# Report: Mid Evaluation - Embedding and Chunking Strategy Analysis

**Project:** Subject Matter Expert Agent (Indian Constitutional Rights)
**Prepared by:** LMA_TEAM

25 October 2025

## Contents

## 1 Introduction

This report details the design, implementation, and evaluation of a Subject Matter Expert (SME) AI Agent focused on the legal domain, specifically concerning the Indian Constitution and related rights. The primary objective is to create a highly extensible Retrieval-Augmented Generation (RAG) system capable of assisting users with legal research, understanding legal concepts, and basic document preparation. The project emphasizes agentic capabilities, robust RAG pipeline construction, workflow orchestration, and tool integration.

The agent aims to provide accurate, contextually relevant information by leveraging a curated knowledge base derived from various legal documents and responding to user queries through a conversational interface, potentially invoking tools for tasks like document generation or summarization.

## 2 SME Definition and scope

Chosen Domain: Legal (Indian Constitution Rights)

AI SME Role Definition:

The AI agent acts as a knowledgeable assistant for understanding aspects of the Indian Constitution, fundamental rights, related legal statutes, and illustrative case law.

### 2.1 Capabilities of the SME

- Create case study summaries.
- Generate court style documents.
- Summarize statutes.
- AI Chat bot for learners to understand legal terms - This helps the litigants (for the person who sues and the person who gets sued) to understand the legal terms.
- Prepare arguments and counter-arguments for any given case - This internally requires a tool call for all the above 3 operations.

## 3 High Level Design of the system

## 4 Data Preparation

### 4.1 Document Collection and Organization

Source documents were collected from various online repositories and organized in a root-level directory structure (./data). The system is designed to handle heterogeneous formats including PDF, PPTX, TXT, MD, CSV, JSONL, and Parquet.

**Dataset Information:**

The following table details the datasets collected for the project's knowledge base and potential few-shot examples:

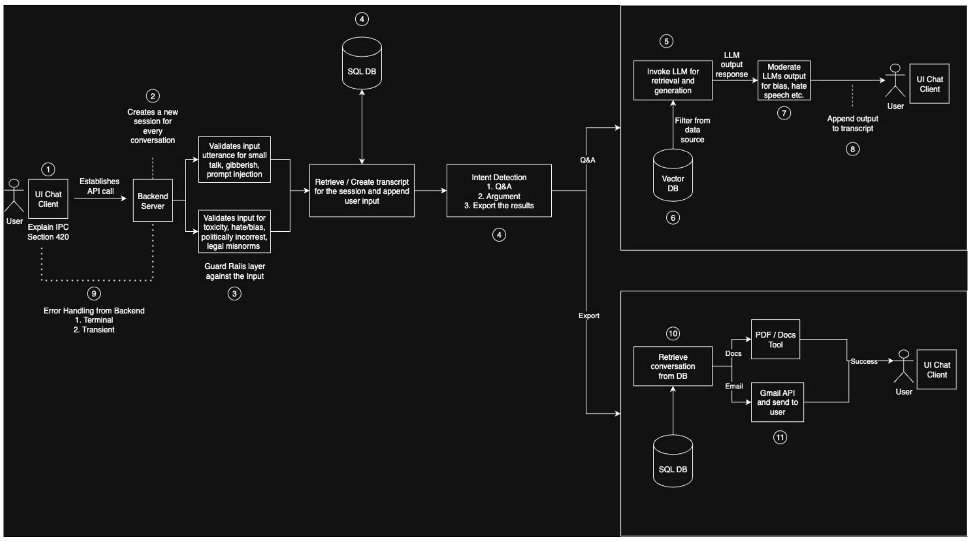
User
UI Chat Client
Explain IPC Section 420
1
Establishes API call
Backend Server
2
Creates a new session for every conversation
Validates input utterance for small talk, gibberish, prompt injection
Validates input for toxicity, hate/bias, politically incorrect, legal misnorms
Guard Rails layer against the Input
3
9
Error Handling from Backend
1. Terminal
2. Transient
4
SQL DB
Retrieve / Create transcript for the session and append user input
Intent Detection
1. Q&A
2. Argument
3. Export the results
4
Q&A
Export
5
Invoke LLM for retrieval and generation
LLM output response
Moderate LLMs output for bias, hate speech etc.
7
Filter from data source
Vector DB
6
Append output to transcript
8
User
UI Chat Client
10
Retrieve conversation from DB
SQL DB
Docs
Email
PDF / Docs Tool
Gmail API and send to user
11
Success
User
UI Chat Client

Figure 1: High level architectural diagram of the RAG system

Table 1: Collected Datasets for Legal SME Agent

| **Source** | **Link** | **Name of Dataset** | **Type of Information Contained** | **Mime Type(s)** |
|---|---|---|---|---|
| Hugging Face | Link | Indian Constitution Articles | Text of Articles of the Indian Constitution | CSV |
| Hugging Face | Link | Indian Law Dataset | Various Indian legal acts/statutes sections | CSV |
| Hugging Face | Link | Indian Constitution QA (Gemini) | Question/Answer pairs about Indian Constitution | Parquet |
| Hugging Face | Link | Lawyer GPT India | Legal dialogues, Q&A, potentially case snippets | JSONL |
| Hugging Face | Link | ILC (Indian Legal Corpus) | Legal text, potentially case law or statutes | JSONL |
| SlideShare | Link | Human Rights Presentation | Overview of Human Rights concepts | PPTX |
| SlideShare | Link | Our Indian Constitution Presentation | Overview of the Indian Constitution | PPTX |
| SlideShare | Link | Right To Education Presentation | Details on RTE Act and provisions | PPTX |
| SlideShare | Link | National Policy Of Education (1986) | Details on Education Policy | PPTX |
| SlideShare | Link | Confronting Marginalisation | Concepts related to social marginalisation | PPTX |

Table 1: Collected Datasets for Legal SME Agent (Continued)

| **Source** | **Link** | **Name of Dataset** | **Type of Information Contained** | **Mime Type(s)** |
|---|---|---|---|---|
| SlideShare | Link | Criminal Justice System in India Presentation | Overview of India's Criminal Justice System | PPTX |
| SlideShare | Link | Salient Features of Education Policy | Summary points on Education Policy | PPTX |
| SMKV Bastar | Link | Legal Rights of Women | PDF document detailing women's legal rights | PDF |
| MOSPI | Link | Constitutional & Legal Rights (Women) | Provisions for women in India | PDF |
| UN Women | Link | UN Women India Report | Report on gender equality status in India | PDF |
| CUTN | Link | Guide to Gender-Based Laws | Detailed guide on gender laws in India | PDF |
| NHRC | Link | NHRC Human Rights Manual | Manual covering various human rights aspects | PDF |
| NHRC | Link | Completed Research Projects | Various detailed reports (incl. Case Studies) | PDF (Multiple Files) |
| NCERT | [e.g., keps201.pdf] | Class 11 Political Science Books | Textbook chapters on Constitution, Rights, etc. | PDF |
| Mendeley Data | Link | Indic Legal QA Dataset | Case-specific Question/Answer pairs | JSON |
| DevDataLab | Link | Indian District Court Cases | Dataset of district court case information | CSV/Other |
| GitHub | [Local Zip Folder] | Court Case Judgments/Summaries | Text files of judgments and summaries | TXT) |
| Hugging Face | Link | Indian Constitution Instruct (Subset) | Instruction-following dataset (test subset) | JSONL(?) |

The collected datasets, encompassing constitutional articles, statutes, case law summaries, QA pairs, and explanatory materials, provide a robust foundation for the agent's core capabilities. Case-specific data directly supports summarizing judgments, while foundational texts enable accurate statute summarization. The combination of QA datasets, legal texts in context, and simpler explanations from sources like NCERT ensures the chatbot can effectively define legal terms for litigants by retrieving and synthesizing relevant definitions and examples.

Furthermore, the breadth of the dataset, including legal principles from statutes and their application in case law, provides the necessary building blocks for assisting with basic court document drafting and generating arguments/counter-arguments. We also have court case judgements and summaries, with retrieved clauses and case structures offer guidance. Complex tasks like argument preparation leverage the integrated knowledge from statutes, precedents, and QA retrieved via RAG, enabling the agent to function as a knowledgeable SME within its defined scope.

## 5 Chunking and Embedding

The script establishes a robust baseline for a Retrieval-Augmented Generation (RAG) system by implementing multiple sophisticated chunking strategies and utilizing a high-performance open-source embedding model.

The primary embedding model used for vector generation and storage is **sentence-transformers/all-mpnet-base-v2**. This model is coupled with a `Pinecone` serverless vector database configured for 768 dimensions and cosine similarity. The pipeline is notable for its experimentation with three distinct chunking methods (Character, Recursive Character, and Semantic) and its enrichment of each vector with AI-generated metadata (summaries and labels) from Google's `gemini-2.5-pro` model.

While the current stack is powerful and well-chosen for development, this report also outlines key areas for domain-specific improvement and discusses the potential future migration from managed Pinecone to a self-hosted **FAISS** library for cost and performance optimization.

### 5.1 Chunking Strategy Analysis

The script implements three distinct chunking strategies, allowing for flexible experimentation.

1. `chunk_file_text_by_character_splitting` **(Mode:** `character`**)**
   - **Method:** Uses LangChain's `CharacterTextSplitter`. This is the most basic method, splitting text by a list of separators and then forcing chunks to a fixed character size (`chunk_size=500`).
   - **Assessment:** This method is fast but semantically "dumb." It risks splitting sentences or even words, which can destroy the meaning of a chunk and lead to poor retrieval.
2. `chunk_file_text_by_recursive_character_splitting` **(Mode:** `recursive_character`**)**
   - **Method:** Uses LangChain's `RecursiveCharacterTextSplitter`. This is the primary method called in the `__main__` block. It is a more intelligent fixed-size splitter that attempts to split on a hierarchy of separators (e.g., paragraphs, then sentences).
   - **Assessment:** This is the *de facto* standard for modern RAG pipelines. It provides a good balance of speed and semantic coherence, making it a strong default choice.
3. `chunk_file_text_by_semantic_splitting` **(Mode:** `semantic`**)**
   - **Method:** This is an advanced, experimental technique. It splits the text into individual sentences, embeds them (using `gemini-embedding-001`), and calculates the cosine distance between adjacent sentences. It then creates chunks by splitting the text wherever this distance "jumps" above a threshold (the 95th percentile).
   - **Assessment:** This is the most promising strategy for your domain. Legal text is dense and topically structured. Semantic chunking ensures that chunks are divided by *topic* rather than by *character count*, which is ideal for providing accurate context to the SME agent.

## 5.2 Embedding Model Analysis

The core embedding model used to generate the vectors stored in Pinecone is **sentence-transformers/all-mpnet-base-v2**.

This is determined by three key pieces of evidence in the code:

1. **Model Loading:** The `if __name__ == "__main__":` block explicitly loads this model:

   ```
   model = load_embedding_model("sentence-transformers/all-mpnet-base-v2")
   ```

2. **Index Configuration:** The `init_pinecone()` function creates an index with `dimension = 768`, which is the exact output dimension of the `all-mpnet-base-v2` model.

   ```
   dimension = embedding_configs["all-mpnet-base-v2"]["dimensions"], # 768
   ```

3. **Upsert Function:** The `embed_and_upsert_chunks()` function, which is responsible for the final ingestion, receives this `embed_model` and uses it to generate the embeddings:

   ```
   embeddings = embed_model.encode(chunks, batch_size=32, show_progress_bar=True)
   ```

## 5.3 Suitability for Indian Constitutional Rights

- Semantic chunking leverages embeddings to understand the meaning within text, grouping sentences or passages that are conceptually related. Unlike fixed-size methods which can arbitrarily split related ideas, semantic chunking aims to identify natural "break points" where topics shift. This approach ensures that individual chunks maintain better contextual coherence, holding semantically similar information together.

  The core hypothesis is that keeping related ideas within the same chunk improves retrieval relevance for RAG systems. By analyzing embedding distances between sequential text windows (like groups of sentences), this method identifies where the meaning changes significantly, creating boundaries that respect the flow of information and potentially leading to more accurate and contextually complete snippets being retrieved for downstream tasks.

  But the actual performance of these techniques will be known only after the evaluation of downstream tasks of the SME agent.

- For embedding, `all-mpnet-base-v2` is chosen as it is a high-performing, general-purpose model. It has a strong grasp of the English language and complex sentence structures, making it a reliable starting point for encoding legal and constitutional texts. However, as a general-purpose model, it has not been specifically trained on legal or Indian civics corpora. It may struggle to differentiate the nuanced meanings of highly specific legal terms, Latin phrases used in law, or concepts unique to the Indian Constitution.

## 6 Recommendations and Future Improvements

### 6.1 Immediate Recommendations

1. **Evaluate Chunking Strategies:** The `chunk_size=500` for the recursive splitter is very small. For dense legal text, experiments with larger chunk sizes should be tried (e.g., 1000-2000 characters).

2. **Fine-Tune the Embedding Model:** This will provide the single greatest boost in quality. Fine-tuning `all-mpnet-base-v2` on a custom dataset of Indian constitutional law will teach the model the specific vocabulary of your domain.

## 6.2 Future Replacement: Pinecone vs. FAISS

The script currently uses **Pinecone**, a managed, serverless vector database. **FAISS (Facebook AI Similarity Search)** is an open-source *library* for similarity search, not a database.

**Comparison:**

| **Feature** | **Pinecone (Current)** | **FAISS (Future Replacement)** |
|---|---|---|
| **Type** | Fully managed database (SaaS) | Open-source library |
| **Setup** | Easy (API key) | Complex (Requires self-hosting) |
| **Features** | Built-in metadata filtering, serverless scaling, API | Core vector search. Metadata storage must be built separately. |
| **Cost** | Pay-as-you-go (can be expensive at scale) | Free (pay for your own compute/storage) |
| **Performance** | Excellent (managed) | Extremely fast (state-of-the-art) |
| **Ops Overhead** | Very low | **Very high.** You manage scaling, indexing, and API creation. |

Table 2: Comparison of Pinecone vs. FAISS

**Migration Path and Recommendation:**

1. **Short-Term:** Continue using **Pinecone**. The development speed and zero operational overhead are more valuable at this stage.
2. **Long-Term (At Scale):** Migrating to FAISS is a valid strategy for production-grade systems.