CourtNav: Voice-Guided, Anchor-Accurate Navigation of Long Legal Documents in Courtrooms

Sai Khadloya
sai@adalat.ai
Adalat AI, India

Kush Juvekar
kush@adalat.ai
Adalat AI, India

Arghya Bhattacharya
arghya@adalat.ai
Adalat AI, India

Utkarsh Saxena
utkarsh@adalat.ai
Adalat AI, India

Abstract

Judicial work depends on close reading of long records, charge sheets, pleadings, annexures, orders, often spanning hundreds of pages. With limited staff support, exhaustive reading during hearings is impractical. We present CourtNav, a voice-guided, anchor-first navigator for legal PDFs that maps a judge's spoken command (e.g., "go to paragraph 23", "highlight the contradiction in the cross-examination") directly to a highlighted paragraph in seconds. CourtNav transcribes the command, classifies intent with a grammar-first(Exact regex matching), LLM-backed router classifying the queries using few shot examples, retrieves over a layout-aware hybrid index, and auto-scrolls the viewer to the cited span while highlighting it and close alternates. By design, the interface shows only grounded passages, never free text, keeping evidence verifiable and auditable. This need is acute in India, where judgments and cross-examinations are notoriously long.In a pilot on representative charge sheets, pleadings, and orders, median time-to-relevance drops from 3–5 minutes (manual navigation) to 10–15 seconds; with quick visual verification included, 30–45 seconds. Under fixed time budgets, this navigation-first design increases the breadth of the record actually consulted while preserving control and transparency.

1 Introduction

High-volume courts routinely face long filings and crowded dockets (often dozens of matters per day) which leads to massive case delays (Agarwala and Behera, 2024). Despite near-universal digitization (e-Courts) and access to case data at scale, the core interaction problem remains: *how can a judge interrogate a voluminous record quickly and faithfully?*

Summaries aid orientation but can hide citations and miss pivotal passages, even retrieval-augmented systems sometimes surface mis-grounded references (Various, 2025; Stolfo, 2024). Adjudication prioritizes verifiability: decision-makers must jump to the exact locus in the record and see it highlighted. We therefore target navigation, not paraphrase.

We present a voice-guided, *anchor-first* navigator for long legal PDFs that converts a spoken command (e.g., "go to paragraph 23") into a highlighted paragraph within seconds. The system couples layout-aware indexing and anchor generation over scanned/structured PDFs, a constrained command grammar with LLM back-off for coverage, hybrid retrieval with de-duplication, and a viewer that auto-scrolls while preserving on-screen evidence. **Our primary contributions are:**

- A court-facing system that prioritizes direct-to-paragraph, auditable navigation over free-form summarization.
- A dataset and evaluation protocol for long-record navigation measuring time-to-relevance, strict-hit accuracy at anchor level, and end-to-end latency.
- A pilot study on charge sheets, pleadings, and orders showing large reductions in time-to-relevance under fixed time budgets.

2 Related Work

Long-document QA and retrieval in law. Legal QA and retrieval have evolved from sentence-level factoid questions to long-form answers grounded in statutes and case law. Benchmark tasks span holding extraction (e.g., CaseHOLD (Zheng et al., 2021)), case-retrieval datasets such as LeCaRD/LeCaRDv2 (Ma et al., 2021, 2024), and broader evaluation suites like LegalBench (Guha et al., 2023). More recent resources target long-form QA (e.g., LLeQA, Legal-LFQA) (Louis et al., 2024; leg, 2024). While these emphasize retrieval quality and reasoning, they operate at the document level, returning entire cases rather than pinpointed spans, and are not designed for judge-facing inter-

Proceedings of the Natural Legal Language Processing Workshop 2025, pages 351–358

action loops.

Summarization for legal documents. Faithfulness remains a central challenge. Surveys and long-context datasets (e.g., CaseSumm) catalog hallucination modes and metric gaps (Basile et al., 2025; Heddaya et al., 2024). General summarization work similarly shows unsupported content in abstractive outputs (Maynez et al., 2020; Fabbri et al., 2022). Summaries aid orientation but do not replace the need to *jump to the exact place in the record*.

Evidence-first interfaces. Outside law, explainable QA resources require systems to surface supporting sentences (e.g., HotpotQA (Yang et al., 2018)) and page-level localization for document images (DocVQA) (Mathew et al., 2021), improving interpretability. However, most legal QA/summarization systems return text without a UI that *enforces* verification.

Prior legal QA/summarization and DocVQA work does not focus on *navigation* as we do: a voice-guided, anchor-first interface that maps spoken commands to highlighted paragraphs. Our system combines long-document indexing, hybrid retrieval, a domain-adapted query router, and a judge-facing viewer that *enforces* verification. To the best of our knowledge, we are the first ones to attempt building such a system for the legal domain.

3 System Overview

3.1 Ingest and Layout-Aware Indexing

Long records mix scanned pages, numbered paragraphs that reset per section, multi-column text, and tables that span pages. Pure text extraction loses the geometry needed for trustworthy highlights; vision-only pipelines are compute-heavy and brittle on low-quality scans. We therefore perform *layout-aware parsing* that emits canonical spans with stable coordinates and IDs. *Anchor (definition).* We treat every minimal displayable unit as an *anchor* $\langle$ page, bbox, span_id, char_range, type $\in$ {para, heading, table_cell} $\rangle$. Headings, paragraphs, and cross-page tables are extracted (e.g., with Docling) and normalized (hyphenation, numbering). We then build two complementary indices: a *lexical* BM25 index for exact legal cues (sections, names, citations) and a *windowed late-interaction* index for paraphrastic queries, produced over sliding windows to preserve local context (Jha et al., 2024). For tables, we preserve grid structure (table_id, row, col, rowspan/colspan) so cell-level anchors exist even when a table breaks across pages; we also store a light markdown/HTML rendering for downstream snippet previews (Auer et al., 2024; Robertson, 2009; Khattab and Zaharia, 2020; tab, 2022; Huang et al., 2022).

3.2 Query Interpretation and Routing

Spoken requests cluster into three practical families: *temporal* ("go to paragraph 23"), *contextual* ("locate the contradiction in PW-2's cross-examination about the call detail records"), and *summarization* ("summarize the charges"). Latency and predictability are critical in court, so we use a *grammar-first, LLM-backed* router. ASR text is first parsed by a compact command grammar that yields typed intents and slots (page/paragraph, statute, party, exhibit, or table region), if parsing fails or is ambiguous, a lightweight LLM back-off produces a structured action with confidence and a few disambiguating rewrites surfaced to the user. Summarization requests hit a precomputed extractive+abstractive synopsis, but responses still link back to anchors so users can inspect sources rather than accept paraphrase.

3.3 Retrieval and Anchor Alignment

A near hit is not enough; the system must land *on* the paragraph (or cell). We perform hybrid retrieval across the lexical and late-interaction indices, interleave and deduplicate candidates by anchor overlap, then optionally re-rank a short list. Using the ingest-time anchor map, we deterministically map retrieved text offsets back to their anchors resolving OCR drift with tolerant matching and then command the viewer to smooth-scroll to the top anchor and *highlight* all corroborating anchors. Table queries resolve to cell anchors via (table_id, row, col) even across page breaks. If evidence is insufficient (low confidence or conflicting candidates), the UI offers a compact disambiguation list (keyboard/voice selectable) or withholds an answer. In all cases, every line of response is grounded in visible anchors rather than free text.

3.4 Voice Pipeline

Courtrooms are noisy, and users often code-switch. We run an on-premise streaming ASR pipeline (Whisper-based acoustic model with VAD gating and domain lexicon biasing for statutes, party names, and common legal terms)(Radford et al.,

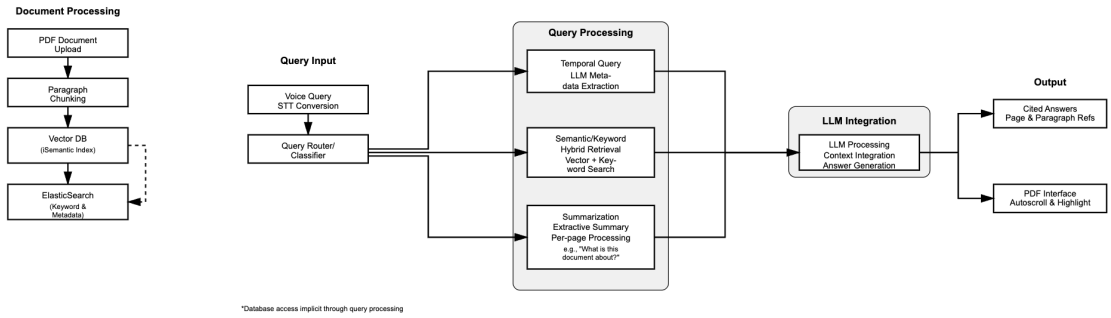

Figure 1: End-to-end flow. An uploaded PDF is parsed into layout-anchored spans and indexed (lexical + dense). Voice commands are transcribed on-prem and mapped to navigation actions. Retrieval produces candidate anchors, whose relevance to the queries is checked by the llm, while the viewer scrolls and highlights all the anchor which have substance related to the query

2022; OpenAI, 2022) to generate partial transcripts quickly enough for responsive UI feedback . The Whisper model is fine-tuned on legal jargon and maintains an acceptable WER even in noisy ambient conditions through post-processing heuristics. The transcript, along with a "confirm/cancel" loop, gives the user opportunity to correct mishears and errors before any jump occurs. All audio is processed ephemerally, and nothing leaves the court network.

3.5 Viewer and Interaction Design

The UI is optimized for *hands free, eyes busy* hearings. We extend a standard viewer (PDF.js) and design around three principles. **Speakable affordances:** every action a judge can perform via keyboard is also addressable through a short utterance with customizable shortcuts (next hit," previous section," toggle highlights"). **Anchored evidence:** the system never answers in free text without pointing to passages. All relevant anchors are highlighted with sentence-level backtracking to the anchor. **Low-drama navigation:** we prefer *smooth scroll to anchor* rather than page jumps to preserve spatial memory. A breadcrumb trail records recent anchors and can be invoked to backtrack quickly. A compact *evidence panel* lists retrieved snippets with page or paragraph badges, and clicking a badge or saying open two" scrolls to that anchor. Keyboard shortcuts are supported for all operations so counsel can use the interface even if the microphone is muted. The layout avoids occluding the document, while transcripts and disambiguation chips collapse automatically after action, ensuring the judge's visual context remains stable (pdf, 2025).

3.6 Privacy and Deployability

All components—ASR, router, retrieval, and viewer—run as independent services within the court's infrastructure. No audio is stored, and logs capture only structured commands and anchor IDs for auditing. This design keeps the UI responsive under load while allowing each service to scale independently. The loose coupling also enables multiple judges to work concurrently without changing the user contract.

4 Evaluation

4.1 Experimental Setup

Corpus and task construction. To approximate day-of-hearing use, we curated long records that judges and counsel routinely handle: charge sheets (with annexures and lists), pleadings, orders, and reasoned judgments. Selection was stratified to cover (i) *born-digital* and *scanned* PDFs; (ii) table-heavy sections (accused/witness lists, seizure memos) and narrative sections; (iii) varied pagination/numbering schemes (paragraphs that reset, annexures, multi-column text). The final set has **15 documents** of **50–350 pages** each (avg. **100**). To elicit realistic queries, practising lawyers first skimmed each document as they would before a hearing and then authored speakable prompts in three families that reflect in-court needs: *temporal* (explicit positions), *contextual* (content descriptions), and *summarization* (brief "what's in the petition/charges" gists). Each query is paired with one or more *gold anchor* paragraphs or table cells that must be annotated at anchor level and verified by a second lawyer, with disagreements adjudicated. The retrieval set comprises **600 contextual** and **50 summarization** queries. Temporal queries are

generated directly from document numbering and appear across all documents.

Participants and protocol. For navigation trials, we recruit lawyers who did *not* annotate the corresponding document. Each participant executes all queries for a document using two conditions: (i) a stock PDF reader (manual scroll and *Find*), and (ii) *CourtNav*. Conditions are counter-balanced across participants to mitigate order effects. Timing starts at query issuance (spoken or typed) and ends when the user lands on the gold anchor (temporal/contextual) or finishes a two-sentence synopsis with at least two paragraph-level citations.

Baselines and measures. The primary baseline is manual/search-based navigation with a stock PDF reader. Within our system we ablate retrieval modes: keyword-only, dense-only, hybrid, and our late-window+keyword variant. We report *time-to-relevance (TTR)* in seconds and *strict-hit F1* at paragraph (or table-cell) granularity, computed as mean $\pm$ sd across participants and documents. For summarization, the baseline corresponds to the protocol above (producing a two-sentence gist with $\geq$2 citations using only the PDF reader), providing a practical comparator rather than full-document reading time.

4.2 Results

Table 1 presents time-to-relevance (TTR). The reader reduces TTR by half on Temporal commands ($t = 13.3$, $p < 10^{-7}$) and shortens Contextual queries from minutes to seconds ($t = 58.6$, $p < 10^{-12}$). For Summarization, we report only system time because manual reading scales with document length. The near-constant response time across query types stems from architectural choices: precomputed synopsis for summaries, direct anchor lookup for temporal spans, and sublinear vector search and fast elastic-search for retrieval (Malkov and et al., 2018).

Retrieval choices significantly influence *strict-hit F1* (Figure 2). Keyword search performs well on statute or party mentions, dense-only aids paraphrase but misses exact citations, and a simple hybrid offers further improvement. However, our late-window+keyword variant achieves the best *strict-hit F1* within the same latency budget.

Query type	**Baseline** (seconds)	**Ours** (seconds)
Temporal	10 ± 2.0	5 ± 0.5
Contextual	200 ± 15.0	6 ± 1.0
Summarization	—	6 ± 1.2

Table 1: Time-to-relevance (mean $\pm$ sd). Baseline is manual navigation with a stock PDF reader. "—" indicates no comparable baseline because manual reading depends on document length, and with our document length it scales to days

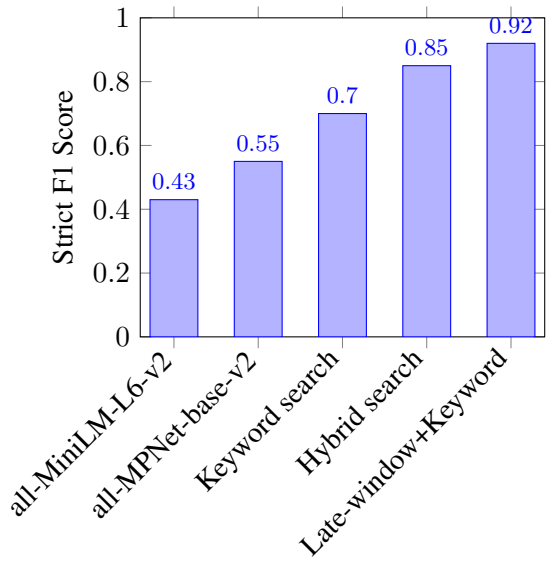

Figure 2: Strict-hit F1 for different retrieval settings.

5 Conclusion

We presented a voice-driven anchor-first reader that couples layout-aware indexing, hybrid retrieval, and a LLM-backed router to make long legal PDFs navigable in real time. In a pilot on charge sheets, pleadings, and orders, it cut time-to-relevance from several minutes to seconds (halved for *temporal* jumps, orders-of-magnitude for *contextual*) while preserving paragraph-level strict-hit accuracy and keeping every jump auditable. For Next steps, we will extend multilingual commands/ASR, and run field trials. We also release a long-form Indian legal retrieval dataset[1] which we plan to keep expanding, enabling Indian legal research.

Limitations

Our system currently supports documents up to 350 pages seamlessly, but as size increases, the responsiveness of the PDF.js reader declines. In future work, we plan to build a custom PDF viewer designed to operate smoothly with much larger doc-

[1] https://huggingface.co/datasets/adalat-ai/Indian-Legal-Retrieval-Generation

uments. While the LLM-based query router shows strong accuracy in blind trials, absolute guarantees are impossible due to the stochastic nature of queries. RAG helps reduce hallucinations (Johnston, 2025; Banerjee et al., 2024) , but does not fully eliminate them (sta), even though we use a model adapted to strong instructions with explicit prompts to avoid ambiguous queries and to abstain when retrieved content is insufficient for a truthful answer. ASR errors are infrequent but non-negligible, and output varies with dialect or accent (especially given the wide range of accents in India). The system assumes English input, support for vernacular Indian languages remains future work on both the ASR side and document navigation side. A judge-in-the-loop feedback system is also missing, which will be essential for pilot testing and for developing stronger query classification models.

Ethical Considerations.

Deploying AI in judicial settings raises ethical concerns. Generative models can reproduce biases present in training data, and their overconfidence may mislead users (sta). We mitigate this by grounding answers in the document and by surfacing retrieved passages for verification.If no relevant retrievals exist, no answer is given, ensuring all responses remain strictly within the document. The system does not make substantive recommendations, it only navigates to requested text. User data is never sent to foreign APIs, is stored on Indian servers, and is deleted immediately upon user request. No data is used to train any models. We follow proper licensing, and all external software is open source under the Apache 2.0 License (The Apache Software Foundation, 2004). Our retrieval evaluation was fully transparent, but no benchmark covers every scenario due to the stochastic nature of information retrieval. We plan to improve incrementally by expanding the size of the dataset.

References

2022. Table transformer (tatr). https://github.com/microsoft/table-transformer. Microsoft.

2024. Towards legal long-form question answering with grounded contexts. In *CIKM*.

2025. Pdf.js: A web standards-based pdf renderer. https://mozilla.github.io/pdf.js/. Mozilla.

Sugam Agarwala and Smruti Ranjan Behera. 2024. Mammoth backlog of court cases pending in india: A spatial visualisation. *Regional Studies, Regional Science*, 11(1):757–760.

Christoph Auer, Maksym Lysak, Ahmed Nassar, and et al. 2024. Docling technical report. *arXiv:2408.09869*.

Sourav Banerjee, Ayushi Agarwal, and Saloni Singla. 2024. Llms will always hallucinate, and we need to live with this. *arXiv preprint arXiv:2409.05746*.

Valerio Basile and 1 others. 2025. A comprehensive survey on legal summarization. *Preprint*, arXiv:2501.17830.

Alexander R. Fabbri, Chien-Sheng Wu, Wenhao Liu, and Caiming Xiong. 2022. QAFactEval: Improved qa-based factual consistency evaluation for summarization. In *NAACL*.

Neel Guha, Julian Nyarko, Daniel E. Ho, Christopher Ré, and 1 others. 2023. Legalbench: A collaboratively built benchmark for measuring legal reasoning in large language models.

Mourad Heddaya and 1 others. 2024. Casesumm: A large-scale dataset for long-context summarization from U.S. supreme court opinions. *Preprint*, arXiv:2501.00097.

Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. 2022. Layoutlmv3: Pre-training for document ai with unified text and image masking. *arXiv:2204.08387*.

Rohan Jha, Bo Wang, Michael Günther, Georgios Mastrapas, Saba Sturua, Isabelle Mohr, Andreas Koukounas, Mohammad Kalim Akram, Nan Wang, and Han Xiao. 2024. Jina-colbert-v2: A general-purpose multilingual late interaction retriever. In *Proceedings of the Fourth Workshop on Multilingual Representation Learning (MRL 2024)*, pages 159–166, Miami, Florida, USA. Association for Computational Linguistics.

Peter Johnston. 2025. Retrieval-augmented generation (rag): towards a promising llm architecture for legal work? Accessed 2 August 2025.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of SIGIR*.

Annie Louis and 1 others. 2024. Interpretable long-form legal question answering with expert-annotated evidence. In *AAAI*.

Yue Ma and 1 others. 2021. Lecard: A legal case retrieval dataset for chinese law system. In *SIGIR*.

Yue Ma and 1 others. 2024. LeCaRDv2: A large-scale chinese legal case retrieval dataset. In *SIGIR*.

Yu. A. Malkov and et al. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(6):1341–1354.

Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. 2021. Docvqa: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages XXX–YYY. IEEE/CVF.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *ACL*.

OpenAI. 2022. Introducing whisper. https://openai.com/index/whisper/. Accessed: 2025-08-30.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2022. Robust speech recognition via large-scale weak supervision. *arXiv:2212.04356*.

Stephen Robertson. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389.

Alessandro Stolfo. 2024. Groundedness in retrieval-augmented long-form generation: An empirical study. *arXiv preprint*.

The Apache Software Foundation. 2004. Apache License, Version 2.0. Updated and maintained by the Apache Software Foundation.

Various. 2025. A comprehensive survey on automatic text summarization. *arXiv preprint*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Lucy Lu Wang Zheng and 1 others. 2021. When does pretraining help? assessing self-supervised learning for law and the casehold dataset. In *NeurIPS Datasets and Benchmarks*.

A System User Interface

The system interface demonstrates the core functionality described in Section 3, providing judges with direct document access through both voice and traditional input methods. The interface maintains the principle of anchored evidence display while supporting hands-free operation during hearings.

B Indexing Architecture Details

B.1 Elasticsearch Integration

Our lexical indexing layer utilizes Elasticsearch 8.x as the primary engine for BM25-based keyword matching. The choice of Elasticsearch provides several advantages for legal document retrieval:

- **Legal-specific tokenization**: Custom analyzers handle legal citation formats, statute references, and party name patterns
- **Field-specific boosting**: Paragraph headers, section titles, and table captions receive higher relevance weights
- **Real-time indexing**: Supports incremental document addition during active court sessions

Index configuration includes custom mappings for legal document structure:

```
{
  "mappings": {
    "properties": {
      "content": {"type": "text"},
     "paragraph_id": {"type": "keyword"},
      "page_number": {"type": "integer"},
     "section_type": {"type": "keyword"},
       "bbox_coords": {"type": "object"}
    }
  }
}
```

B.2 Milvus Vector Database

The dense retrieval component leverages Milvus 2.x for high-performance vector similarity search. Milvus provides:

- **Scalable vector storage**: Handles embedding collections for documents up to 350 pages efficiently
- **GPU acceleration**: Supports CUDA-enabled similarity search for sub-second response times
- **Index optimization**: Uses IVF_FLAT indexing with 1024 clusters for optimal recall-latency trade-off
- **Hybrid search support**: Enables metadata filtering combined with vector similarity

Vector collection schema:

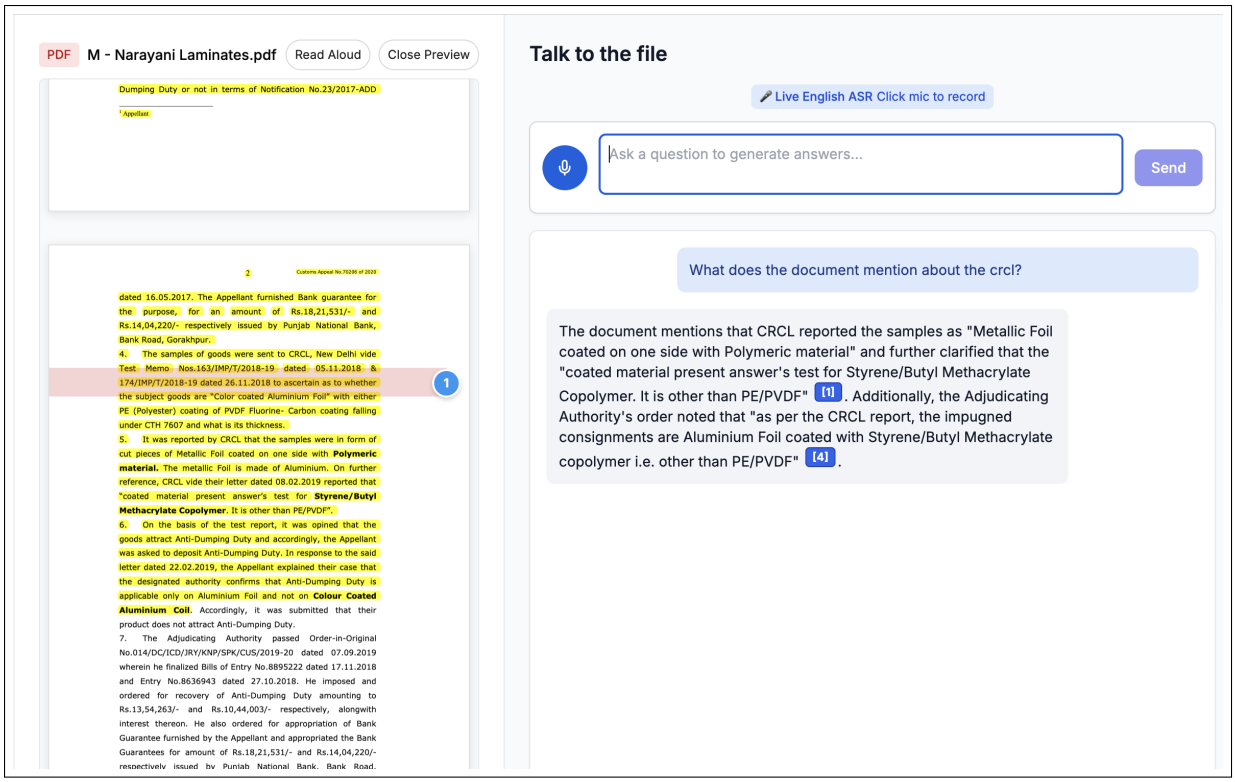

Figure 3: User interface of the system showing the PDF viewer with document navigation capabilities and voice command interface.

```
collection_schema = {
    "chunk_id": DataType.VARCHAR,
    "embedding": DataType.FLOAT_VECTOR,
   "paragraph_anchor": DataType.VARCHAR,
    "document_id": DataType.VARCHAR,
    "page_range": DataType.VARCHAR
}
```

C Late-Interaction Sliding Window Mechanism

C.1 Architecture Overview

The late-interaction sliding window approach addresses two critical challenges in legal document retrieval: maintaining sufficient context for semantic understanding while preserving fine-grained anchor precision.

Traditional dense retrieval methods encode fixed-size chunks independently, potentially fragmenting legal arguments that span multiple paragraphs. Our windowed late-interaction mechanism operates as follows:

1. **Sliding window construction**: Generate overlapping windows of paragraphs.

2. **Individual token encoding**: Each token in the window receives its own embedding vector

3. **Query-time interaction**: Compute similarity between query tokens and document tokens independently

4. **Maxpool aggregation**: Select maximum similarity scores across token pairs for final relevance scoring

C.2 Mathematical Formulation

Given a query $Q = \{q_1, q_2, ..., q_m\}$ and document window $D = \{d_1, d_2, ..., d_n\}$, the late-interaction score is computed as:

$$\text{Score}(Q, D) = \sum_{i=1}^{m} \max_{j=1}^{n} \text{sim}(q_i, d_j)$$

Where $\text{sim}(\cdot, \cdot)$ represents cosine similarity between token embeddings. This formulation allows fine-grained matching while maintaining computational efficiency through maximum operations.

D Hybrid Search Implementation

The hybrid search combines Elasticsearch and Milvus results using a weighted scoring approach:

$$\text{Final\_Score} = \alpha \cdot \text{Keyword} + (1 - \alpha) \cdot \text{Vector}$$

Where $\alpha = 0.7$ provides optimal balance for legal queries, emphasizing keyword matching while incorporating semantic similarity. Score normalization ensures comparable ranges across both retrieval methods.

E LLM Usage and Parameters for Reproducibility

All language understanding, summarization, and translation tasks within the pipeline were performed using the **Qwen3-Coder-30B-A3B-Instruct-FP8** model[2], deployed via the `vLLM` inference engine for high-throughput serving.

The model operates in FP8 precision, enabling significantly reduced memory footprint and faster inference with negligible degradation in output quality. To ensure reproducibility, all experiments used the default `vLLM` sampling parameters unless otherwise stated.

- **Model:** Qwen3-Coder-30B-A3B-Instruct-FP8
- **Serving Framework:** vLLM (GPU inference optimized)
- **Precision:** FP8 quantized weights
- **Max context length:** 8192 tokens
- **Default Sampling Parameters:**
  - `temperature` = 0.7
  - `top_p` = 0.9
  - `top_k` = 50
  - `repetition_penalty` = 1.0
  - `max_tokens` = 2000
- **Deployment:** Self-hosted GPU inference cluster
- **Integration:** Invoked via FastAPI microservice supporting both synchronous and streaming responses.

The combination of vLLM's optimized memory paging and Qwen's efficient A3B architecture provides low-latency, high-throughput inference suitable for real-time document understanding and generation workloads.

[2] https://huggingface.co/Qwen/Qwen3-Coder-30B-A3B-Instruct-FP8

F Performance Optimization

The document processing pipeline achieves real-time performance through:

- **Parallel processing**: Simultaneous embedding generation and Elasticsearch indexing
- **Connection pooling**: Persistent connections to both Elasticsearch and Milvus clusters
- **Loose coupling**: ASR, Index stores and self hosted llms are loosely coupled and can scale independetly enabling a highly scalable and efficient architecture.