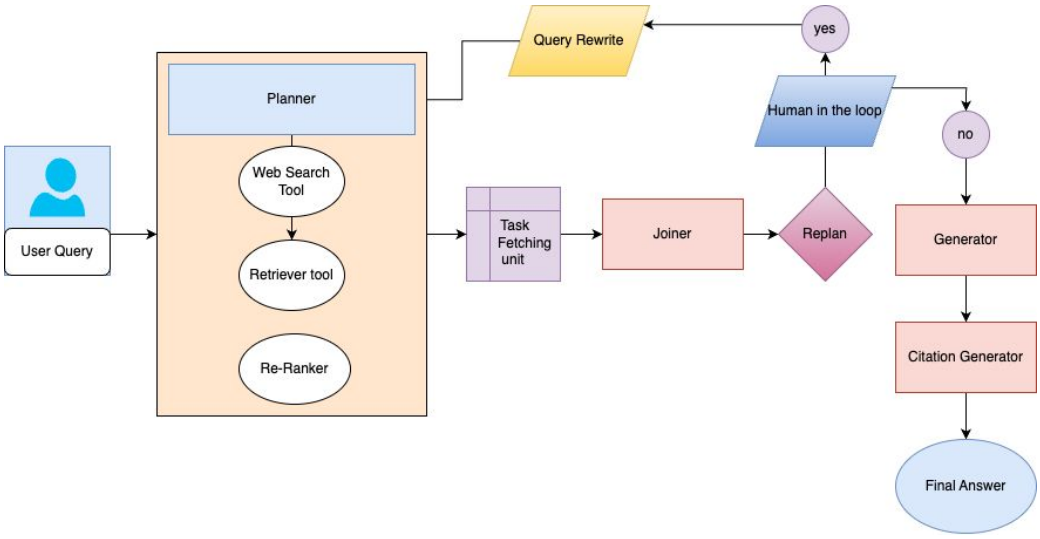

Figure 1. Overview of the current pipeline

more action.

Task Fetching Unit This component schedules the tasks. It receives a stream of tools of the following format: tool: BaseTool, dependencies: number[], The basic idea is to begin executing tools as soon as their dependencies are met. This is done through multi-threading.

Joiner So now we have the planning and initial execution done. We need a component to process these outputs and either respond with the correct answer or loop with a new plan. We are using function calling to improve parsing reliability.

Dynamic Replanning In various applications, the execution graph may need to adapt based on intermediate results that are a priori unknown. When replanning, the intermediate results are sent back from the Executor to the Function Calling Planner which then generates a new set of tasks with their associated dependencies. These tasks are then sent to the Task Fetching Unit and subsequently to the Executor. This cycle continues until the desired final result is achieved and can be delivered to the user.

2.6. Citation Generation

Citation generation in legal contexts has dual benefits: it serves the practical need for precise legal references, while also helping to mitigate issues like **factual errors and hallucinations** or fabricated details, which LLMs can sometimes produce. First, **citation generation** enables real-time verification of facts. Such a system also directly addresses the need for generating relevant citations. Legal practitioners often require precise references to statutes, case law, or legal theories to build or substantiate arguments, support motions, or draft legal documents. We can explore two basic methods— **few-shot** and **fine-tuning—to guide** LLMs in generating responses with citations. In existing works, there are generally two types of methods to provide citations for responses: the **pre-hoc citation** and the **post-hoc citation**. The pre-hoc method treats citations as **regular tokens** and generates them directly during the inference process of LLMs, which places high demands on the capabilities of the LLMs. In contrast, the post-hoc method first generates a response w,ithout citations and then matches the content of the response with references to determine whether citations need to be added. We thought of fine-tuning an LLM as a refiner to have three capabilities: (1) keep relevant citations within the response; (2) add necessary citations that are missing; (3) remove any irrelevant citations that are present.

3. System Architecture

For our system architecture we propose 2 pipelines here, one that builds upon LLMCompiler Kim et al. (2024b) and another that takes inspiration from Jiang et al. (2024) and Kim et al. (2024b). The purpose of proposing two pipelines here is necessitated by the complexity of modifying the LLMCompiler architecture for our purposes and use case. The alternative second pipeline allows us to easily modify and test out different components of our envisioned pipeline.

3.1. LLMCompiler

LLMCompiler is an agent architecture designed to speed up the execution of agentic tasks by eagerly-executed tasks within a DAG. It also saves costs on redundant token usage by reducing the number of calls to the LLM. It has 3 main components:

- Planner: stream a DAG of tasks

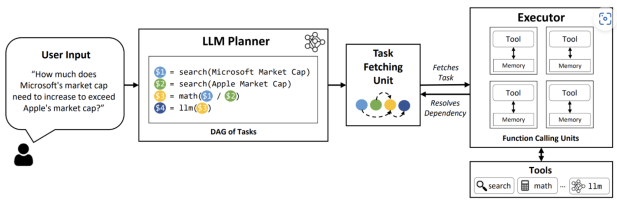

Figure 2. **LLMCompiler** is a framework that enables an *efficient and effective orchestration of parallel function calling* with LLMs, including both open-source and close-source models, by automatically identifying which tasks can be performed in parallel and which ones are interdependent

- Task Fetching Unit: schedules and executes the tasks as soon as they are executable
- Joiner: Responds to the user or triggers a second plan

3.2. Pipeline Overview

3.2.1. Main Pipeline

Our main pipeline proposes using **LLMCompiler** for efficient tool calling with remarkable cost optimization to schedule and execute tasks gracefully. In this pipeline, we have a **planner agent** which makes a plan on how to go about answering a given query and based on the plan, a **scheduler** schedules tasks and an **executer** executes them. Here we propose giving the planner multiple tools, including a diverse set of retriever tools and reranker tools among others. The novelty in this part of the pipeline is the **efficient parallel tool execution**, which is incredibly cost-effective. The retrieved documents from the output of the planner are fed into a joiner agent which evaluates, using LLM as a judge, whether the retrieved documents are relevant to the query.

Based on the response of the judge, the joiner then redirects the flow to either the Rewrite Agent. This agent, based on the previous response as its context, attempts to rewrite the query by decomposing it into sub-queries and feeding it to the planner to get a new plan and execute it. The planner is also fed the history of the tools used in the previous plan and is asked to consider using alternate retriever tools or reranker tools in the new plan, to provide alternative pathways for the agent to execute the task.

Our pipeline employs human-in-the-loop at decision points where the joiner must determine whether to proceed with **re-ranking** available answers or **directly generating a response**. At these junctures, the system triggers an **interrupt** for human feedback. This feedback guides the joiner, enabling it to make the best decision based on human judgment and improving overall responsiveness to specific needs.

After the joiner deems the retrieved documents to be relevant to the query and is then fed to the generator which finally generates the answer. The pipeline is then redirected to the citation generator agent which generates the proper references for the given answer in the widely used **Oscola** format before giving the final answer to the user.

3.2.2. Alternative Pipeline

This pipeline leverages a combination of ReSP (Retrieval-Enhanced Specialized Processing) and an LLM Compiler-like framework to handle complex legal queries efficiently. The system is designed to process queries through multiple specialized agents and dynamic planning, ensuring high accuracy and relevance in the final output. The user query is first processed through Voyage embeddings. If Voyage embeddings fail to yield a relevant match—either due to insufficient contextual overlap, out-of-distribution input, or lack of pertinent information in the database—the query is passed to Mistral embeddings for further processing. The embeddings are used to classify the query into one of the following categories:

- Case Law Agent
- Contract Law Agent
- Due Diligence Agent

Within each specialized agent, the query is sent to a Planner Agent inspired by the LLM Compiler framework. The Planner decomposes the query into sub-queries, each designed to address a specific aspect of the original query. Sub-queries are routed based on their nature to either a web search agent or a retriever agent leveraging adaptive RAG. Once sub-query results are retrieved, they are evaluated using Corrective RAG: If irrelevant documents are being used, the sub-query is rewritten and sent back to the Planner Agent for another iteration. This process is repeated n times until the retrieved results meet relevance criteria. Sub-query answers that pass corrective evaluation are stored in local memory for later use. Simultaneously, the overarching user query is passed to a Summarizer Agent, which maintains a high-level context in global memory. The global memory is enriched with summaries that ensure coherence and connectivity between sub-queries. Content from local memory and global memory is fed to a Generator Agent. The generator combines the retrieved information to produce a cohesive final response while eliminating irrelevant context. Once the final answer is generated, citation generation is appended to attribute sources, ensuring verifiability and transparency.

3.3. Integrating Pathway

3.3.1. Pathway

Pathway works with dynamic data, we added, and deleted files in the doc, and the corresponding nodes retrieved were

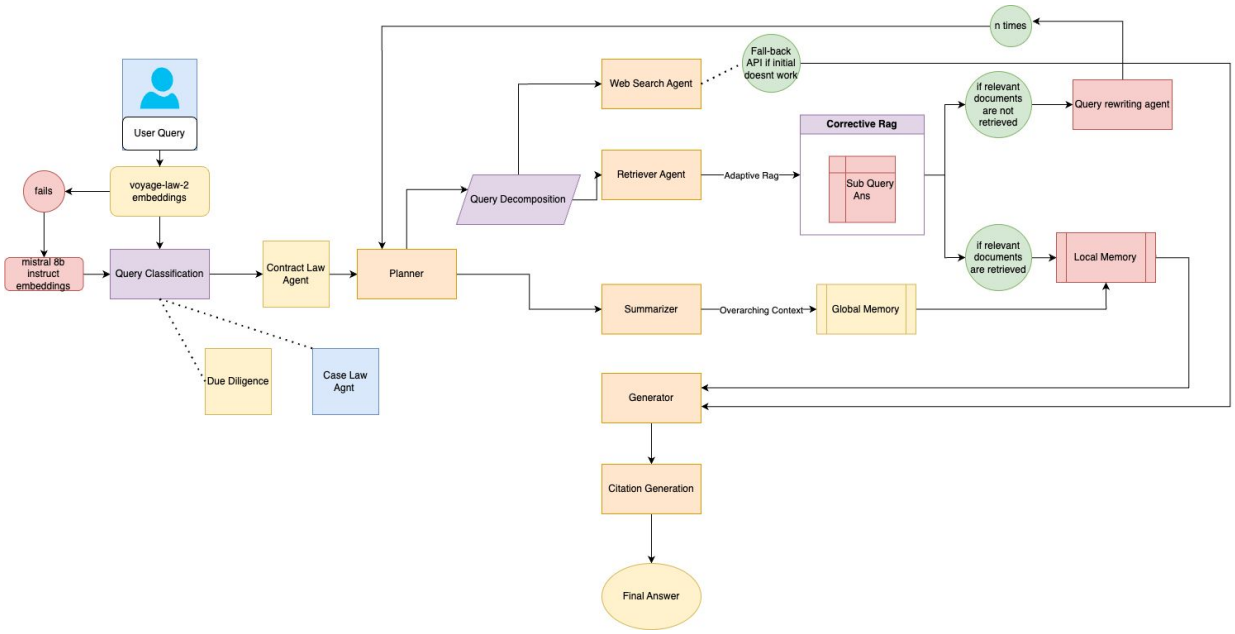

Figure 3. Alternate Pipeline

changed.

HNSW is an algorithm for fast approximate nearest neighbor (ANN) search. It builds a multi-layered, navigable small-world graph, where each layer has increasingly dense connections, similar to a skip list structure. This enables efficient ”skipping” through layers, starting from sparse connections at the top and gradually moving to dense connections at lower levels. Instead of brute-force comparisons, HNSW quickly locates neighbors by hopping through these layers, reducing search time from O(N) to approximately O(log N). This makes HNSW much faster and more scalable than brute-force KNN, especially in high-dimensional spaces. We are also using the default pathway's KNN indexing which uses the HNSW algorithm.

4. Experimentation and initial results

4.1. Setup

Our setup consists of a LangGraph based implementation of LLMCompiler. We experimented with OpenAI and VoyageAI embeddings, along with Cohere and JinaAI rerankers. Additionally, we used Pathway's VectorStore. Our base setup used Pathway's retrieval mechanism, while Auto-Merging retrieval and Contextual Compression retrieval were also experimented with.

4.2. Datasets

We used the CUAD dataset which is a specialized dataset designed to enhance natural language processing (NLP) applications in legal contract review. We also used AILA which focuses on developing AI models to assist in legal contexts, specifically in identifying relevant prior cases and statutes for given legal scenarios. Both of these datasets were used for evaluation and measuring how well our pipeline was working.

4.3. Evaluation

For generating question-answer format queries we used GiskardAI. The queries were of the format :

For assessing the performance of the RAG pipeline, we used RAGAS, focusing on two key metrics: **Factual Correctness** and **Semantic Similarity**. We introduced the **Factuality-Weighted Semantic Score (FWS)**. This score combines factual correctness (F) and semantic similarity (S), with each metric weighted based on its relevance to our application. Since we prioritize accuracy, Factual Correctness was given a weight of 0.75, and the remaining was assigned to Semantic Similarity.

$$FWS = (1 - \alpha) \times SS + \alpha \times FC \tag{1}$$

where:

- α is the weighting factor between semantic search and factual correctness.
- Semantic Search represents the semantic similarity score.
- Factual Correctness represents the factual correctness score.

Queries Generated	Query Type	Query Example
15	Clause Identification	What is the penalty for late payment in this contract?
15	Multi-hop Reasoning	If a vendor fails to deliver the product on time, what are the consequences for both parties?
15	Long Context Understanding	What rights does the purchaser have if the vendor delivers faulty products?
5	Clause Matching	Find the clause that mentions the duration of the agreement.
10	Specific Condition Query	What happens if either party breaches confidentiality?
5	Clause Extraction with Context	What is the payment schedule for the project outlined in the agreement?
10	Multi-step Inference	What happens if a payment is delayed and the product is delivered late at the same time?
5	Clause Impact Analysis	If the delivery clause is breached, what penalty applies according to the contract?
5	Temporal Reasoning	When must the final payment be made according to the contract?

Table 6. Simulated CUAD Queries Dataset for Evaluation

5. Challenges and Next Steps

5.1. Pathway

5.1.1. HTTP Time Out

The vector store server takes a noticeable amount of time to initialize before it can respond to queries; otherwise, it gives an HTTP timeout error. Currently, we're using a try-except block to handle this. We didn't use thread synchronization because the LLM compiler isn't thread-safe with it. In the final pipeline, we'll be deploying our vector server on the cloud and connecting via a Google Drive connector instead of running it locally. This way, we won't need to start the server each time, so the HTTP timeout issue shouldn't occur.

5.1.2. Integration Issues

Integrating external retrievers and indexing methods not natively supported by Pathway has proven to be challenging without extensive modifications to the source code. We did actually modify the code to accommodate different indexing and retrieval types, such as those from LlamaIndex. However, we found that using Pathway as an initial retrieval layer (setting KNN to 50), followed by external retrieval and reindexing, significantly improved performance. Based on these results, we plan to implement a "two-layer" retrieval system in the final pipeline, using Pathway as the first layer for faster initial retrieval before incorporating external methods.

5.2. Next Steps

5.2.1. Late Chunking

A brand-new chunking technique, **Late Chunking** can be experimented with. However, integrating this method with the LangChain framework is a complex and time-consuming task. Given these considerations, the integration of Late Chunking is a future project, post the midterm evaluation.

5.2.2. Agentic Chunking

Another new chunking technique, **Agentic Chunking** can be experimented with as well. It breaks the paragraphs into propositions, which can help avoid the major issues that are faced when the context is lost due to pronouns or previous references.

5.2.3. CORAG

We hope to use a **Monte Carlo Tree Search (MCTS)**-based policy framework to sequentially find optimal chunk combinations, accounting for correlations among chunks and incorporating budget constraints along the way to consider the non-monotonic utility of chunks. Wang et al. (2024)

5.2.4. Beam Retriever

We also aim to implement a version of Beam Retriever (Zhang et al. (2024a)) as our retriever, due to its excellence in retrieval in multi-hop reasoning tasks. It achieves this by using an encoder and two classification heads to model the multi-hop retrieval process. It also maintains multiple hypotheses for possible relevant passages, reducing the impact of retrieval errors.

5.2.5. Contextual Retrieval

Currently, we faced difficulties in retrieval within the Contract Law domain itself where most of the statements within a document do not specify any context of their relevance. Our retrieval mechanism suffers because of this.

To tackle this, we plan to use Anthropics's Contextual Retrieval Anthropic (2024a) which solves this problem by prepending chunk-specific explanatory context to each chunk before embedding ("Contextual Embeddings") and creating the BM25 index ("Contextual BM25").

Additionally, we plan to use Prompt Caching Anthropic (2024b) to reduce the costs of Contextual Retrieval Assuming 800 token chunks, 8k token documents, 50 token context instructions, and 100 tokens of context per chunk, the one-time cost to generate contextualized chunks is $1.02 per million document tokens.

References

Anthropic (2024a). Introducing contextual retrieval. Accessed: 2024-11-12.

Anthropic (2024b). Prompt caching with claude. Accessed: 2024-11-12.

Auer, C., Lysak, M., Nassar, A., Dolfi, M., Livathinos, N., Vagenas, P., Ramis, C. B., Omenetti, M., Lindlbauer, F., Dinkla, K., Mishra, L., Kim, Y., Gupta, S., de Lima, R. T., Weber, V., Morin, L., Meijer, I., Kuropiatnyk, V., and Staar, P. W. J. (2024). Docling technical report.

Blecher, L., Cucurull, G., Scialom, T., and Stojnic, R. (2023). Nougat: Neural optical understanding for academic documents.

Cohere (2024). Introducing rerank 3: A new foundation model for efficient enterprise search & retrieval. Accessed: 2024-11-12.

Cui, Y., Sun, Z., and Hu, W. (2024). A prompt-based knowledge graph foundation model for universal in-context reasoning.

Dhani, J. S., Bhatt, R., Ganesan, B., Sirohi, P., and Bhatnagar, V. (2024). Similar cases recommendation using legal knowledge graphs.

Finardi, P., Avila, L., Castaldoni, R., Gengo, P., Larcher, C., Piau, M., Costa, P., and Caridá, V. (2024). The chronicles of rag: The retriever, the chunk and the generator.

Günther, M., Mohr, I., Williams, D. J., Wang, B., and Xiao, H. (2024). Late chunking: Contextual chunk embeddings using long-context embedding models.

Jiang, Z., Sun, M., Liang, L., and Zhang, Z. (2024). Retrieve, summarize, plan: Advancing multi-hop question answering with an iterative approach.

Kalra, R., Wu, Z., Gulley, A., Hilliard, A., Guan, X., Koshiyama, A., and Treleaven, P. (2024). Hypa-rag: A hybrid parameter adaptive retrieval-augmented generation system for ai legal and policy applications.

Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and tau Yih, W. (2020). Dense passage retrieval for open-domain question answering.

Kim, J., Nam, J., Mo, S., Park, J., Lee, S.-W., Seo, M., Ha, J.-W., and Shin, J. (2024a). Sure: Summarizing retrievals using answer candidates for open-domain qa of llms.

Kim, S., Moon, S., Tabrizi, R., Lee, N., Mahoney, M. W., Keutzer, K., and Gholami, A. (2024b). An llm compiler for parallel function calling.

Li, W., Li, J., Ma, W., and Liu, Y. (2024). Citation-enhanced generation for llm-based chatbots.

Mamakas, D., Tsotsi, P., Androutsopoulos, I., and Chalkidis, I. (2022). Processing long legal documents with pretrained transformers: Modding legalbert and longformer.

Mavromatis, C. and Karypis, G. (2024). Gnn-rag: Graph neural retrieval for large language model reasoning.

Santhanam, K., Khattab, O., Saad-Falcon, J., Potts, C., and Zaharia, M. (2022). Colbertv2: Effective and efficient retrieval via lightweight late interaction.

Sarmah, B., Hall, B., Rao, R., Patel, S., Pasquali, S., and Mehta, D. (2024). Hybridrag: Integrating knowledge graphs and vector retrieval augmented generation for efficient information extraction.

Shi, Y., Zi, X., Shi, Z., Zhang, H., Wu, Q., and Xu, M. (2024a). Enhancing retrieval and managing retrieval: A four-module synergy for improved quality and efficiency in rag systems.

Shi, Z., Ming, Y., Nguyen, X.-P., Liang, Y., and Joty, S. (2024b). Discovering the gems in early layers: Accelerating long-context llms with 1000x input token reduction.

Singh, I. S., Aggarwal, R., Allahverdiyev, I., Taha, M., Akalin, A., Zhu, K., and O'Brien, S. (2024). Chunkrag: Novel llm-chunk filtering method for rag systems.

Wang, Z., Yuan, H., Dong, W., Cong, G., and Li, F. (2024). Corag: A cost-constrained retrieval optimization system for retrieval-augmented generation.

Wiratunga, N., Abeyratne, R., Jayawardena, L., Martin, K., Massie, S., Nkisi-Orji, I., Weerasinghe, R., Liret, A., and Fleisch, B. (2024). Cbr-rag: Case-based reasoning for retrieval augmented generation in llms for legal question answering.

Wu, J., Yang, L., Ji, Y., Huang, W., Karlsson, B. F., and Okumura, M. (2024). Gendec: A robust generative question-decomposition method for multi-hop reasoning.

Zhang, J., Zhang, H., Zhang, D., Liu, Y., and Huang, S. (2024a). End-to-end beam retrieval for multi-hop question answering.

Zhang, Y., Sun, R., Chen, Y., Pfister, T., Zhang, R., and Arik, S. (2024b). Chain of agents: Large language models collaborating on long-context tasks.

6. Appendix

6.1. Competitive Audit

Company	Website	Description	USP	Feature List	User Group	Market Practice Area	Pricing Model	Integrations	Tech Stack	Market Presence and Reputation	Geographical Locations	Customer Satisfaction
Casetext (CoCounsel)	https://casetext.com/	First AI legal assistant tailored for legal professionals with capabilities like document review, legal research, and contract analysis.	The first and only professional-grade AI legal assistant seamlessly integrated with Thomson Reuters.	Contract Analysis Deposition Preparation Document Review Drafting Correspondence Legal Research Summarization Timeline Drafting	In-House Legal Law Firms	General Litigation Transactional Law	CoCounsel Core: $250/user/month with commitment options	Document Management Systems Westlaw Precision	- GPT-4 Powered - Web-based platform - Security-focused	Used by 45+ large U.S. law firms - Recognized in industry publications	Australia Canada US	Users note time savings and integration benefits
Luminance	https://www.luminance.com/	Proprietary Legal LLM trained on extensive documents for contract analysis, due diligence, and eDiscovery.	The only proprietary legal LLM, purpose-built for comprehensive contract lifecycle management	Autopilot Chatbot Contract Analysis Due Diligence Negotiation Repository Workflow Automation	Financial Institutions In-House Legal Law Firms	Transactional Law	Unavailable	APIs Document Management Systems Dropbox Microsoft Word Salesforce Westlaw Precision	- Proprietary Legal LLM - Cloud-based - ISO 27001, SOC2 certified	Used by 700+ global organizations, including notable corporations	Global	Commended for efficient document review and AI insights
Harvey	https://www.harvey.ai/	AI platform with domain-specific legal assistants enhancing efficiency and accuracy in legal tasks.	Domain-specific AI platform tailored to enhance efficiency and accuracy for legal professionals and enterprises.	AI Assistant Legal Research Secure Vault Workflow Automation	In-House Legal Law Firms Professional Service Providers	General	Unavailable	APIs Microsoft Word Westlaw Precision	- Custom-trained LLMs - Cloud-based - Enterprise-grade security	Valued at $1.5 billion after 2024 funding	Global	Users praise streamlined research, document analysis
Ironclad	https://ironcladapp.com/	Comprehensive AI-powered CLM platform to streamline entire contract lifecycle.	Comprehensive AI-powered CLM platform designed to streamline contract management across all business functions.	Analytics Repository Workflow Automation	In-House Legal Operations & HR	Transactional Law	Custom pricing	DocuSign Dropbox HelloSign Salesforce	- Cloud-based - Strong security practices	Trusted by companies like L'Oréal, Mastercard	Global	Users appreciate intuitive design, streamlined contract processes
Everlaw	https://www.everlaw.com/	Cloud-based eDiscovery and litigation platform with analytics and collaboration tools.	The world's most advanced cloud-based eDiscovery platform with analytics and collaboration for faster, deeper insights.	Analytics Data Visualisation eDiscovery	In-House Legal Law Firms	Litigation	Subscription-based, custom pricing	APIs Google Workspace Office 365	- Cloud-based - Proprietary AI - SOC 2, HIPAA compliant	Widely used by law firms, corporate legal departments	Global	Recognized for intuitive design, powerful functionality
Kira Systems	https://kirasystems.com/	AI-powered contract analysis platform for quick, accurate clause identification and analysis.	Patented AI-powered platform for quick, accurate clause identification and contract analysis.	Contract Analysis Custom ML Due Diligence Lease Abstraction Workflow Automation	In-House Legal Law Firms Professional Service Providers	Transactional Law	Unavailable	APIs Document Management Systems	- Proprietary AI models - Cloud-based - Strong security protocols	Widely used globally in legal and corporate sectors	Global	Praised for accurate analysis, streamlined workflows
Lex Machina	https://lexmachina.com/	Legal Analytics platform leveraging AI to provide court, judge, and litigation insights.	AI-driven Legal Analytics platform for actionable court, judge, and litigation insights.	Analytics Litigation Analytics	Academic Institutions In-House Legal Law Firms	Litigation	Unavailable	APIs Data Export	- Proprietary AI - Cloud-based - Robust data security	Leader in legal analytics, globally used by top firms	Global	Users value actionable insights for litigation strategies
LexisNexis	https://www.lexisnexis.com/en-us	Comprehensive legal research and analytics tools powered by Lexis+ AI for efficiency in research and document drafting.	Lexis+ AI: Comprehensive legal research and drafting tools, personalized with trusted LexisNexis content and Shepard's® validation.	Legal Research Litigation Analytics Shepard's Citations	Academic Institutions In-House Legal Law Firms	General Litigation Transactional Law	Lexis+: $105/month (3-year commitment); Lexis+ AI custom pricing	APIs Microsoft Word	- Proprietary AI - Cloud-based - SOC 2 compliant	Widely adopted by law firms, corporate legal departments	Global	Known for extensive resources and advanced search features
ROSS Intelligence	https://www.rossintelligence.com/what-is-ai	AI-powered research platform for efficient legal research through natural language processing.	AI-powered research platform enabling fast, in-depth legal research with natural language processing.	Document Review Shepard's Citations	Academic Institutions Law Firms	General	Unavailable	APIs Document Management Systems	- Proprietary AI - Cloud-based - Security-focused	Recognized for innovative legal research solution	North America	Praised for efficiency in legal research
SuperLegal	https://www.superlegal.ai/	AI-powered contract review tailored for SMBs, offering fast, affordable analysis with attorney oversight.	AI-driven contract review with attorney oversight, saving 90% on legal costs	Attorney Verification Contract Analysis Contract Management	Corporates SMB	Transactional Law	Transparent pricing with savings up to 90%	APIs Document Management Systems	- Proprietary AI - Cloud-based - Strong security measures	Featured in publications for its approach to SMB contract needs	Global US	Users commend for speed and affordability
ThoughtRiver	https://www.thoughtriver.com/	AI-powered contract review to reduce contract review times and ensure compliance with digital issue tracking.	AI-powered contract review for rapid, compliant third-party responses with digital issue tracking.	Contract Analysis Contract Management	Corporates In-House Legal Law Firms	Transactional Law	Unavailable	APIs Office 365	- Proprietary AI - Cloud-based - Secure data protection	Leader in AI-powered contract review	Global	Known for providing detailed contract insights
Juro	https://juro.com/	All-in-one contract automation platform for creating, approving, signing, and managing contracts.	All-in-one contract automation platform for seamless creation, approval, signing, and management across teams	Contract Analysis Contract Management Workflow Automation	Corporates In-House Legal Operations & HR	Transactional Law	Custom, scalable pricing with tailored plans	Greenhouse HubSpot Salesforce Slack	- AI Assistant - Cloud-based - SOC 2 Type 2 compliant	Trusted by over 5,000 companies globally	Global	Users value seamless integration and efficiency in contract workflows

Figure 4. The competitive audit reveals a landscape where AI-driven legal tools focus on document review, contract lifecycle management, and litigation support, with varying degrees of specialization. Our audit examined leading platforms such as Casetext, Luminance, Ironclad, and Everlaw, assessing their unique selling points, feature sets, integrations, and user group focus. This analysis helps identify key market gaps and strengths, guiding our product development to emphasize research-integrated contract drafting for non-standard, compliance-focused legal documents.