

---

---

# When LLMs Lie: Building Observable LLMOps for Evaluating EU Legislative RAG - From GPU-Poor to GPU-Rich

---

---

Master of Science in Medialogy (Media Technology) 2025  
Tony Thai Do

Aalborg University Copenhagen  
The Technical Faculty of IT and Design  
CREATE - Department of Architecture, Design and Media Technology,  
Augmented Performance Lab  
A. C. Meyers Vænge 15, 2450, Copenhagen SV, Denmark  
May 26, 2025

Copyright © Aalborg University Copenhagen 2025

Here you can write something about which tools and software you have used for typesetting the document, running simulations and creating figures. If you do not know what to write, either leave this page blank or have a look at the colophon in some of your books.

# Abstract

This thesis explores how to build observable and reliable AI systems for legal text analysis using Large Language Models (LLMs). Focusing on EU legislative documents from MultiEURLEX, a baseline LLM system, a Retrieval-Augmented Generation (RAG) pipeline, and an agentic multi-step variant are developed and compared. The systems are evaluated using a curated gold-standard dataset, quantitative metrics (F1, precision, recall), and qualitative assessments (LLM-as-a-Judge). Tools like Langfuse, LiteLLM provide full observability, tracing, metric logging across local free open-source, open-weights and cloud based proprietary LLM configurations. Key findings reveal direct LLM access outperforms RAG variants due to low retrieval recall, highlighting retrieval is current bottleneck in specific domain RAG application. The work skills and competences demonstrate a full-stack MLOps deployment on AAU's uCloud HPC GPU platform and highlights importance of traceability, and human centered evaluation in trustworthy AI. This thesis and related research contribute both a methodological blueprint and critical insights for operationalizing GenAI in high stakes domains.

**Keywords:** Legal Tech, EU Legislation, Large Language Models (LLMs), Retrieval-Augmented Generation (RAG), LLMOps, Observability, Open-weights models, Local LLM, Cloud LLMs, LLM Evaluation, AI Judge, LLM, Generative AI, MLOps, Software Engineering, HCI, Cloud Computing, AI Engineering, Full Stack Development, DevOps, DevSecFinPlatOps, Software Development.



# Contents

<b>Abstract</b>	<b>iii</b>
<b>Thesis Details</b>	<b>vii</b>
<b>Introduction</b>	<b>1</b>
1 Introduction . . . . .	1
1.1 Problem Statement . . . . .	1
1.2 Research Questions . . . . .	2
1.3 Contribution and Scope . . . . .	2
<b>Literature Review</b>	<b>3</b>
2 Large Language Models and Their Challenges . . . . .	3
3 Evaluating LLM Application Systems Beyond Standard Metrics . . . . .	3
3.1 Retrieval Augmented Generation (RAG) for Enhanced LLM Performance . . . . .	4
3.2 LLMOps Observability for Reliable AI Systems . . . . .	4
4 Acknowledgements . . . . .	5
<b>Methodology</b>	<b>7</b>
5 Data Foundation MultiEURlex Gold Standard Creation . . . . .	7
5.1 Core MLOps LLM infrastructure . . . . .	7
5.2 Baseline: Direct LLM Invocation . . . . .	8
5.3 Poc 2 Annotation Process . . . . .	8
5.4 Poc 2 Retrieval-Augmented Generation (RAG) Pipeline . . . . .	8
5.5 PoC 3 Two Step Agentic System . . . . .	9
5.6 Evaluation Metrics . . . . .	9
5.7 Experimental Execution Environments . . . . .	9
6 Implementation Details . . . . .	10
6.1 System Architecture . . . . .	10
6.2 Core Tech Libraries . . . . .	10

6.3	Proof of Concept Progression . . . . .	10
<b>Results Findings</b>		<b>13</b>
7	Results . . . . .	13
7.1	Poc 1 Article ID extraction local ollama Gemma3 . . . . .	13
7.2	PoC 2 RAG Evaluation Results . . . . .	13
7.3	PoC 3 Agentic RAG Results . . . . .	16
<b>Discussion</b>		<b>17</b>
8	Discussion . . . . .	17
8.1	Recapitulation of Key Findings Analysis . . . . .	17
8.2	Evaluating POC 3 Agentic RAG + Critique Results . . . . .	18
8.3	LLMOps Observability: Aid to Systematic Investigation . . . . .	18
8.4	Difficulties Limitations . . . . .	18
9	Conclusion . . . . .	20
10	Future work . . . . .	20
<b>A Appendix Supplementary Material</b>		<b>27</b>
1	. . . . .	27
1.1	POC2 Annotation System Details . . . . .	27
2	Baseline Evaluation Results . . . . .	27
3	RAG Pipeline Evaluation . . . . .	34
4	System Architecture Diagrams . . . . .	34
4.1	Code Snippets and Configuration . . . . .	34
4.2	RAG Pipeline Execution . . . . .	34
5	Evaluation Metrics Calculation . . . . .	43
6	LLM-as-a-Judge Evaluation . . . . .	44
6.1	Extended Evaluation Metrics . . . . .	45

# Thesis Details

**Thesis Title:** When LLMs Lie: Building Observable LLMOps for Evaluating EU  
Legislative RAG - From GPU-Poor to GPU-Rich

**Subtitle:**

**Keywords:**

**Supervisor:** Cumhur Erkut, Dr. Sc., Aalborg University Copenhagen

**Student:** Tony Thai Do

Aalborg University, Copenhagen May 26, 2025





# Introduction

## 1 Introduction

European Union (EU) legal documentation is complex, this is a challenge for legal professionals and others alike. Tasks such as clause extraction, semantic understanding, compliance checking, and summarizing obligations need attention to detail. The advent of Deep Learning's generative Artificial Intelligence (AI), Large Language Models (LLMs), promises "transformative" increased efficiency and accuracy (Zhao *et al.* 2023). Yet, effective application of these models in high stakes legal domain calls for robust methodologies.

Transformer LLMs promise vast capabilities in Natural Language Processing (NLP). Their application to legal texts, come with challenges as with navigating legal jargon, mitigating "hallucinations", ensuring factual accuracy, managing multilingual contexts is hard (Guha *et al.* 2023; Hong *et al.* 2024). This work confronts "When LLMs Lie" problem. The unreliability of LLMs by investigating observable system for clause type identification in EU legislation.

To enhance LLM performance, ground their responses in factual data, Retrieval-Augmented Generation (RAG) architectures are used (Lewis *et al.* 2020). This research emphasizes LLMOps (Large Language Model Operations) principles, specifically observability, to evaluate application's system behavior, identify performance bottlenecks, and guide iterative software development. Practical deployment considerations motivate a comparative analysis of "GPU-poor" environments on locally sourced, smaller LLMs as Gemma-3 (Kamath *et al.* 2025)) versus "GPU-rich" environments on large, proprietary hyperscalers, someone else's computer of "cloud". This thesis focuses on English language EU legislation to establish a foundational methodology, acknowledging the broader context of AI sustainability and need for efficient practices.

### 1.1 Problem Statement

The problem addressed is the design, implementation, and evaluation of an observable RAG system for clause type identification within English language EU legislative doc-

uments. The research explores the unreliability of LLMs by use of RAG for grounding and LLMOps by integration of observability framework via Langfuse and model abstractions with LiteLLM. Comparative analysis is conducted with a system using local, small scale consumer based LLMs against large scale, cloud based LLMs, with evaluation by AI judge methodology (Zheng *et al.* 2023; Shankar *et al.* 2024).

## 1.2 Research Questions

Primary question I seek to answer is the following:

**Main RQ** : How can observable RAG system, integrating local and cloud based LLMs, be effectively designed, evaluated, and iteratively improved for clause type identification in EU legislative documents?

To address further, following sub questions are investigated:

- SRQ1: How can RAG pipeline be applied to EU legislative texts for clause type identification?
- SRQ2: How do local LLMs (e.g., Gemma 3) compare to proprietary LLMs (e.g., GPT-4.x, Claude 3.x) in terms of accuracy and qualitative assessment by an LLM-as-a-Judge for clause type identification within observable RAG system?

## 1.3 Contribution and Scope

The primary contribution is experimental framework, development and quantitative evaluation of novel observable multi step LLM application for EU legal domain.

- The work experiments integrate Langfuse for observability, tracing of LLM application interactions and LiteLLM for abstracting, swapping with various LLM providers, including local ones.
- comparative performance analysis of local small LLMs against large propitiatory LLMs within RAG and agentic refinement stages.
- Insights into RAG pipeline bottlenecks, particularly low retrieval recall, in the context of EU legal texts.
- Application and discussion of the LLM-as-a-judge paradigm for specialized legal task evaluation.

This research focuses on English language EU legislative documents from the MultiEURLEX dataset (Chalkidis *et al.* 2021).

# Literature Review

**Chapter** provides review of literature essential for contextualizing research on observable application for Retrieval-Augmented Generation (RAG) for clause type classification in EU legislative texts. I cover LLMs, RAG, LLM evaluation methodologies, and role of LLMOps and observability.

## 2 Large Language Models and Their Challenges

The fields of Deep Learning and Human-Computer Interaction (HCI), Artificial Intelligence (AI) advanced with Transformer architecture (Vaswani *et al.* 2017). These models show capabilities in predicting, generating text, which lead to adoption across use cases (Zhao *et al.* 2023). Yet, the models come with limitations. Experts use illustrative terms to describe their behavior, such as "Stochastic parrots" (Bender *et al.* 2021), "Chickenized reverse centaurs", "Sycophant machines". The terms highlight LLMs ability reproduce patterns without factuality, correctness. These pose risks, especially in high stakes domain such as law (Guha *et al.* 2023), where accuracy, reliability are paramount. Trending is also increasingly capable smaller, open-weights models (e.g., Gemma 3 (Kamath *et al.* 2025)), which offer alternatives to large proprietary models regarding accessibility and resource requirements (Huyen 2025).

## 3 Evaluating LLM Application Systems Beyond Standard Metrics

Evaluating LLM answers for specialized or generative tasks, is hard (Huyen 2025). Traditional NLP, ML metrics may be insufficient. For RAG systems, frameworks like RAGAS and DeepEval offer specialized metrics, but their direct application can be extensive (Reddy and contributors 2023; D. Contributors 2024). The "AI Judge" approach, using one LLM to evaluate another's answer (Zheng *et al.* 2023), offers scalability for qualitative assessment but has its own limitations, including potential judge bias and

the challenge of validating the judge itself (Shankar *et al.* 2024; Panickssery *et al.* 2024).

### 3.1 Retrieval Augmented Generation (RAG) for Enhanced LLM Performance

While LLMs possess knowledge learned during pre-training, they often produce factually incorrect or outdated information - hallucinations, and they can be very persuasive (Hong *et al.* 2024; Bender *et al.* 2021). Also they struggle with tasks requiring access to specific, private, or very recent data. Retrieval Augmented Generation (RAG) promises to mitigate these limitations by grounding LLM responses in externally retrieved knowledge, typically sourced with semantic search over vector embeddings (Han *et al.* 2023; Kukreja *et al.* 2023) stored in Vector databases like ChromaDB (Chroma Contributors 2025). While RAG enhances reliability, its effectiveness lies on quality of retrieval and generation components, evaluating failures in RAG pipelines remains a challenge. Recent work explored RAG for commercial enterprise contract clause comparison (Narendra *et al.* 2024), highlighting RAG benefits in legal tech domain.

### 3.2 LLMOps Observability for Reliable AI Systems

Operationalization of LLMs in production (LLMOps) requires robust engineering practises. Central to this is observability, the ability to monitor, trace, log, debug software applications (Charity Majors *et al.* 2022). Tools like Langfuse provide detailed tracing of LLM interactions, inputs, outputs, costs, latencies, prompt management, enabling developers to evaluate issues, understand system behavior, and improve reliability (L. Contributors 2025). This is crucial for building dependable systems, especially when dealing with the inherent unpredictability of LLMs. The user query event metric can be observed, traced through the system, when something goes wrong, teams can ideally know what and where went wrong at exact step of the system (Huyen 2025). This diagnostic ability is important to move beyond "vibe based" evaluation towards more rigorous, data driven software engineering approach to AI Engineering.

**TrustWorthy, Responsible AI** has principles as transparency, accountability, human oversight, specifically with regulatory framework as EU AI Act (Commission 2021). It categorizes AI systems by risk, imposes requirements related to data governance, transparency for users. While observability focuses on system behavior and performance for developers, it is foundational step towards Explainable AI (XAI). Understanding how system arrived at an output, even if it is through tracing intermediate steps in RAG pipeline, contributes to demystifying the black box. The ability to trace data provenance in RAG (knowing which retrieved chunks influenced the output) is a basic form of explainability. Documented are also misbehaviors of LLMs, such as generating harmful content when attacked with "jail-breaking" techniques (Inie *et al.* 2025). Ensuring

AI applications meet ethical, legal standards is intrinsically linked to ability to build, monitor and understand these complex systems.

## 4 Acknowledgements

My deep gratitude goes to supervisor Cumhur Erkut who supported me throughout this work.

DeiC Interactive HPC UCloud (DeiC-AAU-L1-432510) partially supported this work for experiments. I am also grateful for the collaboration and insights by Sykora IT, which enriched the research topic and its practical relevance. Additional cloud credits for Anthropic Claude were provided from Student Builder Program.

I would also thank the Tech meetups in Copenhagen, where discussions flourished with industry experts, researchers. Immense value goes to the human touch in today's state-of-the-art technologies.

**AI-assisted Writing and Coding:** Throughout this project, I made use of gen AI tools to support boilerplate coding and writing workflows. In particular, Copilot Chat in Visual Studio Code Insiders and Windsurf LaTeX editor assisted with code completion and LaTeX editing. For drafting, refining sections of this thesis, I benefited from conversational and text generation models, acknowledge mistakes are my own oversight (Google DeepMind 2025; Anthropic 2025a; Anthropic 2025b; OpenAI 2025).



# Methodology

**Overview** The methodology employed for developing, evaluating the clause extraction systems. It outlines data preparation, architectures of the baseline, RAG, and agentic systems, MLOps tooling for observability, and evaluation metrics used. Components are integrated under a unified observability framework

## 5 Data Foundation MultiEURlex Gold Standard Creation

MultiEURlex (Chalkidis *et al.* 2021) was chosen for structured EU legislative content.

### 5.1 Core MLOps LLM infrastructure

**LiteLLM Proxy for Unified Model Access** proxy provided standardized interface to all LLMs used in research. This included local Gemma-3 series models served on Ollama, external API based models such as OpenAI’s GPT series and Anthropic’s Claude series. This abstraction layer simplified experimentation and model swapping.

**Langfuse for Observability and Evaluation Management** was central LLMOps platform, enabling

- Comprehensive Tracing - detailed, nested traces of all LLM calls (capturing inputs, outputs, model parameters, latency, token counts, and costs) were automatically logged via LiteLLM callbacks and direct Langfuse SDK wrappers (e.g., langfuse.openai). Multi step RAG and agentic pipelines were traced using the @observe decorator
- Dataset Management: English gold standard (eu-clauses-gold-en-v1) was managed as Langfuse Dataset, allowed systematic linking of evaluation runs to specific data items

- **Metrics & Score Logging:** Quantitative metrics (F1, P, R, Retrieval Recall) and qualitative LLM-as-a-Judge scores were programmatically logged to corresponding traces for comparative analysis within the Langfuse UI and via local summary files

## 5.2 Baseline: Direct LLM Invocation

Direct API call of Claude 3.5 Sonnet model via LiteLLM to extract clauses directly from the full document text. LLM was prompted to extract all instances of five target clause types in a single pass, outputting a JSON list. This system served as benchmark for RAG and agentic approaches.

## 5.3 Poc 2 Annotation Process

- Initial "silver" annotations were generated for 100 documents using Claude 3.5 via LiteLLM.
- Additional targeted run of 100 documents identified underrepresented clauses (Definitions, Penalties).
- final manually curated "gold" set of 27 documents was prepared, verifying clause accuracy and boundaries, yielding distributions:
  - Entry into force & Application: 26 instances
  - Obligations: 15
  - Subject matter & Scope: 15
  - Definitions: 12
  - Penalties: 3

## 5.4 Poc 2 Retrieval-Augmented Generation (RAG) Pipeline

Implemented RAG pipeline stages:

1. **Chunking:** LangChain RecursiveCharacterTextSplitter (1000 char chunks, 150 overlap)
2. **Embedding:** OpenAI text-embedding-3-small
3. **Vector Store:** ChromaDB for embedding storage and retrieval
4. **Retrieval:** Top-K semantic similarity retrieval (K=3, K=5 experiments), using generic query



5. **Reader LLMs:** GPT-4.1, GPT-4.1-mini, local Gemma-3 4B QAT, Claude 3.5/3.7 all via LiteLLM. The reader was prompted to extract target clauses from the retrieved context
6. **Output Validation** - LLM JSON output validated against predefined schema

## 5.5 PoC 3 Two Step Agentic System

Agentic system designed used LangGraph, consists of:

- **RAG Node:** Performs initial clause extraction via RAG pipeline, with configurable reader LLMs (e.g., local Gemma-3, GPT-4.1-mini).
- **Critique Node:** Refines outputs through a critique model (Gemma-3 or Claude 3.7).
- The agents execution flow (RAG -> Critique -> END) was traced using Langfuse.

## 5.6 Evaluation Metrics

- **Type F1, Precision, Recall:** Primary quantitative metric was Set-based Type F1, along with Precision and Recall - Measures how well the system identified correct categories of clauses present, ignoring counts or textual accuracy.
- **Retrieval Recall:** Evaluated retrievers effectiveness by matching retrieved chunks with gold clause snippets.
- **LLM-as-a-Judge Score:** Qualitative scores from LLM evaluating semantic accuracy, relevance.  
Ai judge was provided with original document snippet, system's predicted clauses, with gold standard clauses. It was prompted to provide an `llm_judge_overall_quality_score` (0.0-1.0) and a textual `llm_judge_rationale` based on coverage, precision, and accuracy of predictions.
- **Operational Metrics:** Latency and token costs recorded via Langfuse.

## 5.7 Experimental Execution Environments

Ollama local setup ran on GPU: RTX 4060 Ti 16 GB VRAM, CPU: Ryzen 5 2600, WSL2 on Windows with Docker Desktop for Langfuse and LiteLLM.

**AAU uCloud** exploration of local model serving in HPC environment, experiments conducted in AAU uCloud 'Coder' application on a uc1-a10 GPU node. In Coder, Ollama was used to serve Gemma-3 models (4B and briefly 27B), and LiteLLM proxy instance helped access to this local Ollama server and external APIs. Langfuse Cloud was used for tracing these uCloud runs. The setup aimed to demonstrate feasibility of integrating local model inference on uCloud into the MLOps workflow.

## 6 Implementation Details

### 6.1 System Architecture

### 6.2 Core Tech Libraries

**LLMs Used** Gemma3-4B QAT, GPT-4.1-mini, Claude 3.5/3.7 Sonnet, text-embedding-3-small (Kamath *et al.* 2025).

- **Development Environment** was developed in Python 3.11.
- **Key Libraries**
  - Langfuse SDK: For all tracing, dataset management, and scoring.
  - LiteLLM: As the central proxy for all LLM API calls (local Ollama, OpenAI, Anthropic).
  - LangChain (langchain\_text\_splitters, langchain\_openai): Utilized for document chunking and interfacing with embedding models via LiteLLM.
  - ChromaDB: For local vector storage and retrieval.
  - LangGraph: For orchestrating the two step agentic system.
  - datasets Hugging Face For loading MultiEURlex.
  - jsonschema For validating LLM JSON outputs.
  - Pandas: For local analysis of evaluation summaries.

### 6.3 Proof of Concept Progression

- **PoC 1 (Foundational Setup):** Focused on establishing the local development environment, integrating LiteLLM and Langfuse for basic tracing with a simple "Article ID" extraction task using Gemma-3 via Ollama. MLflow was used for initial high level parameter logging in this phase.
- **PoC 2 (RAG Development & Evaluation):** Involved silver/gold annotation generation (using Claude 3.5), development of the RAG pipeline (`poc2_rag_pipeline.py`),

and systematic evaluation against the gold set using `poc2_baseline_eval.py` and `poc2_run_evaluation.py`, with metrics calculated by `poc2_eval/poc2_metrics.py` and LLM judging by `poc2_llm_judge.py`.

- **PoC 3 (Agentic System & uCloud Exploration):** Implemented a two step RAG+Critique agent (`poc3_agentic_extractor.py`) , evaluated its final outputs using `poc3_run_agent_eval.py`. Conceptual exploration, limited testing of local model serving (Gemma-3 via Ollama) were conducted within the AAU uCloud Coder environment, with traces sent to hosted Langfuse Cloud.



# Results Findings

## 7 Results

### 7.1 Poc 1 Article ID extraction local ollama Gemma3

PoC Established prerequisites for POC 2 evaluations by successfully gluing, testing integration of the LiteLLM proxy for accessing various language models, setting up Langfuse for comprehensive tracing of LLM calls and pipeline steps. This ensured that the more complex RAG pipeline and LLM evaluation components in POC2 could connect to necessary services and their operations would be monitored, observed and logged. The number of articles extracted varied per run, upon review, all were correct.

### 7.2 PoC 2 RAG Evaluation Results

**Proof of Concept 2 objective** was to develop, evaluate a Retrieval-Augmented Generation (RAG) pipeline for clause type identification within English EU legislative documents from MultiEurlex (Chalkidis *et al.* 2021). I compared RAG implementation against baseline of directly invoking LLM and assessed performance on different RAG configurations, various RAG LLM readers, both local and LLM API based. Retrieval parameter adjusted was top-K number of retrieved chunks. Evaluation used by myself selected 27 "gold" annotations from initial 100 "silver" annotations of clause types annotated by Claude 3.5 sonnet. The custom metrics are set-based Type F1, Precision, Recall, Retrieval Recall, tracked and managed using Langfuse LLM observability

**Table 1:** Performance Comparison of Models

Model	F1	Prec.	Recall	Retr.	Success
Baseline (Claude)	0.893	0.895	0.895	N/A	25/27
RAG (GPT-4.1)	0.698	0.641	0.830	0.173	27/27
RAG (GPT-4.1-mini)	0.686	0.654	0.750	0.093	27/27

platform locally set up.

Table 2: Complete RAG System Performance Analysis

System	$\Delta$ F1	$\Delta$ Precision	$\Delta$ Recall	$\Delta$ Retrieval Recall
<i>Baseline (No RAG)</i>				
Claude 3.5 (Direct)	<b>0.893</b>	0.895	0.895	–
<i>RAG Systems</i>				
GPT-4.1 (K=3)	0.698	0.641	0.830	0.173
GPT-4.1-mini (K=3)	0.686	0.654	0.750	0.093
GPT-4.1-mini (K=5)	0.686	0.654	0.750	0.136
Gemma-3 (K=3)	<b>0.735</b>	0.611	<b>0.963</b>	0.093
Gemma-3 (K=5)	0.614	0.519	0.787	0.136
Claude 3.7 (K=3)	0.686	0.654	0.750	0.093
Claude 3.7 (K=5)	0.686	0.654	0.750	0.136

Embedding: `text-embedding-3-small`. Success rate: 27/27 for all systems. Best RAG F1: Gemma-3 (K=3).

Table 3: Key Performance Insights by Approach

Approach	Best F1	Key Finding
Baseline (No RAG)	0.893	Direct LLM outperforms all RAG
RAG (Best)	0.735	Gemma-3 (K=3) leads RAG systems
Retrieval Impact	0.093–0.173	Low retrieval recall limits performance

**LLM-as-Judge Evaluation**    Qualitative assessment using Claude 3.5 as judge showed patterns:

- **GPT-4.1-mini (K=5):** Judge score 0.269 (F1: 0.686) - outputs often irrelevant despite correct format
- **Gemma-3 (K=5):** Judge score 0.556 (F1: 0.614) - better semantic understanding of legal context

**Table 4:** Performance Summary: Quantitative vs Qualitative Assessment

System Type	Best F1	Best Config	Judge Score	Assessment
Baseline	0.893	Claude 3.5 Direct	N/A	Gold standard
RAG-only	0.735	Gemma-3 (K=3)	N/A	Best quantitative
RAG-judged	0.686	GPT-4.1-mini (K=5)	0.269	Format over content
RAG-judged	0.614	Gemma-3 (K=5)	0.556	Better semantics

Comparison shows judge scores don't always align with F1 performance, suggesting complementary evaluation approaches needed.

7.3 PoC 3 Agentic RAG Results

The two-step agent RAG node Critique node. Specify models used for each component in the tested configurations.

Table 5: Agent System Performance with LLM-as-Judge Evaluation

Agent Configuration	F1	Prec	Rec	Judge Score
Gemma3 RAG + Gemma3 Critique	0.430	0.509	0.432	0.683
Gemma3 RAG + Claude 3.7 Critique	0.164	0.138	0.216	0.376
GPT-4.1-mini RAG + Claude 3.7 Critique	0.247	0.264	0.253	0.347



# Discussion

## 8 Discussion

### 8.1 Recapitulation of Key Findings Analysis

**Experimental evaluation revealed** The direct to LLM baseline system Claude 3.5 outperformed all implemented RAG and agentic configurations in identifying clause types. Baseline had an average Type Set F1 of 0.893, Precision of 0.895, and Recall of 0.895 across 27 gold documents.

Of all RAG configurations, retrieval recall is consistently low (max around 0.173). This means 80-90% of the time, the chunks containing the gold clauses are not being retrieved. No RAG reader LLM overcame this issue, indicating retrieval is the bottleneck. Qualitative assessment via LLM-as-a-Judge underscored challenges, frequently revealing content inaccuracies in RAG and agent outputs despite potentially moderate F1 scores.garb

**Retrieval Bottleneck Failures** The prompt query for RAG reader model "Extract legal clauses such as ..." might be too broad and missed specific keywords. Length to actual text of desired clauses, especially for rarer types like "Penalties" or "Definitions" for this domain without better query formulation or re-ranking. Embedding model limitations, text-embedding-3-small is generally good, the nuances and terminology of EU legal text might not be captured for effective similarity search. The implemented chunking strategy used Langchain's RecursiveCharacterTextSplitter 1000 char chunk size over 150 char overlap perhaps split awkwardly, making them harder to retrieve or less relevant individual chunks.

**Reader Model Differences** Among RAG readers, Gemma-3 4B K=3 surprisingly outperformed the API models (GPT-4.1-mini K=3, Claude 3.5 K=3) in F1 despite similar (low) retrieval recall. Increasing Top K Chunks from 3 to 5 improved retrieval recall slightly but did not consistently improve (sometimes worsened) the final F1, suggesting reader models struggled with increased noisy retrieved context.

## 8.2 Evaluating POC 3 Agentic RAG + Critique Results

This agentic approach did not improve performance over the best RAG only configurations and remained significantly below the baseline. Agent A (Gemma3 RAG + Gemma3 Critique) against Agent B (Gemma3 RAG + Claude Critique), a supposedly better critique model Claude lead to worse F1, indicates, that Gemma3 RAGs output interpretation of "good" is wrong upon review in Sessions Trace. This is supported by an API based RAG reader, GPT-4.1-mini followed by Claude critique (Agent C), the agent's F1 (0.247) remained low.

Langfuse traces of agent executions with Langgraph visualized the two step process. Analysis of the critique text and final clauses showed the critique rational such as, the output of previous step was wrong.

Overall on agent effectiveness, the agents still underperforms baseline direct call. Likely because foundational RAG step was weak, garbage in, garbage out.

**Insights with LLM-as-a-Judge** AI judge highlighted inside Langfuse trace, the contents inaccuracies in RAG outputs. For example, for document 32012D0637, the judge noted: "The machine's predicted clauses are largely irrelevant... clauses are from a completely different fishing regulation..."

## 8.3 LLMOps Observability: Aid to Systematic Investigation

The adoption of MLOps principles, tooling was central here. Langfuse was instrumental for tracking all experiments. It enabled systematic evaluation (Baseline, RAG variations, Agent runs) with varying parameters and configurations. Managing the gold standard dataset, I linked evaluation runs under sessions, traces, userID to it. It helped logging diverse custom metrics (quantitative F1, P, R, Retrieval Recall, LLM Judge scores, native operational metrics like latency/cost from traces). The platform's ability to visualize execution flows, like the LangGraph agent trace, helped me understand internal states and outputs of each component, LLM call. This detailed observability was key to debugging issues, finding critical retrieval bottleneck in RAG systems. The use of LiteLLM provided a unified interface for accessing various local (Ollama served Gemma3) and API based (OpenAI, Anthropic) models, simplifying process of swapping components for comparative experiments. The practice of saving local JSONL summaries of evaluation results complemented Langfuse, enabled flexible data analysis.

## 8.4 Difficulties Limitations

**Difficulties POC 2 Annotation** From the main 100 silver set annotation run I initially lacked 2 Clause types, specifically Definitions and Penalties. These had to be found by keywords in the validation split data set and another run of 100 document

annotation was made for candidates for Definitions and Penalties clause types, which is where 12 Definitions, 3 Penalties clause types were found.

**Limitations** With  $N=27$ , small size per condition is a limitation. The primary quantitative metric, Type Set F1, assesses the presence of clause types rather than the textual accuracy or semantic completeness of the extracted clause content. While complemented by LLM-as-a-Judge, more granular text-based metrics (e.g., ROUGE, BERTScore, or span-F1) would provide further insights in future work.

While the system was designed with multilingual evaluation in mind (e.g. using multilingual embedding models conceptually), extensive multilingual evaluation was beyond the scope of PoC, as I encountered parsing error with German annotation splits.

Due to the exploratory nature and limited size of the evaluation set per condition, formal statistical significance tests were not performed. Comparisons are based on observed mean differences.

### HPC GPU AAU uCloud vLLM / Ollama

To explore feasibility and operational aspects of utilizing local LLMs within an HPC environment, components of the agentic system were executed on Aalborg University's AAU/K8s uCloud infrastructure. The hosting environment chosen was the "Coder" interactive application available on uCloud, launched on a uc1-a10 GPU node. This provided a persistent VS Code server environment with terminal access, ability to mount project files from Ceph storage (/work). Dependencies were managed using uv within a Python 3.11 virtual environment.

The gemma3-4b-it-qat model and experiments with bigger gemma3:27b-it-qat was served using vLLM and mainly Ollama, installed and ran directly within the Coder job's environment. Ollama listened on localhost:11434 (internal to the Coder job), making the Gemma3 model accessible to other processes within the same job. This approach mirrored the local development setup.

**Observability with Langfuse Cloud** Given unavailable Docker in the HPC Coder app flavor, direct self-hosting of the full Langfuse stack via Docker Compose inside the Coder job was not pursued for this PoC. The setup captured traces to Langfuse Cloud hosted environment, while evaluating the system, I encountered parsing errors in the traces. Further investigations. This MLOps approach on uCloud, while simplified for this PoC, lays groundwork for more advanced HPC GPU deployments serving models using optimized engines like vLLM.

### User Interview

Brief Interview was conducted with the collaboration company's founder. at the end of PoC experiments, I asked if the clause type or EU legislation RAG system would be

relevant. I was confirmed there is potential, but further iteration on more product of user centered use case would be needed.

## 9 Conclusion

This work went on developing, evaluating LLM systems for EU legislative documents, emphasizing an LLMops approach centered on observability. The investigation compared a direct LLM baseline, various RAG pipeline configurations, and a two step agentic (RAG + Critique) system, utilizing both local and API inferred based LLMs. The experimental results consistently demonstrated that, for the specific task of identifying the presence of predefined clause types within the 27-document English gold standard, a powerful baseline LLM (Claude 3.5 Sonnet) with access to the full document context (F1 0.893) outperformed all RAG and agentic configurations. The primary limiting factor for all RAG-based approaches was identified as a retrieval bottleneck, with retrieval recall scores consistently below 18%. This inability to surface relevant context to the reader LLM rendered even advanced reader models, later agentic critique steps ineffective in matching baseline performance. While local Gemma 3 model showed competitive F1 scores as a RAG reader in certain low context scenarios (K=3, F1 0.735), this did not generalize with increased context. The LLM-as-a-Judge component proved valuable, offering qualitative insights that often highlighted content inaccuracies and irrelevance in RAG - agent outputs, which were not fully captured by the set-based F1 metric. Ultimately, this work shows that while RAG and agentic architectures hold promise for enhancing LLM applications in domain as law, their success is dependent on performance of each component, particularly information retrieval and the person in front of monitor. The LLMops framework, centered around observability and systematic evaluation, was instrumental in finding traces of RAG inefficiencies and understanding the shades of performance in different LLM configurations. The findings show that simply adding more LLM steps or more powerful models downstream may not yield desired improvements and implementing RAG is not straightforward for the first time.

## 10 Future work

Findings open ways for future research, development aimed at creating more robust and reliable LLM systems:

**Enhance RAG Retrieval Performance** Future work should explore query formulation techniques variables, top K, prompt engineering techniques manage evaluation query text. Different baseline model than annotation model, there is LLM self preference (Shankar *et al.* 2024). Research can evaluate more retrieval strategies (e.g., Par-

entDocumentRetriever, re-ranking algorithms, hybrid search combining semantic and keyword approaches), Optimize chunking strategies for legal document structures.

**Robust Multilingual** Evaluation and adaptation of pipeline for more languages.

**Optimized Local Model Serving on HPC, Private Cloud** can systematically deploy, benchmark larger local models e.g., Gemma 12B/27B or other open-weight models using optimized serving frameworks like vLLM. Integrate these and compare performance against local Ollama setups, API models.

**User Feedback Loop Human in the loop systems** : Further research can conceptualize legal professionals use of the system with interactive UI, provide feedback on extractions and help the actual user, potentially fine tuning models with popular LoRa technique.

**Scalable deployment** can explore CI/CD pipeline for model agent updates. For production grade operationalization deployment can go for hyperscale cloud platforms using Infrastructure as Code (Terraform/ OpenTofu), Langfuse provides such templates.



# References

- [1] W. X. Zhao *et al.*, “A survey of large language models,” 2023, Publisher: arXiv Version Number: 16. DOI: 10.48550/ARXIV.2303.18223. [Online]. Available: <https://arxiv.org/abs/2303.18223> (visited on 03/21/2025).
- [2] N. Guha *et al.*, *LegalBench: A collaboratively built benchmark for measuring legal reasoning in large language models*, Aug. 20, 2023. DOI: 10.48550/arXiv.2308.11462. arXiv: 2308.11462[cs]. [Online]. Available: <http://arxiv.org/abs/2308.11462> (visited on 03/11/2025).
- [3] G. Hong *et al.*, *The hallucinations leaderboard – an open effort to measure hallucinations in large language models*, Apr. 17, 2024. DOI: 10.48550/arXiv.2404.05904. arXiv: 2404.05904[cs]. [Online]. Available: <http://arxiv.org/abs/2404.05904> (visited on 04/08/2025).
- [4] P. Lewis *et al.*, “Retrieval-augmented generation for knowledge-intensive nlp tasks,” *ArXiv*, vol. abs/2005.11401, 2020. [Online]. Available: <https://api.semanticscholar.org/CorpusID:218869575>.
- [5] G. T. A. Kamath *et al.*, “Gemma 3 technical report,” *ArXiv*, vol. abs/2503.19786, 2025. [Online]. Available: <https://api.semanticscholar.org/CorpusID:277313563>.
- [6] L. Zheng *et al.*, “Judging LLM-as-a-judge with MT-bench and chatbot arena,” *ArXiv*, Jun. 9, 2023. [Online]. Available: <https://www.semanticscholar.org/paper/Judging-LLM-as-a-judge-with-MT-Bench-and-Chatbot-Zheng-Chiang/a0a79dad89857a96f8f71b14238e5237cbfc4787> (visited on 03/20/2025).
- [7] S. Shankar, J. D. Zamfirescu-Pereira, B. Hartmann, A. G. Parameswaran, and I. Arawjo, *Who validates the validators? aligning LLM-assisted evaluation of LLM outputs with human preferences*, Apr. 18, 2024. DOI: 10.48550/arXiv.2404.12272. arXiv: 2404.12272[cs]. [Online]. Available: <http://arxiv.org/abs/2404.12272> (visited on 02/26/2025).

- [8] I. Chalkidis, M. Fergadiotis, and I. Androutsopoulos, “Multieurlex – a multilingual and multi-label legal document classification dataset for zero-shot cross-lingual transfer,” in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021. [Online]. Available: <https://arxiv.org/abs/2109.00904>.
- [9] A. Vaswani *et al.*, “Attention is all you need,” in *Neural Information Processing Systems*, 2017. [Online]. Available: <https://api.semanticscholar.org/CorpusID:13756489>.
- [10] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, “On the dangers of stochastic parrots: Can language models be too big?” *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, 2021. [Online]. Available: <https://api.semanticscholar.org/CorpusID:262580630>.
- [11] C. Huyen, *AI Engineering: Building Applications with Foundation Models*. O’Reilly Media, Jan. 2025, p. 532, ISBN: 978-1098166304. [Online]. Available: <https://www.oreilly.com/library/view/ai-engineering/9781098166298/>.
- [12] T. K. Reddy and contributors, *Ragas: Retrieval augmented generation assessment*, Version 0.1.7, 2023. [Online]. Available: <https://github.com/explodinggradients/ragas> (visited on 05/25/2025).
- [13] D. Contributors, *Deepeval: Llm evaluation framework*, Version 0.17.16, 2024. [Online]. Available: <https://github.com/confident-ai/deepeval> (visited on 05/25/2025).
- [14] A. Panickssery, S. R. Bowman, and S. Feng, “LLM evaluators recognize and favor their own generations,” 2024, Publisher: arXiv Version Number: 1. DOI: 10.48550/ARXIV.2404.13076. [Online]. Available: <https://arxiv.org/abs/2404.13076> (visited on 04/25/2025).
- [15] Y. Han, C. Liu, and P. Wang, “A comprehensive survey on vector database: Storage and retrieval technique, challenge,” *ArXiv*, vol. abs/2310.11703, 2023. [Online]. Available: <https://api.semanticscholar.org/CorpusID:264289073>.
- [16] S. Kukreja, T. Kumar, V. Bharate, A. Purohit, A. Dasgupta, and D. Guha, “Vector databases and vector embeddings-review,” *2023 International Workshop on Artificial Intelligence and Image Processing (IWAIP)*, pp. 231–236, 2023. [Online]. Available: <https://api.semanticscholar.org/CorpusID:268545576>.
- [17] Chroma Contributors, *Chroma: The open-source embedding database*, Version 1.0.10, 2025. [Online]. Available: <https://www.trychroma.com/> (visited on 05/25/2025).



- [18] S. Narendra, K. Shetty, and A. Ratnaparkhi, “Enhancing contract negotiations with LLM-based legal document comparison,” in *Proceedings of the Natural Legal Language Processing Workshop 2024*, N. Aletras, I. Chalkidis, L. Barrett, C. Goanță, D. Preoțiuc-Pietro, and G. Spanakis, Eds., Miami, FL, USA: Association for Computational Linguistics, Nov. 2024, pp. 143–153. DOI: 10.18653/v1/2024.nllp-1.11. [Online]. Available: <https://aclanthology.org/2024.nllp-1.11/> (visited on 03/25/2025).
- [19] Charity Majors, Liz Fong-Jones, and George Miranda, *Observability Engineering*. O’Reilly Media, Inc., May 2022, ISBN: 978-1-4920-7643-8. [Online]. Available: <https://learning.oreilly.com/library/view/observability-engineering/9781492076438/> (visited on 03/24/2025).
- [20] L. Contributors, *Langfuse: Open source llm observability platform*, Version 2.0.0, 2025. [Online]. Available: <https://github.com/langfuse/langfuse> (visited on 05/26/2025).
- [21] E. Commission, *Proposal for a regulation laying down harmonised rules on artificial intelligence (artificial intelligence act)*, COM/2021/206 final, 2021. [Online]. Available: <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52021PC0206>.
- [22] N. Inie, J. Stray, and L. Derczynski, “Summon a demon and bind it: A grounded theory of LLM red teaming,” *PLOS ONE*, vol. 20, no. 1, C. S. Lee, Ed., e0314658, Jan. 15, 2025, ISSN: 1932-6203. DOI: 10.1371/journal.pone.0314658. [Online]. Available: <https://dx.plos.org/10.1371/journal.pone.0314658> (visited on 05/24/2025).
- [23] Google DeepMind, *Gemini pro*, 2025. [Online]. Available: <https://deepmind.google/gemini> (visited on 05/25/2025).
- [24] Anthropic, *Claude 3.7*, 2025. [Online]. Available: <https://claude.ai> (visited on 05/25/2025).
- [25] Anthropic, *Claude 4*, 2025. [Online]. Available: <https://claude.ai> (visited on 05/25/2025).
- [26] OpenAI, *Chatgpt (may 2025 version)*, 2025. [Online]. Available: <https://chat.openai.com/chat> (visited on 05/25/2025).

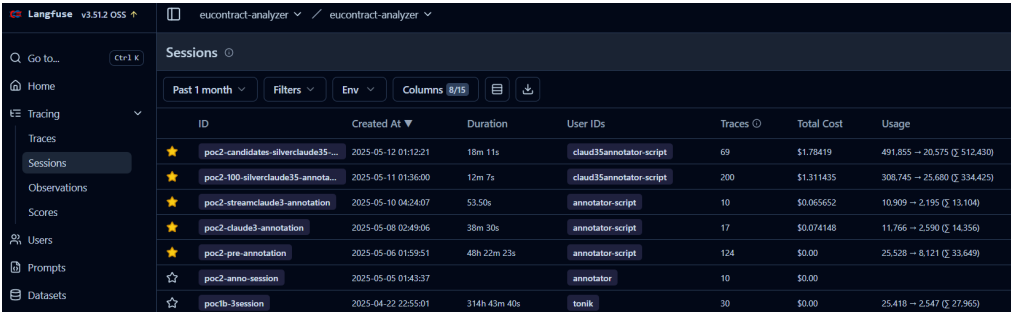


# Paper A

## Appendix Supplementary Material

### 1

#### 1.1 POC2 Annotation System Details



ID	Created At	Duration	User IDs	Traces	Total Cost	Usage
poc2-candidates-silverclaude35-...	2025-05-12 01:12:21	18m 11s	claud35annotator-script	69	\$1.78419	491,855 → 20,575 (512,430)
poc2-100-silverclaude35-annota...	2025-05-11 01:36:00	12m 7s	claud35annotator-script	200	\$1.311435	308,745 → 25,680 (334,425)
poc2-streamclaude3-annotation	2025-05-10 04:24:07	53.50s	annotator-script	10	\$0.065652	10,909 → 2,195 (13,104)
poc2-claude3-annotation	2025-05-08 02:49:06	38m 30s	annotator-script	17	\$0.074148	11,766 → 2,590 (14,356)
poc2-pre-annotation	2025-05-06 01:59:51	48h 22m 23s	annotator-script	124	\$0.00	25,528 → 8,121 (33,649)
poc2-anno-session	2025-05-05 01:43:37		annotator	10	\$0.00	
pocfb-3session	2025-04-22 22:55:01	314h 43m 40s	tonik	30	\$0.00	25,418 → 2,547 (27,965)

Fig. A.1: Poc2 Annotation - Sessions Traces, Total Costs, Token Usage

### 2 Baseline Evaluation Results

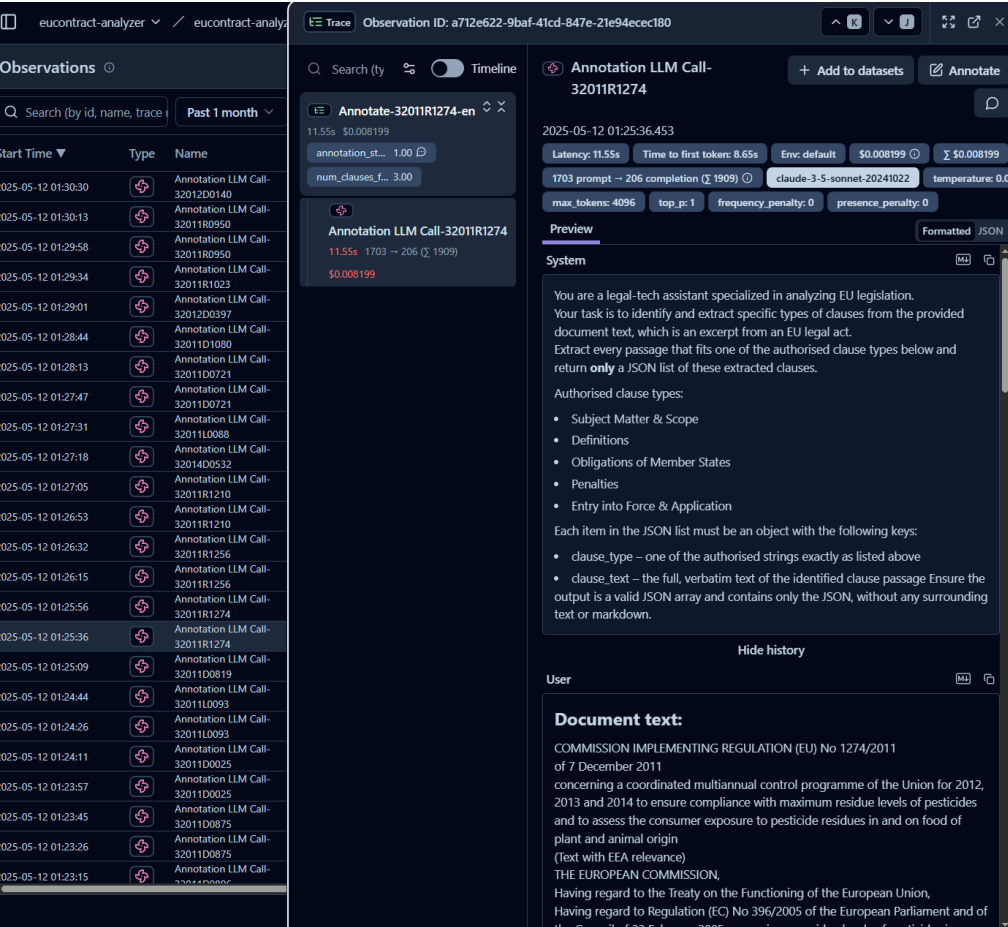
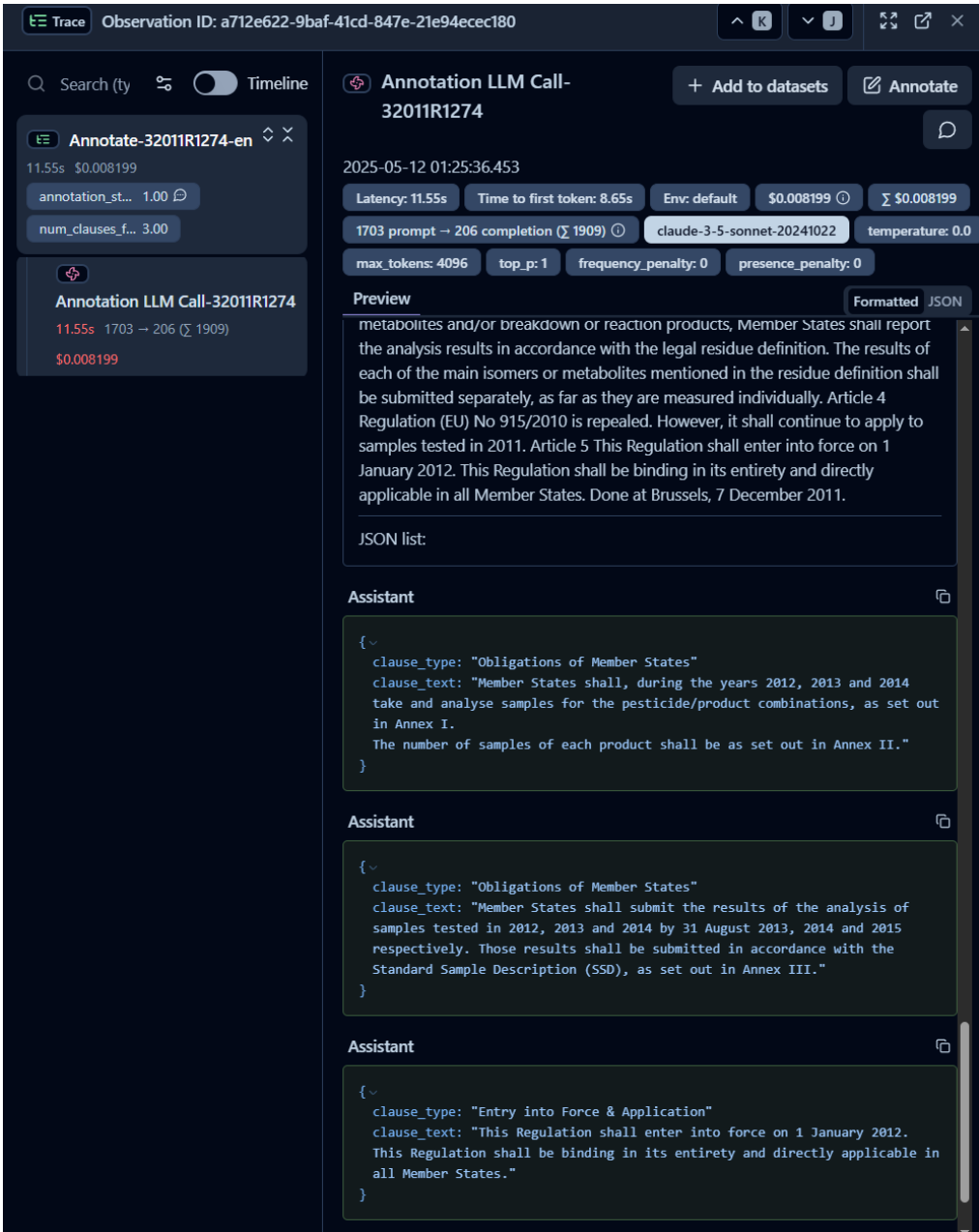


Fig. A.2: POC2 Annotation System User Prompts - Langfuse trace showing the interaction flow between user inputs and the annotation system



**Fig. A.3:** POC2 Annotation Observation Output - System response generation and clause type predictions

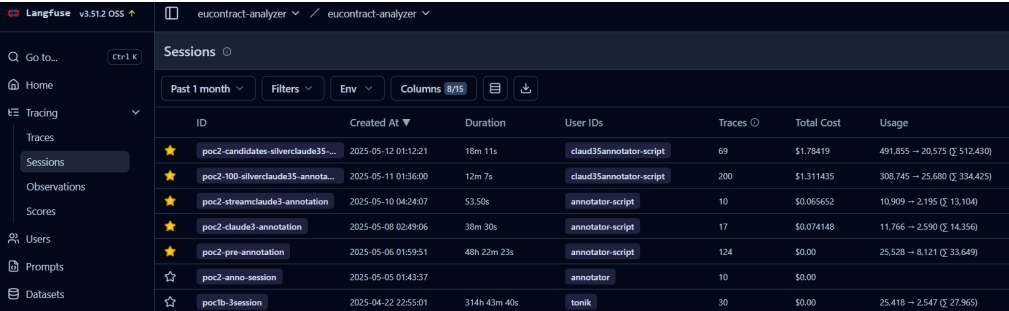


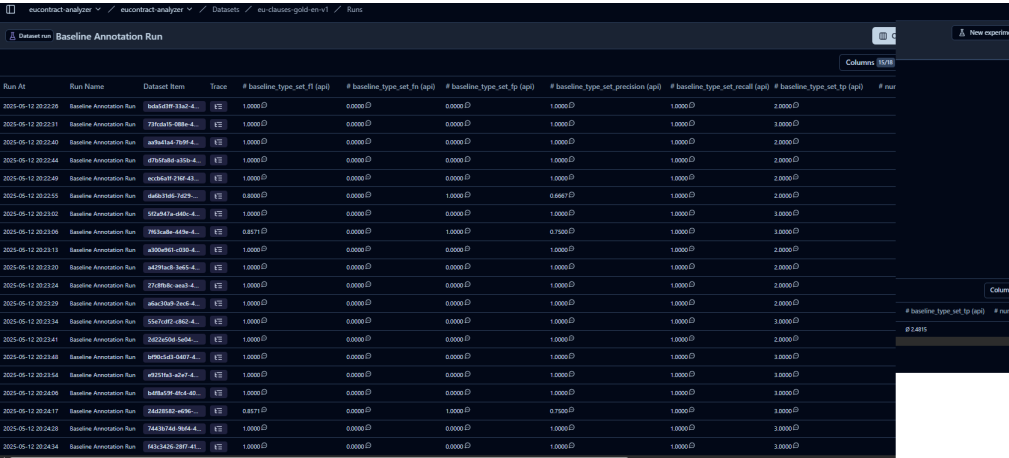
Fig. A.4: POC2 Annotation Sessions Overview - Total costs, token usage, and trace summaries



Fig. A.5: POC2 Dataset Baseline Evaluation Scores - First run performance metrics across different clause types



**Fig. A.6:** POC2 Baseline Evaluation Metrics - Average latency, costs, F1-score, precision, recall, and confusion matrix components (TP: True Positive, FP: False Positive, FN: False Negative)



**Fig. A.7:** POC2 Baseline Evaluation Dataset Runs - Comprehensive view of evaluation runs across the gold standard dataset

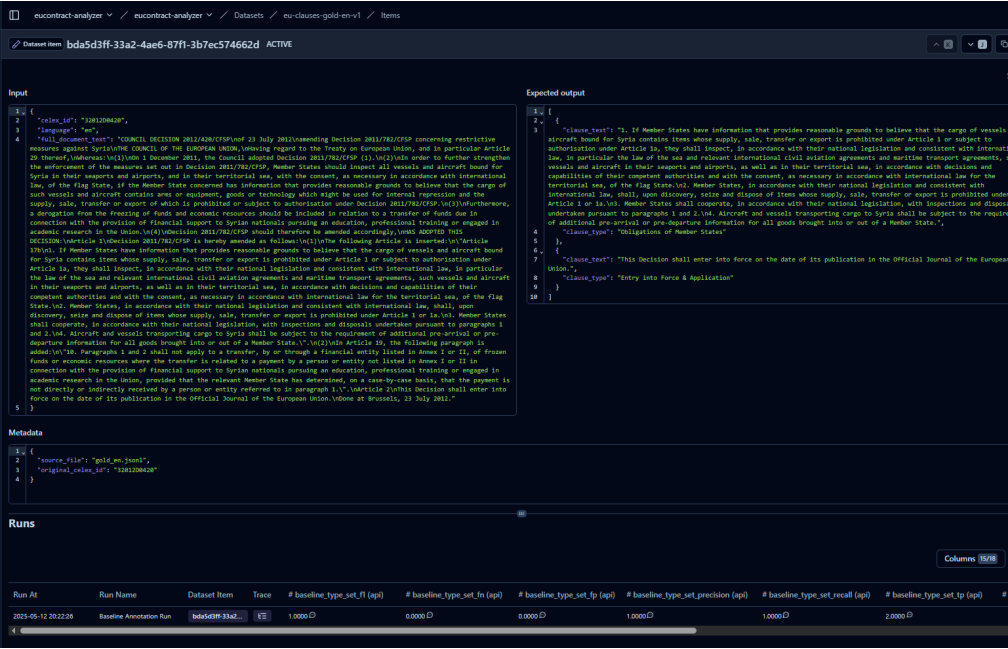
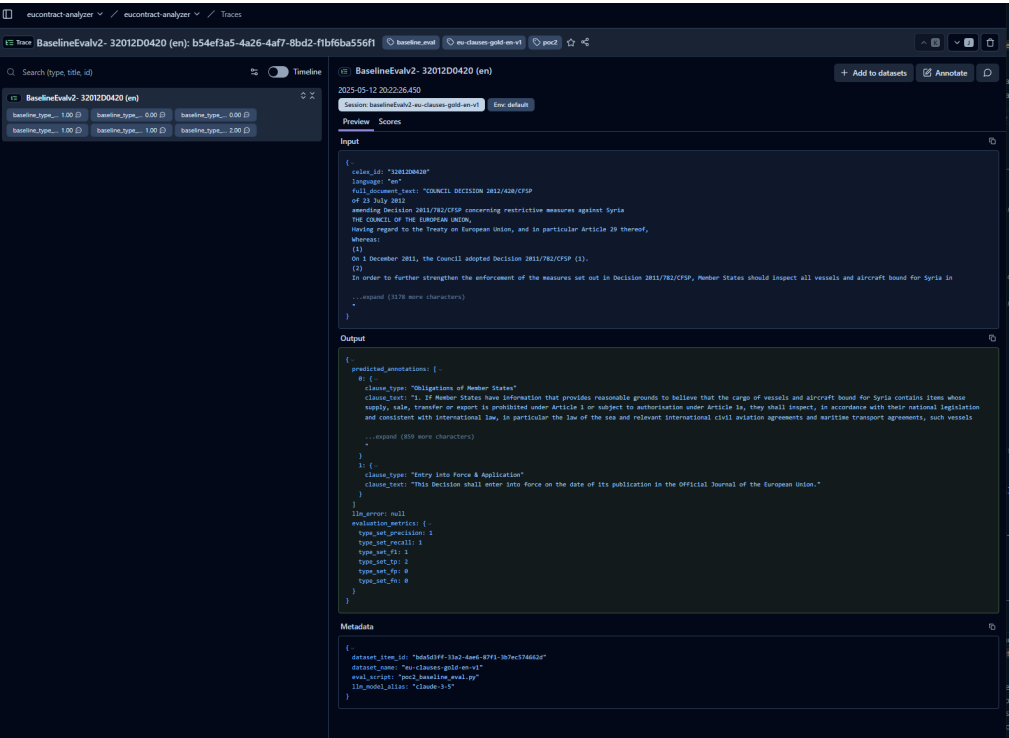


Fig. A.8: POC2 Baseline Evaluation Dataset Items - Individual document processing results and performance breakdown





**Fig. A.9:** POC2 Baseline Evaluation Trace Details - Specific trace metrics and output analysis for individual processing steps

### 3 RAG Pipeline Evaluation

## 4 System Architecture Diagrams

### 4.1 Code Snippets and Configuration

#### 4.2 RAG Pipeline Execution

# poc2\_rag\_pipeline.py

```
@observe()
def run_rag_clause_extraction(
    document_text: str,
    document_id: str,
    language: str = "en",
    session_id: Optional[str] = None,
    user_id: Optional[str] = None,
    reader_model_alias_param: Optional[str] = None,
    litellm_proxy_url_param: Optional[str] = None,
    embedding_model_alias_param: Optional[str] = None,
) -> Dict[str, Any]:
    # Determine actual models and proxy to use, prioritizing passed parameters
    )
    rag_trace_id = langfuse_context.get_current_trace_id()
    langfuse_context.update_current_trace(
        name=f"RAG-ClauseExtrc_Pipe-{current_reader_model}",
        metadata={
            "document_id": document_id,
            "language": language,
            "reader_model": current_reader_model,
            "embedding_model": current_embedding_model,
            "litellm_proxy_url": current_proxy_url,
            "chunk_size": CHUNK_SIZE,
            "chunk_overlap": CHUNK_OVERLAP,
            "top_k_retrieval": TOP_K_RETRIEVAL,
        },
        session_id=session_id,
        user_id=user_id,
    )

    # 1. Chunking
```

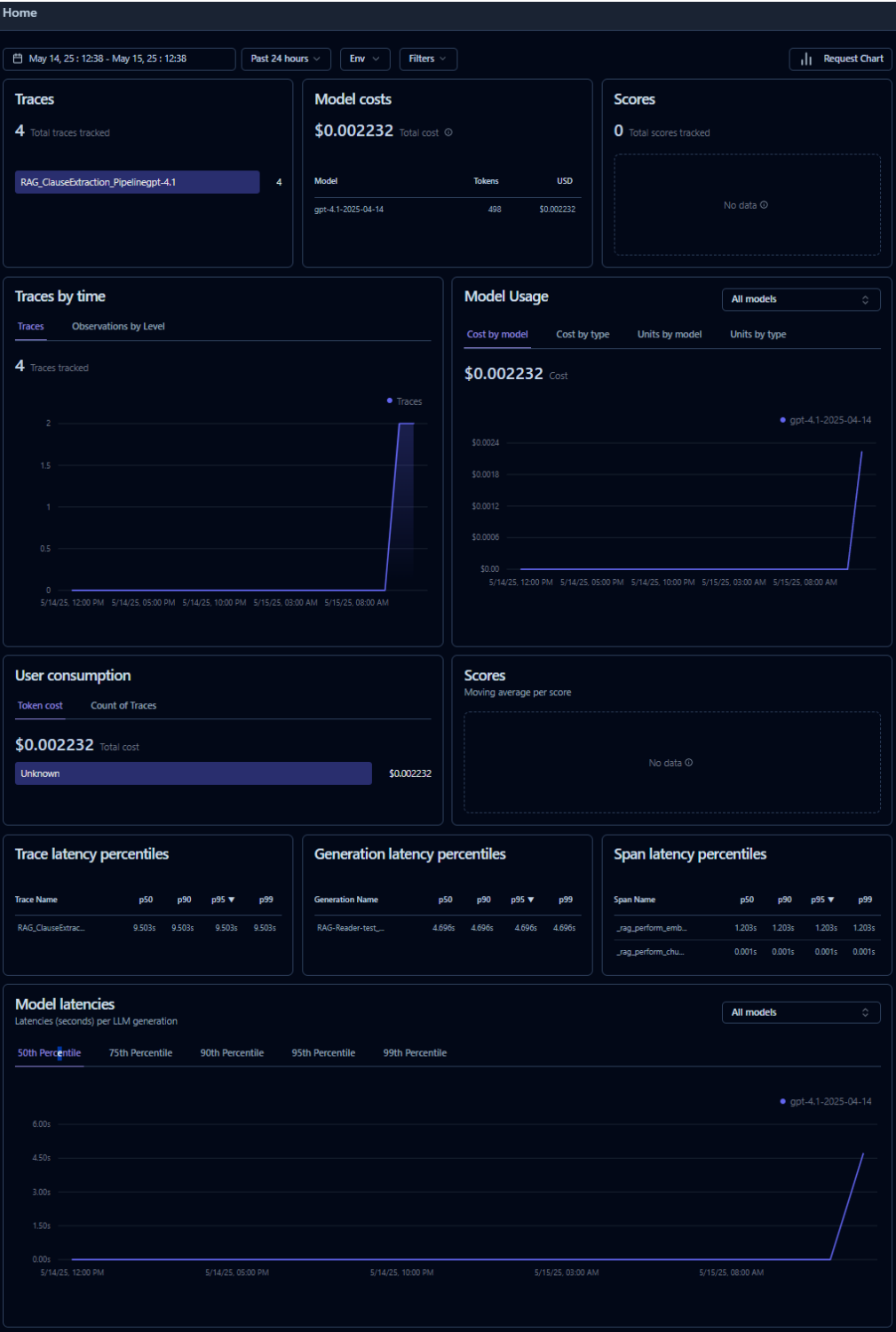


Fig. A.10: RAG Evaluation Test Run - Trace latency percentiles and span analysis showing retrieval and generation performance

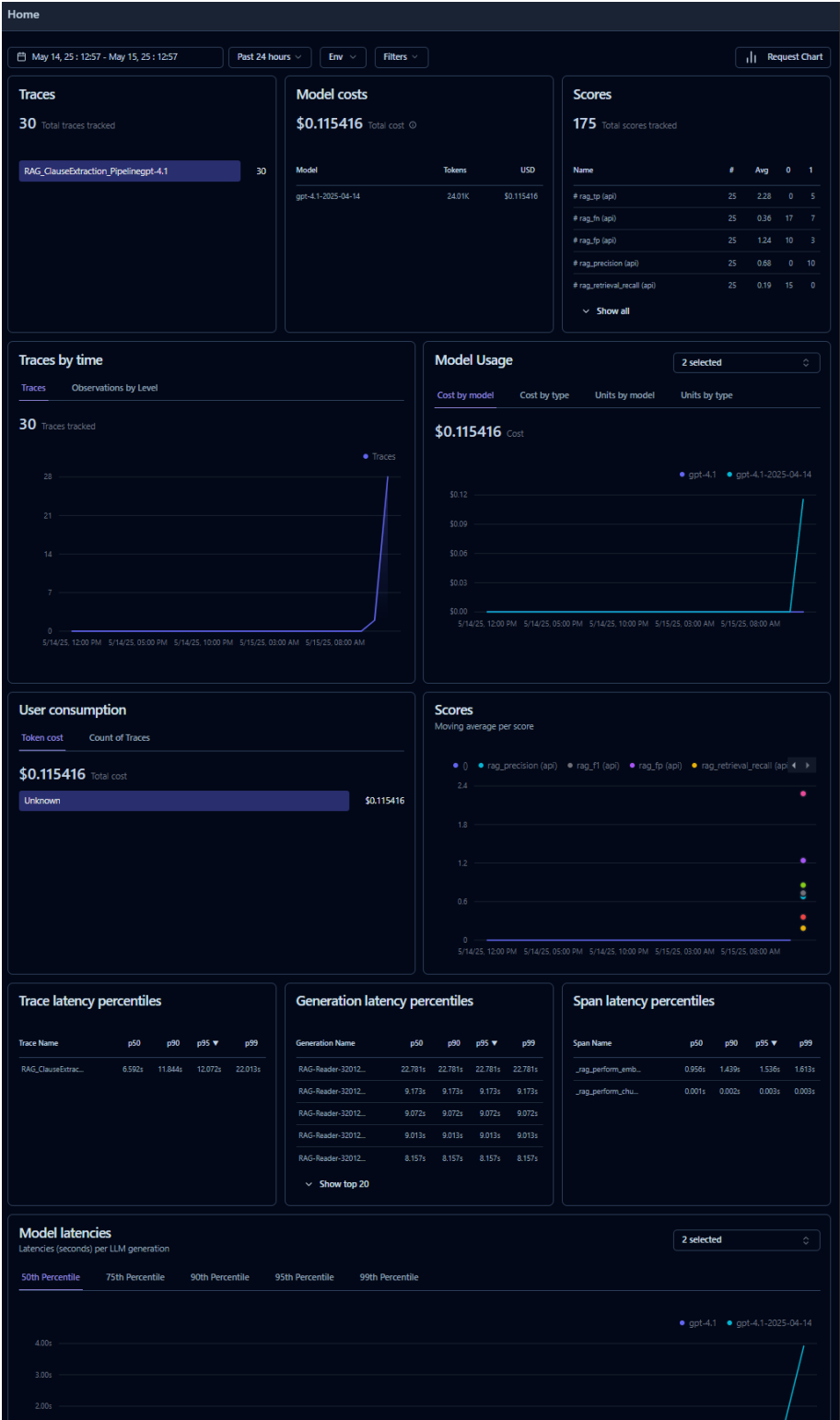


Fig. A.11: RAG Pipeline GPT-4.1 Overview - End-to-end evaluation results and performance summary

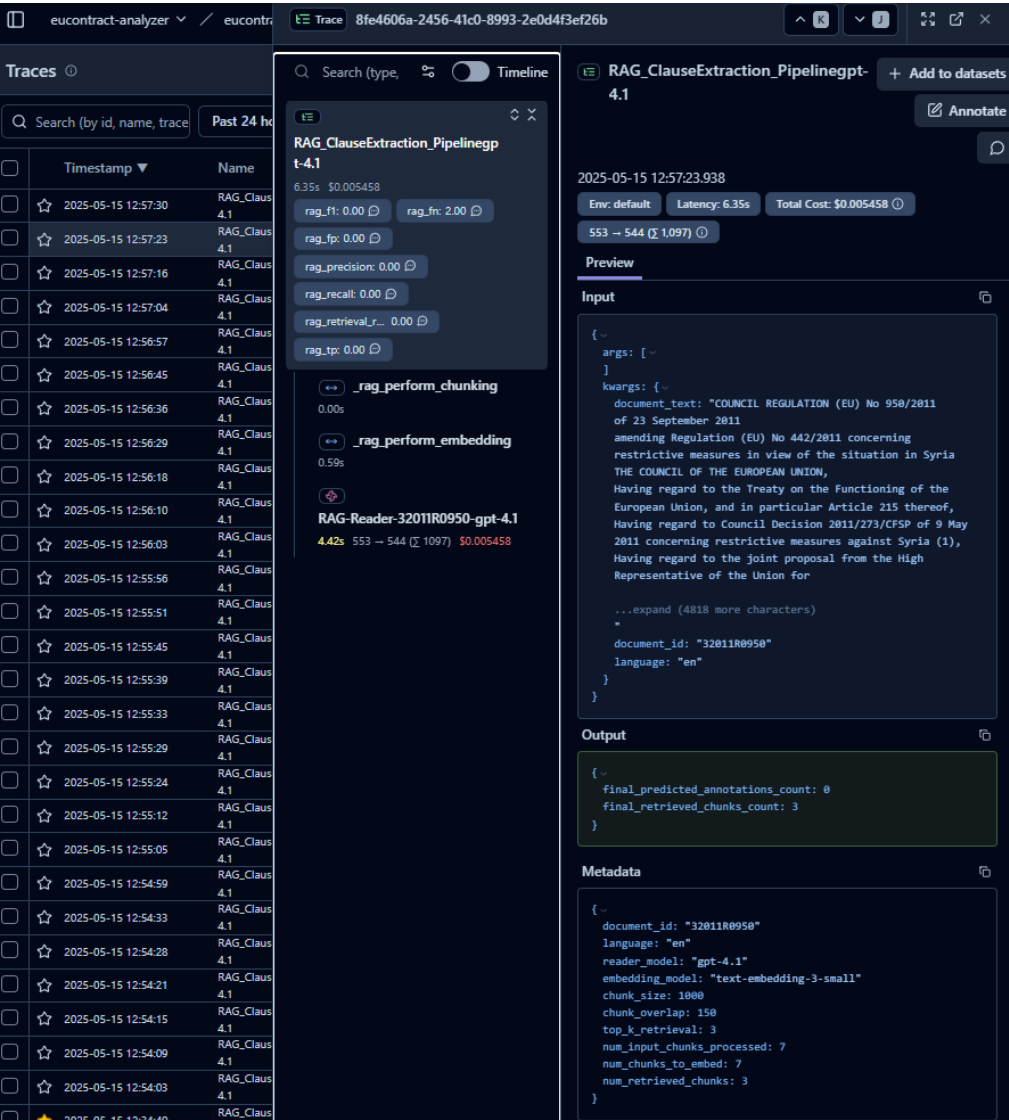


Fig. A.12: RAG Pipeline GPT-4.1 Trace Analysis - Detailed execution trace showing retrieval-generation workflow

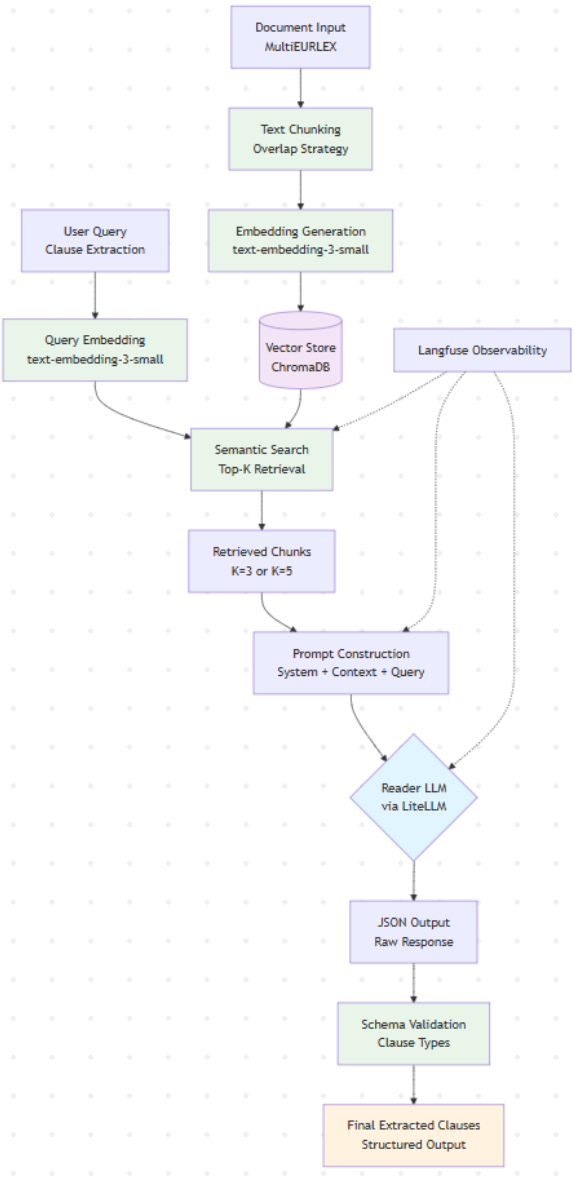


Fig. A.13: RAG Pipeline Architecture with Langfuse Observability Integration

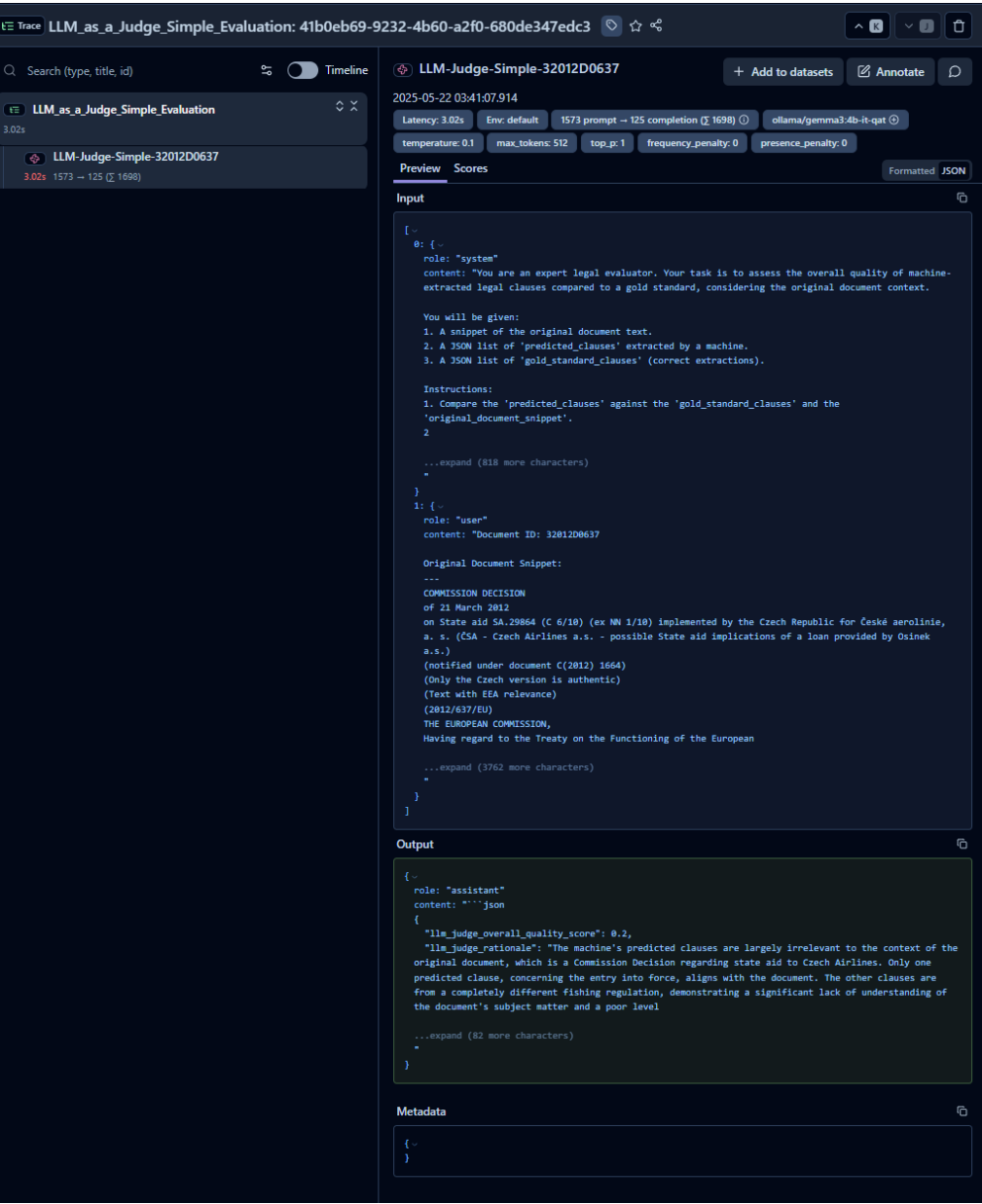


Fig. A.14: Judge evaluation Rationale qualitative review

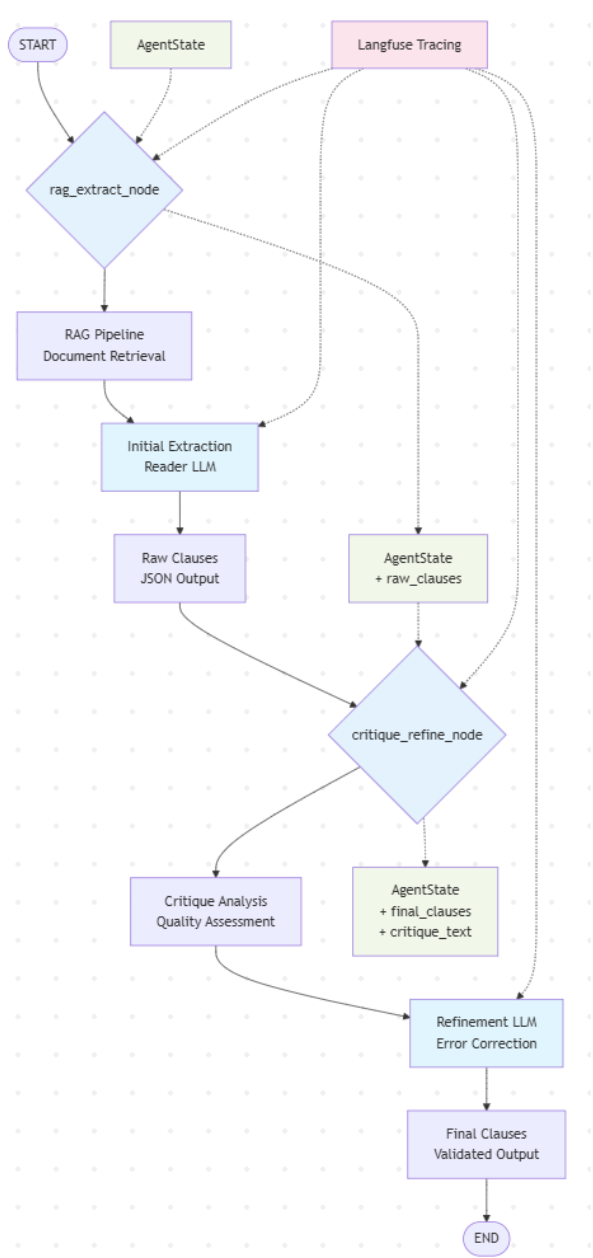


Fig. A.15: Two-Step Agentic System Architecture using LangGraph



```

chunks = _rag_perform_chunking(
    document_text=document_text,
    document_id=document_id,
    session_id=session_id,
    user_id=user_id,
)
    return {
        "predicted_annotations": [],
        "rag_trace_id": rag_trace_id,
        "retrieved_chunk_texts": [],
        "error": error_message_for_output,
    }
) # 2. Embedding Chunks
chunk_embeddings = _rag_perform_embedding(
    chunks=chunks,
    document_id=document_id,
    embedding_model_alias_param=current_embedding_model,
    session_id=session_id,
    user_id=user_id,
)
return {
    "predicted_annotations": [],
    "rag_trace_id": rag_trace_id,
    "retrieved_chunk_texts": chunks,
    "error": error_message_for_output,
}

# 3. Vector Store (Load/Index) - Simple in-memory for this example
collection = get_rag_collection(RAG_COLLECTION_NAME)
chunk_ids = [f"{document_id}_chunk_{i}" for i in range(len(chunks))]
if chunk_ids:
    collection.add(embeddings=chunk_embeddings,
        documents=chunks, ids=chunk_ids)
else:
    logger.warning(f"No chunks to add to vector
        store for document {document_id}.")

# 4. Retrieval
retrieved_docs_texts = []
query_text = f"Extract legal clauses such as {'', '
.join(CLAUSE_TYPES)} from a European Union regulation document."

```

```

try:
    query_embedding = get_embeddings(
        [query_text], model_alias=current_embedding_model
    )[0]
except Exception as e:
    logger.error(f"Failed to embed query for {document_id}: {e}",
        exc_info=True)
    langfuse_context.update_current_trace(
        output={
            "status": "Query embedding failed",
            "error": error_message_for_output,
        }
    )
return {
    "predicted_annotations": [],
    "rag_trace_id": rag_trace_id,
    "retrieved_chunk_texts": [],
    "error": error_message_for_output,
}

# Final update to the main trace output
langfuse_context.update_current_trace(
    output={
        "final_predicted_annotations_count": len(predicted_annotations),
        "final_retrieved_chunks_count": len(retrieved_docs_texts),
    }
)
return {
    "predicted_annotations": predicted_annotations,
    "rag_trace_id": rag_trace_id,
    "retrieved_chunk_texts": retrieved_docs_texts,
}

```

## 5 Evaluation Metrics Calculation

```
# poc2_metrics.py
# 1. set-Based Type-level - Exact Match on Clause Type P/R/F1
def calculate_set_based_type_metrics(
    predicted_clauses: Sequence[Dict[str, Any]],
    gold_clauses: Sequence[Dict[str, Any]]
) -> Dict[str, float]:
    """
    Calculates Precision, Recall, F1 based on the SETS of unique clause types.
    Ignores counts of each type, only cares about presence/absence of the type.
    """

    predicted_types_set: Set[str] = {
        clause["clause_type"] for clause in predicted_clauses if
        "clause_type" in clause
    }
    gold_types_set: Set[str] = {
        clause["clause_type"] for clause in gold_clauses if
        "clause_type" in clause
    }
    tp = float(len(predicted_types_set.intersection(gold_types_set)))
    fp = float(len(predicted_types_set.difference(gold_types_set)))
    fn = float(len(gold_types_set.difference(predicted_types_set)))
    precision = _safe_div(tp, tp + fp)
    recall = _safe_div(tp, tp + fn)
    f1 = _f1(precision, recall)
    return {
        "type_set_precision": precision,
        "type_set_recall": recall,
        "type_set_f1": f1,
        "type_set_tp": tp, # Count of matching types
        "type_set_fp": fp, # Count of predicted types not in gold
        "type_set_fn": fn, # Count of gold types not predicted
    }

# core helpers
def _f1(p: float, r: float) -> float:
    return 2 * p * r / (p + r) if (p + r) > 0 else 0.0
# f1 = (
# 2 * (precision * recall) / (precision + recall)
# if (precision + recall) > 0
```

```
# else 0.0

def _safe_div(num: int, denom: int) -> float: # allow float for generality
    return num / denom if denom > 0 else 0.0
    # precision = tp / (tp + fp) if (tp + fp) > 0 else 0.0
    # recall = tp / (tp + fn) if (tp + fn) > 0 else 0.0
```

## 6 LLM-as-a-Judge Evaluation

```
# poc2_llm_judge.py
# --- Judge Prompts (Simplified System Prompt) ---
JUDGE_SYSTEM_PROMPT = """You are an expert legal evaluator.
Your task is to assess the overall quality of machine-extracted
legal clauses compared to a gold standard, considering
the original document context.
```

You will be given:

1. A snippet of the original document text.
2. A JSON list of 'predicted\_clauses' extracted by a machine.
3. A JSON list of 'gold\_standard\_clauses' (correct extractions).

Instructions:

1. Compare the 'predicted\_clauses' against the 'gold\_standard\_clauses' and the 'original\_document\_snippet'.
2. Consider:
  - Coverage: Did the machine find most of the important clauses from the gold standard?
  - Precision: Did the machine avoid extracting irrelevant or badly formed clauses?
  - Accuracy: For clauses that seem to match, is the 'clause\_type' correct and is the 'clause\_text' accurate and complete?
3. Based on your holistic assessment, provide an 'overall\_quality\_score' as a single float between 0.0 (very poor, many errors, missed key items) and 1.0 (excellent, accurately captures all gold items with no significant errors).
4. Provide a brief 'rationale' (1-3 sentences) explaining your score, highlighting the most significant strengths or weaknesses.

Output your evaluation as a single, valid JSON object with the following exact keys:

- "llm\_judge\_overall\_quality\_score": float
- "llm\_judge\_rationale": string"""

```
JUDGE_USER_PROMPT_TEMPLATE = """Document ID: {document_id}

Original Document Snippet:
---
{original_document_snippet}
---

Predicted Clauses (Machine Extracted):
---
{predicted_clauses_json}
---

Gold Standard Clauses (Correct):
---
{gold_standard_clauses_json}
---

Based on your evaluation,
provide your assessment as a single JSON object
with 'llm_judge_overall_quality_score' and 'llm_judge_rationale' keys: """
```

## 6.1 Extended Evaluation Metrics

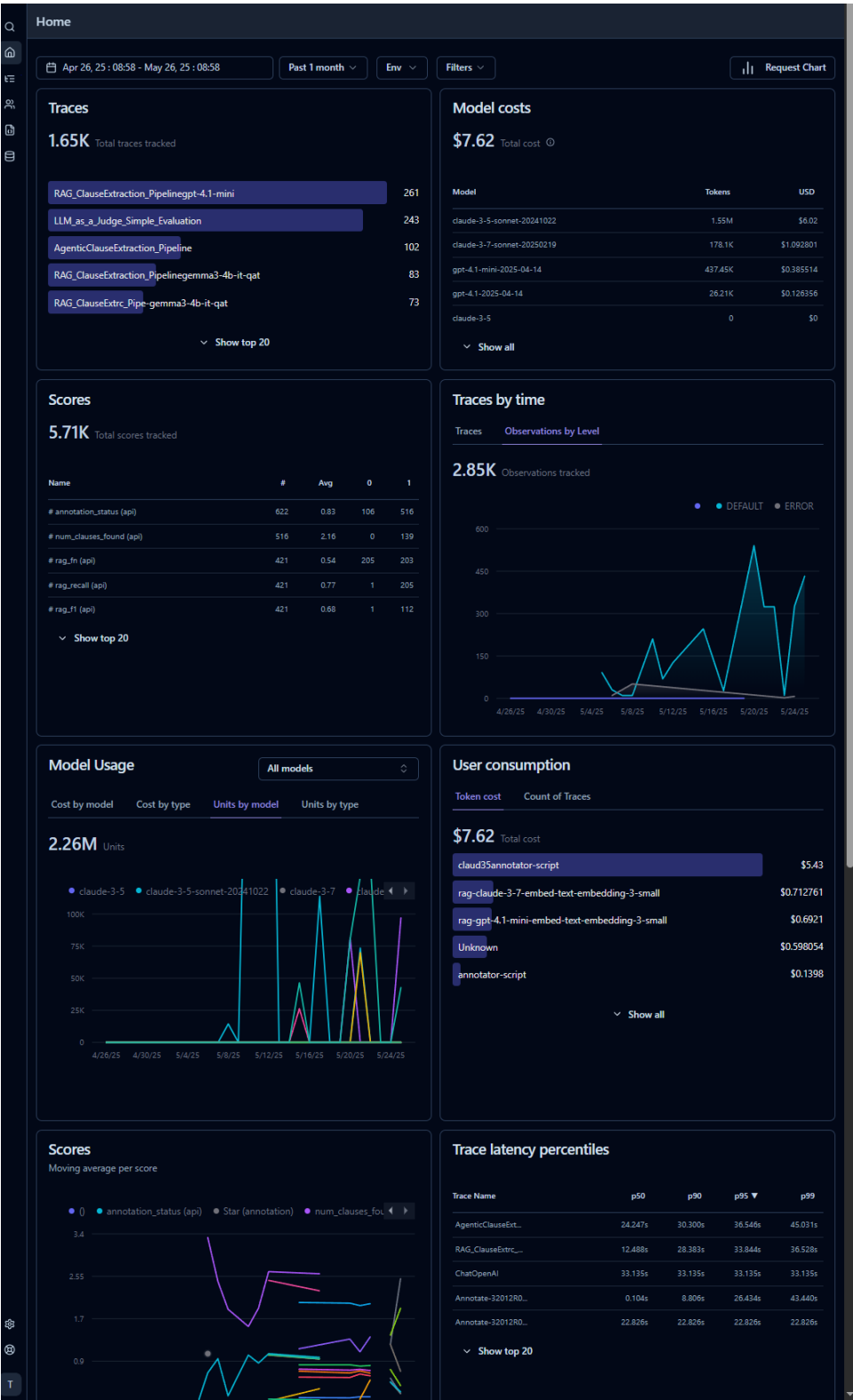


Fig. A.16: Total Final API Model Costs, Traces by time Model Usages