INTER IIT
TECH MEET 13.0

HIGH PREP

.pathway

DYNAMIC AGENTIC RAG WITH PATHWAY

TEAM 64

INDEX

ABSTRACT

We developed a multi-agent RAG system for the legal domain that simulates court proceedings. Leveraging Pathway's dynamic RAG capabilities and it's VectorStore, the system autonomously retrieves and analyzes legal data and synthesizes arguments. By implementing strategies like chain of thoughts (CoT) prompting, human-in-the-loop RAG, and ensuring correctness of data along with handling API failures, it provides a robust framework for intelligent legal information processing and case resolution.

1. INTRODUCTION

Legal systems involve complex decision-making processes requiring accurate and efficient information retrieval. Traditional systems often fail to adapt to the dynamic nature of legal queries and confidential data handling. This project addresses these challenges by building a multi-agent system that simulates court proceedings, leveraging advanced RAG techniques such as human-in-the-loop RAG, dynamic RAG and chain of thoughts (CoT) prompting. We strive to build responsible agents by implementing an independent and impartial fact-checking system. The following sections discuss system design, implementation details, evaluation results, and potential future improvements.

2. UNIQUENESS

1. Our solution uniquely simulates the proceedings of a Legal Criminal Court, unlike general question-answering RAG based apps.
2. Each agent in our system utilizes "**chain of thought**" reasoning, allowing for step-by-step reasoning processes that are especially useful for legal argumentation.
3. Use of Pathway's Vectorstore database allows for efficient storage and retrieval of legal precedents. By **compartmentalizing** case data (i.e., public and private databases), we ensure user privacy by storing user-data separately and deleting it after every instance.
4. Real-time **feedback from the user on the case arguments** when specific clarifications or external expertise is required.

2.1 USE-CASE SELECTION

Our use case and implementation are unique because unlike traditional RAG systems that rely on a single retrieval and generation pipeline, our design features **multiple agents** that collaborate and leverage unique capabilities to address complex legal scenarios. The agents collaborate to simulate court proceedings, where a lawyer and a prosecutor state argument and a judge analyses them to give the final verdict.

3. SOLUTION

3.1 IMPLEMENTATION

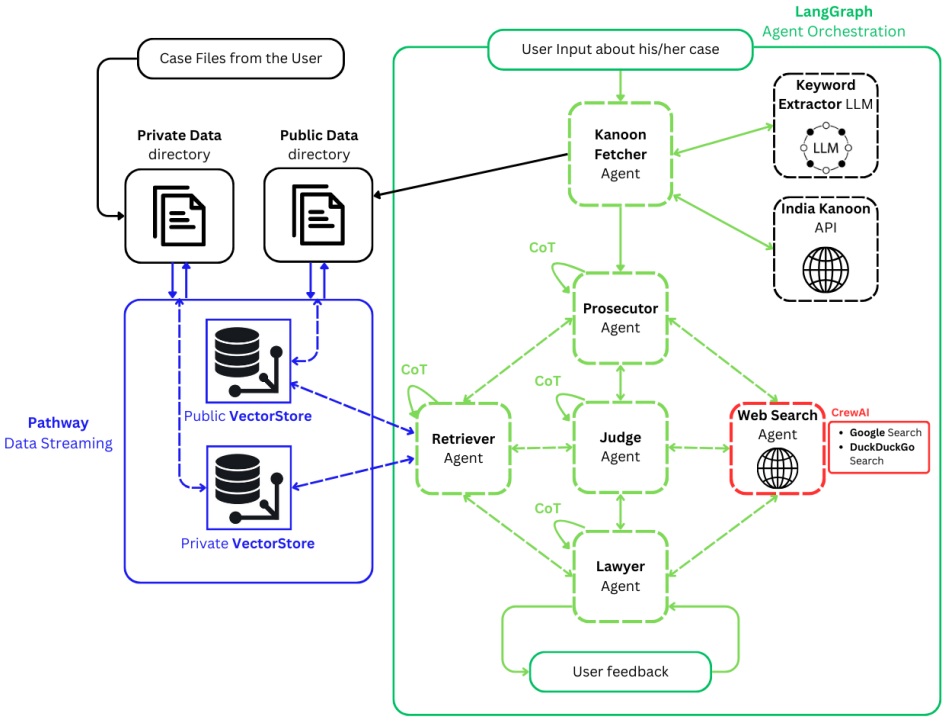


Figure1: System Architecture

The above diagram clearly visualizes the system working and architecture. It shows the pathway's dynamic data pipeline and its vector databases, all the agents and their interaction. Below is a detailed breakdown of the system architecture:

3.1.1 AN OVERVIEW OF AGENTS

1. Lawyer - This agent represents the user in the trial. It has access to both public and private databases, ensuring comprehensive argument formulation. It employs chain-of-thought reasoning for analysis of information available. It can call the internet data retriever agent and ask the user for additional information.
2. Prosecutor - This agent is tasked with countering the lawyer. It has access to the public and private databases. Similar to the lawyer, it employs chain-of-thought reasoning for analysis of information available and has access to the internet data retriever.
3. Judge - This agent acts as an impartial entity and fact-checks every argument presented by the Lawyer and Prosecutor and ensures that the arguments are based on actual information. It also gives the final verdict at the end of each trial simulation based on the arguments and facts presented.
4. Internet Data Retriever - This agent has access to the internet. It receives an argument as input. Based on the argument to be countered, it forms relevant search queries to search the internet. Using the information retrieved from the web, it makes returns a counter argument, citing the relevant sources.
5. Retriever Agents – This agent retrieves data from the pathway vector database using similarity search technique discussed in appendix. The kanoon retriever agent extracts keywords from user query and documents, fetch relevant case documents from the online database, preprocesses them and stores them in the public database.

3.1.2 DATABASE

We utilize Pathway's VectorStores and dynamic pipeline to build a real-time RAG system with up-to-date information from files stored in a directory along with 2 separate datastores- a public vectorstore and a private vectorstore. The public database stores documents available publicly such as the IPC and relevant cases fetched from Indiankanoon.org – an online database of Indian court hearings by the kanoon retriever agent. The private database stores the private documents of the user. This separation ensures user privacy of our client.

Why Pathway?
Pathway is an open data processing framework. It is written in Rust, which is computationally faster and gives results with reduced latency during data processing. It is easily integrable with python code, and it has pipelines that work with live data sources. It handles loading and indexing without ETL. Once updates are made to the files in the directory, the updated content is immediately re-indexed, and we don't have to deal with rerunning the pipeline.

3.1.3 WORKFLOW

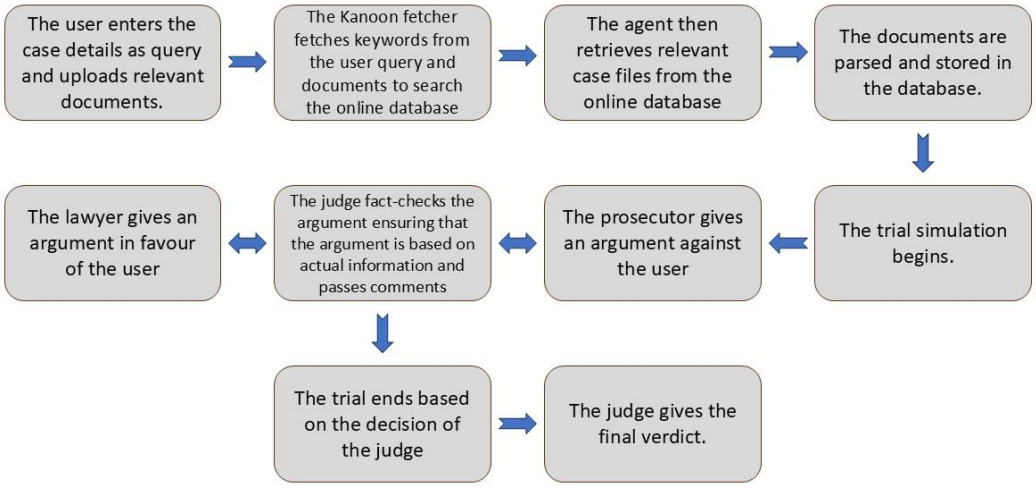


Figure2: Workflow Flow Chart

3.1.4 DEPLOYMENT

- **Backend:** Implemented using Python, with Pathway's API for vector search and data handling along with langgraph. Also used frameworks such as crewai for building the internet data retriever agent.
- **Frontend:** A lightweight UI built with Streamlit to allow users to visualize agent decisions, reasoning steps, and case outcomes.
- **Containerization:** The entire system is packaged in Docker for consistent deployment across environments.

3.2 FEATURES

1. Dynamic RAG with Pathway:

We have made use of Pathway's dynamic pipelines that handles loading and indexing without ETL, i.e., if any changes are made to the files in the directory, these changes automatically reflect, and the content is immediately re-indexed without rerunning the pipeline. This makes

our system dynamic and always keeps it up to date.

2. Chain of Thoughts Prompting:

We have employed chain of thought prompting for our agents. It is a technique where the model is guided to generate intermediate reasoning steps explicitly before arriving at a final answer. By reasoning step-by-step, the system reduces errors and avoids oversimplifications, especially when dealing with intricate legal scenarios. Based on the research paper by the Google Brain Team, it is observed that CoT improves the commonsense reasoning abilities of language models.

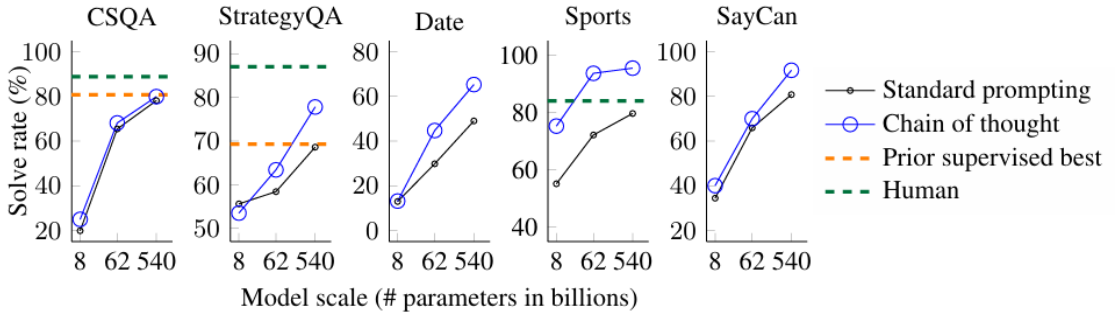


Figure3: Results from Google Brain Paper

The above results are borrowed from the paper "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models" by the Google Brain Team. Here, the language model shown is PaLM. Prior best numbers are from the leaderboards of CSQA (Talmor et al., 2019) and StrategyQA (Geva et al., 2021) (single model only, as of May 5, 2022). Implementing CoT shows increased reasoning abilities and accurate outputs.

3. Human-in-the-loop RAG:

Our Retrieval-Augmented Generation (RAG) system integrates a human-in-the-loop mechanism to enhance accuracy and decision-making. This feature enables users to provide input or validate outputs at critical stages, ensuring greater contextual relevance and reducing the risk of errors or biases. In our implementation, users can offer additional information or feedback after every argument, making this interactive approach particularly effective for addressing complex legal queries and delivering precise, tailored insights.

4. RESILIENCE TO ERROR HANDLING

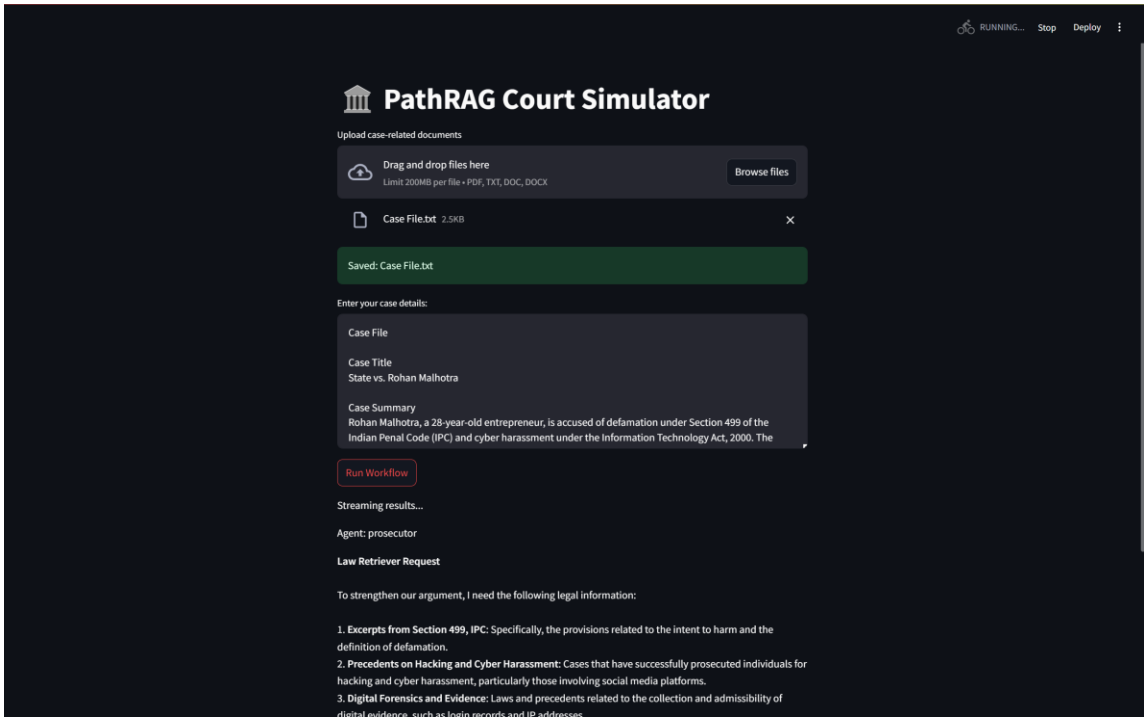
Our agents are designed to ensure robustness by leveraging multiple information sources to handle potential failures. In addition to accessing data from the vector store, agents can retrieve internet data and rely on the ultimate fallback source: the user. We utilize Serper's Google Search API, and to enhance reliability, we have implemented a fallback mechanism to perform web searches using DuckDuckGo if the API fails or the key is invalid. To manage rate limit errors when utilizing LLMs from providers like Groq, we implemented a solution where models are arranged in a list. This setup allows the system to seamlessly switch to an alternative model if the current one reaches its usage limit.

5. RESPONSIBLE AI PRACTICES

One of the significant challenges with Large Language Models (LLMs) is the issue of hallucination—responses that include false or misleading information presented as fact. This undermines trust in AI systems. To address this, we have implemented an independent agent dedicated to fact-checking all arguments to ensure alignment with retrieved information. Additionally, all web-sourced information is appropriately cited, with website links provided for transparency.

Ensuring user privacy is another critical aspect of responsible AI practices. To safeguard confidentiality, we have established separate databases for storing private user data and publicly available information. Furthermore, all data is securely cleared after each instance, ensuring no residual information remains.

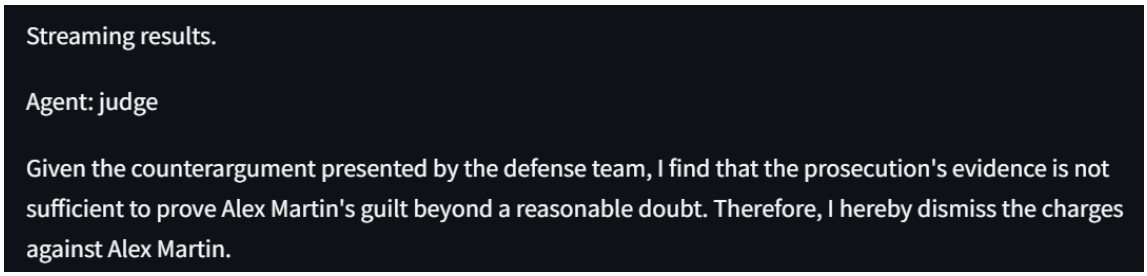
6. USER INTERFACE



Our user interface is built using Streamlit, chosen for its simplicity and clean design. The application features a document upload section, allowing users to add files related to the case, as well as a dedicated input area for entering or editing case details. The **"Run Workflow"** button initiates the workflow, enabling real-time updates. Users can observe the current agent in action, along with its dynamically generated responses.

7. BENCHMARKING/TESTING/RESULTS

Agent / Task	Average Time Taken
Retriever Agent – forming queries, querying from the vector databases and returning the information after analysis.	36.20 seconds



As you can see, our project successfully finishes with its verdict.

8. TAKEAWAY

8.1 LESSONS LEARNT

1. **Effectiveness of Multi-Agent Architecture:** Specialization Enhances Performance: Implementing specialized agents (Lawyer, Prosecutor, Judge, etc.) significantly improved the system's ability to handle complex legal scenarios. Each agent's focused role allowed for more coherent and contextually relevant argument generation, mirroring real courtroom dynamics.
2. **Enhanced Reasoning Capabilities using CoT:** Implementing chain of thought reasoning improved the logical structure and transparency of the arguments generated by each agent. This method enhanced the system's ability to simulate complex legal reasoning and decision-making processes effectively.
3. **Agent Coordination and Concurrency**: Streamlining communication among multiple agents revealed significant coordination and concurrency challenges.

8.2 CHALLENGES FACED

We faced several challenges. Initially, we could only **implement a single retrieval strategy** (cosine similarity) in the pathway, while our goal was to include both BM25 and cosine similarity, which we couldn't accomplish. Additionally, **parsing PDFs** proved difficult even after using the Pathway Parsers, which don't work well without the help of LLMs. Pathway docs, while exhaustive, don't go into much detail about any components and how to implement them with various other components, leaving much to be desired for us developers. **Implementing Chain-of-Thought** (CoT) reasoning was another major hurdle, as ensuring the agent consistently followed CoT logic and produced structured outputs required significant effort. We also encountered technical challenges, such as **integrating Streamlit** with FastAPI and **addressing rate limit** constraints when working with LLM APIs.

8.3 FUTURE SCOPE

1. Modular Agent Expansion: Introducing additional specialized agents to cover specific legal domains (e.g., tax law, intellectual property) will broaden the system's applicability and provide more nuanced legal analyses.
2. Implementing better hybrid retrieval algorithms: We can use improved algorithms like BM25 or TF-IDF along with cosine similarity which have proved to be significantly better in legal cases.
3. Integration with Legal Workflow Tools: Integrating the RAG system with existing legal practice management and workflow tools will streamline its adoption and enhance its utility in real-world legal settings.

9. CONCLUSION

We developed a responsible multi-agent RAG system using Pathway, incorporating advanced techniques to ensure accurate, secure, and resilient information retrieval and argument construction. This system offers a robust framework with error-handling capabilities, effectively simulating real courtroom proceedings.

10. REFERENCES

1. Advantages of using Pathway to Build a real-time RAG chatbot using Google Drive and Sharepoint
2. **"Chain-of-Thought Prompting Elicits Reasoning in Large Language Models",** https://doi.org/10.48550/arXiv.2201.1193
3. Pathway bootcamp material, https://pathway.com/bootcamps/rag-and-llms/coursework/start-here
4. Introducing Meta Llama 3: The most capable openly available LLM to date
5. Introducing Llama-3-Groq-Tool-Use Models - Groq is Fast AI Inference

11. APPENDIX

1. Online Datasets used:

→ The Indian Penal Code
→ Indiankanoon.org – An online database of Indian court hearings

2. Large Language Model Used:

The **LLaMA 3 8B** model is an advanced language model developed by Meta, designed for high-performance and versatile use cases. With 8 billion parameters, it balances computational efficiency and reasoning power, making it well-suited for our legal multi-agent system. Its capabilities include improved contextual understanding and reasoning, critical for analyzing complex legal documents.

The model has been fine-tuned to ensure safety and alignment using methods such as Reinforcement Learning from Human Feedback (RLHF). Additionally, LLaMA 3 emphasizes privacy and bias mitigation in its training data, aligning with responsible AI practices critical for our use case

Benchmark results (source: Introducing Llama-3-Groq-Tool-Use Models - Groq is Fast AI Inference): Llama-3-Groq-8B-Tool-Use: 89.06% overall accuracy (#3 on BFCL at the time of publishing)

3. Embeddings Used:

The embedding model **sentence-transformers/all-MiniLM-L6-v2** was selected for its balance between efficiency and performance. It creates 384-dimensional vector

representations of sentences and paragraphs, making it ideal for tasks like semantic search and clustering. For our legal domain project, this model ensures accurate semantic matching, enabling effective retrieval of similar case laws and legal documents. Its compact size allows for quick processing, even with large datasets, which is crucial for handling the complexity and volume of legal texts.

4. **Retrieval Strategy:**

Our system utilizes cosine similarity as the retrieval strategy, the only method currently available in Pathway, to fetch relevant documents from the vector store. Cosine similarity measures the cosine of the angle between two vectors in a multi-dimensional space, effectively quantifying their alignment. This technique ensures accurate retrieval by assessing contextual relevance between query and document embeddings. It is especially effective for identifying pertinent legal data, enabling precise and well-supported arguments for complex queries.