ClauseAI: Multi-Agent Contract Intelligence System

AI-Powered Risk Analysis using LangGraph, Pinecone, and RAG

Pari Bhattacharya
B.Tech Artificial Intelligence & Data Science

2026

Contents

1 Abstract

ClauseAI is an AI-powered contract intelligence platform designed to analyze legal agreements using a multi-agent architecture. The system integrates Large Language Models (LLMs), LangGraph-based orchestration, Pinecone vector storage, and Retrieval-Augmented Generation (RAG) to generate structured risk insights, clause-level summaries, and interactive visual dashboards.

The platform aims to simplify contract interpretation for non-technical users, startups, and enterprises by transforming unstructured legal text into structured, actionable intelligence.

2 Problem Statement

Legal contracts are complex, lengthy, and difficult to interpret without domain expertise. Manual review is time-consuming and error-prone. Existing solutions either lack contextual understanding or provide generic summaries without structured risk analysis.

The objective of this project is to build a scalable, AI-driven system that:

- Performs domain-specific contract analysis
- Extracts structured clause-level risks
- Stores semantic contract memory
- Enables context-aware question answering
- Provides dashboard-level visualization

3 System Overview

ClauseAI follows a modular pipeline architecture.

High-Level Workflow

```
User → Streamlit UI
        ↓
Text Extraction
        ↓
LangGraph Orchestration
        ↓
Parallel AI Agents (Groq LLM)
        ↓
Structured JSON Output
        ↓
Pinecone Vector Storage
        ↓
RAG Query Engine
        ↓
Visualization + Reports
```

4 Architecture Design

4.1 1. Frontend (Streamlit)

The user interface is built using Streamlit. It supports:

- Contract upload (PDF, DOCX, TXT)
- Multi-agent risk display
- Customizable report generation
- RAG-based question answering
- Interactive visualization dashboard

Streamlit session state is used for managing multi-page data consistency.

4.2 2. Multi-Agent Architecture

The system implements four specialized AI agents:

- Legal Agent
- Compliance Agent
- Finance Agent
- Operations Agent

Each agent:

- Receives context-specific contract text
- Generates structured JSON output
- Produces domain summary and clause-level insights

Agents are executed in parallel using `ThreadPoolExecutor` for improved performance.

4.3 3. LangGraph Orchestration

LangGraph coordinates the multi-agent pipeline. It ensures:

- Controlled execution order
- Structured flow between components
- Modular and extensible architecture

4.4 4. Pinecone Vector Database

Each extracted clause is converted into an embedding and stored in Pinecone.

Stored Structure Example:

```
{
  contract_id: "contract_123",
  agent: "Legal",
  embedding: [...],
  metadata: {
    clause_type: "Termination",
    risk_level: "High",
    summary: "...",
    recommendation: "..."
  }
}
```

This enables semantic search and contextual retrieval.

4.5 5. Retrieval-Augmented Generation (RAG)

When a user asks a question:

1. Query is converted into embedding.
2. Pinecone retrieves semantically similar clauses.
3. Retrieved clauses are passed to LLM.
4. LLM generates grounded response.

This reduces hallucination and ensures contract-specific answers.

5 Visualization Dashboard

A dedicated dashboard page provides:

- Risk Distribution Pie Chart
- Agent-wise Risk Comparison Bar Chart
- Compliance Score
- Contract Health Score
- Important Clause Table
- Priority Action Panel

Plotly and Pandas are used for data visualization.

6 Custom Report Generation

Users can customize downloadable reports based on:

- Tone (Formal, Executive, Technical, Concise)
- Focus (Balanced, Risk-focused, Finance-focused)
- Length (Short, Medium, Detailed)
- Structure (Bullet-based, Executive Summary, Detailed Sections)

This allows flexibility for different stakeholders.

7 Implementation Challenges

Several practical challenges were encountered during development:

7.1 1. Inconsistent JSON Outputs

LLM responses occasionally returned invalid or incomplete JSON. Solution:

- Strict JSON prompt formatting
- Retry mechanism
- Fallback summary generator

7.2 2. Uniform Risk Classification

Initial prompts resulted in most clauses being classified as "Medium". Solution:

- Prompt refinement to enforce realistic risk assignment
- Domain-specific context segmentation

7.3 3. Multi-Page State Management

Streamlit multi-page navigation initially caused duplicate visualization outputs. Solution:

- Improved session state control
- Single dedicated visualization page

7.4 4. RAG Retrieval Precision

Early retrieval results were too broad. Solution:

- Improved embedding consistency
- Structured metadata filtering

8 Limitations

- Depends on external LLM APIs.
- No formal accuracy benchmarking conducted yet.
- OCR not implemented for scanned documents.

9 Future Enhancements

- Multi-contract comparison
- Version tracking
- OCR integration
- User authentication
- SaaS deployment

10 Technology Stack

- Frontend: Streamlit
- LLM: Groq (LLaMA 3.1)
- Orchestration: LangGraph
- Vector Database: Pinecone
- Visualization: Plotly, Pandas
- Backend: Python
- Version Control: Git + GitHub

11 Conclusion

ClauseAI demonstrates the practical integration of LLMs, vector databases, orchestration frameworks, and interactive dashboards into a cohesive contract intelligence platform. The project reflects applied understanding of AI systems engineering, prompt optimization, retrieval pipelines, and scalable multi-agent architectures.

The system bridges the gap between unstructured legal documents and actionable business intelligence.

Developer: Pari Bhattacharya
AI & Data Science Engineer
Email: paribhattacharya05@gmail.com