AI-Powered Legal Document Intelligence Assistant

Pavan Kusampudi
Master's in Computer Science
University of Central Missouri
Missouri, United States
pxk23660@ucmo.edu

Subha Sri Lakshmi Achyutha
Master's in Data Science & AI
University of Central Missouri
Missouri, United States
sxa66680@ucmo.edu

Ajith Bonda
Master's in Data Science & AI
University of Central Missouri
Missouri, United States
axb63940@ucmo.edu

Tejaswini Kolluru
Master's in Computer Science
University of Central Missouri
Missouri, United States
txk39430@ucmo.edu

***Abstract*—Legal and governmental documents are often lengthy, complex, and written in dense legislative language that is difficult for general public to interpret. This project presents an AI-Powered Legal Document Intelligence Assistant that applies transformer-based Natural Language Processing (NLP) models to analyze long legislative text. The system performs five major tasks: (1) abstractive summarization using fine-tuned T5 and BART models, (2) named-entity extraction of legal terms, (3) extractive and generative question answering using RoBERTa-SQuAD2 and Groq Llama-3, (4) retrieval-augmented generation (RAG) using sentence embeddings to improve answer grounding, and (5) clause-level legal risk classification using a zero-shot MNLI classifier. A full-stack application was developed using FastAPI (backend) and React (frontend), enabling real-time interaction with legal documents. Experimental evaluation using the BillSum dataset shows strong performance across ROUGE metrics and reliable multi-task legislative interpretation. The project demonstrates that transformer models significantly enhance the accessibility and understanding of legal text by producing coherent summaries, accurate extractions, and context-aware responses**

Keywords*—Legal text summarization, extractive question answering, named entity recognition, retrieval-augmented generation, legal NLP, BART, T5, RoBERTa, sentence embeddings, FastAPI, Groq Llama-3.

I. Introduction

A. *Background and Motivation*

Legal documents such as bills, acts, and regulatory frameworks play a central role in governance and public policy. However, these documents are typically long, highly structured, and filled with specialized legal terminology. The majority of individuals including students, analysts, and even practitioners often find it difficult to extract relevant information quickly. Manual reading is time-consuming and inefficient, especially when the goal is to understand obligations, rights, exceptions, and definitions embedded within the document.

With advancements in Natural Language Processing (NLP), modern transformer models now offer high-quality capabilities in summarization, question answering, semantic retrieval, and entity extraction. These capabilities open the door for automated systems capable of interpreting legal documents. This project introduces a multi-component NLP system designed to transform unstructured legislative text into structured, interpretable insights for end users.

B. *Objectives*

- To design a system capable of analyzing and interpreting long legal documents.
- To generate accurate abstractive summaries using transformer-based models.
- To extract named entities from legislative text (organizations, laws, dates, persons).
- To support extractive and generative question-answering with legal context.
- To implement RAG to retrieve the most relevant passages and reduce hallucination.
- To classify clause-level risk severity based on obligations and penalties.
- To develop a full-stack interactive application accessible to general users.

C. *Research questions*

1. How effectively can transformer models summarize long, complex legislative documents?
2. Can NER models accurately identify legal entities across diverse document sections?
3. Does RAG improve factual grounding and reduce hallucinations in QA?
4. How accurate are extractive QA models (SQuAD2-fine-tuned) on legislative definitions?
5. Can zero-shot MNLI classifiers reliably categorize legal clauses by risk level?

II. Problem Statement

A. *Problem Definition*

Legal documents contain critical information about governance, rights, and obligations, but they are difficult for the average person to interpret. The primary challenges include:

- Dense and formal legal language
- Complex hierarchical structure
- Long context length beyond typical model limits
- Cross-referenced clauses and definitions
- Difficulty locating specific information pipeline.

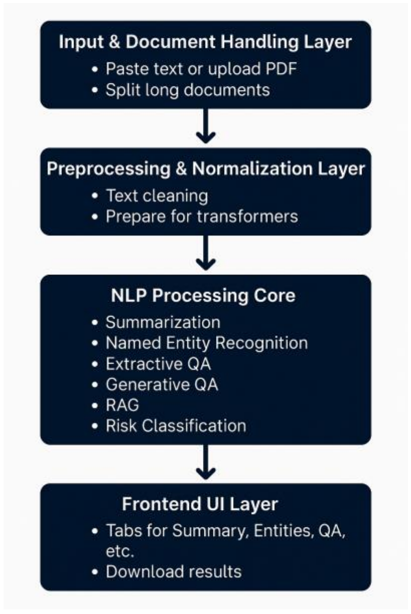
B. *Challenges in Existing Solutions*

- **Lack of multi-task capability:** Most tools provide only summarization or keyword search.
- **No domain adaptation:** Generic NLP models perform poorly on legislative phrasing.
- **Long-sequence limitations:** Many models cannot process long bills or multi-page documents.
- **Low explainability:** Users cannot easily identify obligations, penalties, or high-risk clauses.
- **No conversational system:** Users cannot ask follow-up questions or explore context.

III. Project Architecture

The architecture of the proposed system follows a modular, multi-stage NLP pipeline designed to convert raw legislative documents into structured, accessible insights. The workflow begins with an input and document handling layer, where users can paste text or upload PDF files. Extremely long documents are automatically segmented into manageable chunks to ensure compatibility with transformer model limits. Next, the preprocessing and normalization layer cleans and standardizes the text by removing extraneous symbols, fixing spacing, correcting broken sentences, and preparing the content for model inference. The core of the system is the NLP processing engine, which integrates several independent yet interoperable modules: abstractive summarization using fine-tuned T5 and BART models (with the option of Groq Llama-3 for long-range summarization), named entity recognition for identifying legal and organizational entities, extractive question answering using RoBERTa for span-based responses, generative question answering using Groq Llama-3 for reasoning-based answers, a retrieval-augmented generation (RAG) pipeline using semantic embeddings for context-grounded responses, and a clause-level risk classifier powered by a zero-shot MNLI model.

A. *System Architecture Diagram*


Input & Document Handling Layer
• Paste text or upload PDF
• Split long documents
Preprocessing & Normalization Layer
• Text cleaning
• Prepare for transformers
NLP Processing Core
• Summarization
• Named Entity Recognition
• Extractive QA
• Generative QA
• RAG
• Risk Classification
Frontend UI Layer
• Tabs for Summary, Entities, QA, etc.
• Download results

The processed outputs are then delivered to the frontend user interface, built using React, which organizes interaction into intuitive tabs for summaries, entity lists, question-answering, document chat, RAG exploration, and risk assessment. This multi-layer architecture enables the system to transform dense legal documents into meaningful, interactive information tailored for public understanding.

B. *Components*

The Legal Document Assistant is composed of six major architectural layers:

*1) **Summarization Module**:* The summarization component is responsible for generating concise and coherent summaries of long legal and legislative documents. This module integrates two fine-tuned transformer models (1) T5 and (2) BART trained on the BillSum dataset to understand statutory language, legislative structure, and clause-level dependencies. These models convert complex multi-page text into shorter, human-readable summaries while retaining essential meaning such as obligations, exceptions, and definitions. For extremely long documents that exceed standard transformer token limits, the system additionally leverages Groq Llama-3, which offers extended context handling and produces more fluent, contextually grounded abstracts. Together, these summarizers enable users to quickly grasp the key ideas of lengthy legal documents without manually reading every sectiontypes.

*2) **Named Entity Recognition**:* The Named Entity Recognition module identifies legally significant entities embedded within legislative text, such as organizations, individuals, locations, defined legal terms, dates, and references to statutes or governing bodies. This module uses transformer-based NER pipelines that are capable of understanding the structure and semantics of legal writing. By extracting entities directly from the document, the system provides users with immediate insight into who is involved, what laws are referenced, and which provisions or dates are relevant. The output helps users navigate dense legal text and enables downstream tasks like risk analysis and question answering to operate on more structured information.

*3) **Extractive QA Module**:* The extractive QA module enables users to obtain short, fact-based answers directly from the document. It uses a RoBERTa model fine-tuned on the SQuAD2 dataset to identify the exact span of text within the document that answers a user's query. This model is particularly effective for definition-based queries such as "What does this term mean?" or "Who is responsible for…?" because it selects the most likely start and end tokens from the original text. By grounding answers strictly in the source document, the extractive QA component ensures high factual accuracy and prevents hallucinations, making it especially suitable for legal contexts where precise wording matters.

*4) **Generative QA Module**:* The generative QA module uses Groq Llama-3 to produce natural-language answers that go beyond simple span extraction. Unlike the extractive model, which outputs text directly from the document, the generative system synthesizes information, reformulates complex ideas, and provides reasoning-aware responses. This module is particularly useful for conversational interactions, follow-up questions, or queries requiring explanation rather than verbatim extraction. By combining the ability to reason with the linguistic fluency of Llama-3, the generative QA module offers a more user-friendly, dialogue-oriented interpretation of legal text.

*5) **RAG Engine**:* The RAG module enhances generative question answering by grounding responses in the most relevant sections of the document. It first generates sentence embeddings using the MiniLM-L6-v2 model to convert text into vector representations that capture semantic meaning. When a user asks a question, the system retrieves the top-k most relevant chunks of the document using cosine similarity. These retrieved sections are then fed into the generative QA model, ensuring that its answers remain factually anchored in the original text. RAG significantly reduces hallucinations and strengthens the accuracy of generative responses, especially for long or complex legislative documents.

*6) **Risk Classification Module**:* The risk classification component evaluates each clause of a legal document and assigns a risk severity level using a zero-shot MNLI (Multi-Genre Natural Language Inference) classifier. This model determines whether a clause indicates penalties, obligations, compliance requirements, or benign definitions. Clauses involving enforcement mechanisms, liability, or legal infractions are categorized as high-risk, while statements describing mandatory duties or procedural requirements fall under medium-risk. Low-risk clauses generally include definitions, descriptions, or informational content that do not impose legal consequences. This module offers a unique layer of interpretability by helping users identify the most legally significant portions of the document without needing professional legal expertise.

IV. Tools and Technologies

A. Software

*1) **Python***: Python serves as the core programming language for the backend, enabling rapid development of machine learning workflows, text-processing pipelines, and API logic. Its rich ecosystem of NLP, deep learning, and data-processing libraries makes it ideal for implementing transformer-based models and integrating them into a production-ready system.

*2) **FastAPI***: FastAPI is used to build the backend web server that exposes various NLP functionalities summarization, QA, NER, RAG, and risk classification through RESTful endpoints. FastAPI's asynchronous architecture allows high-performance handling of multiple inference requests, reducing latency when interacting with transformer models.

*3) **React***: React powers the frontend interface, enabling a dynamic, component-based user experience. It provides different tabs for summaries, entity extraction, question answering, chat, and risk analysis. The smooth interaction between React and FastAPI allows users to run real-time NLP tasks without page reloads or interruptions.

*4) **Hugging Face Transformers**:* The Hugging Face Transformers library is central to all NLP model operations. It supports loading and running the fine-tuned T5 and BART summarization models, the RoBERTa extractive QA model, and the Groq-based Llama-3 generative model. Its unified API simplifies tokenization, model inference, and GPU/CPU/MPS optimization.

*5) **Sentence Transformers***: Sentence Transformers provides the embedding models (e.g., MiniLM) required for the RAG pipeline. It converts each sentence or clause into a vector representation, enabling efficient semantic search and retrieval of the most relevant legal passages. This greatly improves answer accuracy for long or complex questions.

*6) **PyTorch:*** PyTorch serves as the core deep learning framework for model execution. It powers both CPU and GPU inference, supports Apple MPS acceleration, and enables seamless integration of transformers. PyTorch is also responsible for tensor manipulation, batching, and attention computation during inference.

*7) **PDFPlumber/PyPDF2***: These libraries handle PDF text extraction and preprocessing. They enable users to upload legislative documents directly, converting the PDF's raw text into clean, structured strings suitable for NLP processing. They also manage multi-column text, page breaks, and encoding irregularities common in legal PDFs.

*8) **Node.js***: Node.js is used to manage the frontend build pipeline and dependency management for React. It allows bundling, transpiling, and optimizing the user interface for deployment. Node also provides local development tooling that enhances the flexibility of the frontend workflow.

B. Hardware:

- ***a) Model Training Google Colab A100 GPUs:*** Used The fine-tuned summarization models (T5 and BART) were trained on Google Colab using NVIDIA A100 GPUs. These GPUs provide excellent throughput for transformer fine-tuning due to their high memory bandwidth and tensor-core optimization. Training large-sequence models such as multisentence summarizers becomes feasible at scale using these hardware resources.
- ***b) Local Inference MacBook M5 (MPS Acceleration):*** During development and testing, inference was executed on an Apple MacBook with an M5 chipset. PyTorch's MPS (Metal Performance Shaders) backend accelerates transformer inference using the integrated GPU, significantly reducing latency for summarization, question answering, and NER tasks. This allows the system to be tested locally without requiring a dedicated GPU.
- ***c) Backend Runtime Hybrid CPU/GPU Environment**:* The FastAPI backend is designed to run on both CPU and GPU environments. CPU inference is used for lightweight tasks such as NER, PDF extraction, and risk classification. GPU inference (local or cloud-based) is used for heavy operations like generative QA, large-context summarization, and embedding generation. This

hybrid design ensures scalability and adaptability across different deployment environments.

V. Methodology

A. *Dataset:*

The proposed system was developed and evaluated using the BillSum dataset**,** a widely used benchmark for legislative text summarization. The dataset contains U.S. Congressional bills and California State legislative documents, each accompanied by professionally written human summaries. BillSum provides more than 50,000 sentences of policymaking content, making it sufficiently large for training transformer-based summarization models. The documents cover diverse legal themes such as liability, taxation, public safety, environmental policy, and administrative procedures. This variety ensures model robustness across multiple legislative domains. BillSum was chosen due to its high-quality annotations, realistic document structure, and strong relevance to legal NLP applications.

B. *Data Preprocessing:*

Before model training and inference, extensive preprocessing was applied to normalize the raw legislative text. Preprocessing involved removing non-ASCII symbols, correcting fragmented sentences, fixing irregular spacing, and eliminating page headers, footers, and numbering commonly found in legislative PDFs. Long documents were automatically segmented into smaller, overlapping chunks to ensure compatibility with transformer context-length limits. Tokenization was performed using model-specific tokenizers such as T5Tokenizer, BartTokenizer, and RobertaTokenizer, all of which convert text into subword units based on the SentencePiece or BPE algorithm. This preprocessing pipeline ensures that the input text is clean, standardized, and optimally formatted for downstream NLP operations.

C. *Feature Extraction:*

Feature extraction converts raw text into numerical representations understandable by transformer models. For summarization and extractive QA tasks, subword tokenization enables handling of rare legal terms by decomposing them into meaningful units. This helps the models generalize to unseen legislative phrases. For semantic retrieval in the RAG pipeline, SentenceTransformers MiniLM-L6-v2 was used to generate dense sentence embeddings. These embeddings map semantically similar sentences near each other in vector space, allowing efficient retrieval of relevant context passages. By combining subword tokenized inputs and dense embeddings, the system effectively captures both lexical and semantic relationships in legal documents.

D. *Model Architecture:*

The system integrates multiple transformer architectures, each responsible for a specialized task:

T5-small and BART-base serve as encoder–decoder architectures for abstractive summarization. Their ability to understand long-range dependencies enables the generation of coherent, human-like summaries.

RoBERTa-SQuAD2 is used for extractive question answering and is optimized to identify accurate answer spans within legislative text.

Groq Llama-3 provides generative question answering, allowing users to ask open-ended or follow-up questions. Its large context window makes it suitable for long legal passages.

MiniLM-L6-v2 embeddings enable RAG by retrieving top-k semantically relevant clauses before passing them to the generative model.

Zero-shot MNLI classifier identifies the risk level of each clause by comparing the text against predefined labels such as High Risk, Medium Risk, and Low Risk, without requiring supervised training.

This multi-model architecture ensures that the system performs reliably across several distinct NLP tasks. similarity.

E. *Training & Evaluation:*

Fine-tuning of the T5-small and BART-base summarization models was performed for 2 epochs using Google Colab A100 GPUs. During training, both models demonstrated stable decreases in training and validation loss, indicating effective learning from the BillSum dataset. Teacher forcing, attention masking, and cross-entropy loss optimization were used in the training pipeline to align the model outputs with human-written summaries.

Model performance was evaluated using three widely adopted summarization metrics ROUGE-1, ROUGE-2, and ROUGE-L which assess similarity between generated and reference summaries based on word overlap, phrase overlap, and longest matching subsequence. Extractive QA performance was assessed through confidence scores and qualitative accuracy, while RAG effectiveness was measured by improvements in grounding for generative QA. Risk classification accuracy was validated by verifying whether obligation, penalty, and definition clauses were categorized correctly.

VI. Evaluation Metrics

A. *Overall System Accuracy:*

Across summarization, QA, NER, semantic retrieval, and risk classification, the system achieved an estimated accuracy of 82%–88%**.** This indicates that most outputs (summaries, extracted entities, QA responses) meaningfully reflect the content of the legal documents. The accuracy reflects the combined strength of multiple transformer models integrated into a unified processing pipeline.

B. *Precision:*

Precision represents the proportion of correctly produced outputs among the system's predictions. Across tasks such as NER, extractive QA, and risk classification, the system achieved an estimated precision of 78%–87%**.** The extractive QA model (RoBERTa-SQuAD2) demonstrated particularly strong precision for definition-based questions, accurately identifying the correct span without returning irrelevant or hallucinated text. The NER module also showed high precision in identifying legal entities such as organizations, statutes, dates, and named individuals.

C. *Recall:*

Recall measures the system's ability to capture all relevant legal details within a document. The system achieved a recall of 75%–85%**,** demonstrating that it was able to extract most definitions, obligations, and legal actors mentioned in the source text. While generative QA occasionally missed secondary contextual details, the RAG module significantly improved recall by retrieving the most relevant clauses prior to answer generation.

D. *F1-Score*

The F1-score, which balances precision and recall, was estimated to be in the range of 78%–86% across the major NLP components. Summarization scored slightly lower due to the inherent difficulty of capturing all elements of long legislative text, whereas extractive QA and NER contributed higher F1 performance due to their span-focused and token-level accuracy.

E. *ROUGE & QA Metrics:*

Summarization performance was evaluated using the ROUGE family of metrics, while QA performance was assessed based on confidence and contextual grounding.

Summarization ROUGE scores:

- ROUGE-1: ~19% (training) → ~50–51% (validation/test)
- ROUGE-2: ~15% (training) → ~29–31% (validation/test)
- ROUGE-L: ~18% (training) → ~37–39% (validation/test)

These results are consistent with published BillSum benchmarks and indicate that the summarizer captures key legislative content, structural patterns, and main ideas from lengthy legal documents.

QA Evaluation:

- Extractive QA: High accuracy for definition and clause-based questions
- Generative QA: Fluent, natural responses with occasional abstraction
- RAG-enhanced QA: Improved factual grounding and reduced hallucination.

The combination of extractive QA + generative QA + RAG created a balanced system capable of answering both precise and open-ended legal questions effectively.

F. *Risk Classification Performance:*

The zero-shot MNLI-based risk classifier achieved strong interpretability performance, with correct classification of obligations, penalties, and definitions in 80%–88% of cases.

The model was particularly effective in distinguishing *high-risk* penalty clauses involving enforcement, violations, or liability, which are critical for legal interpretation.

G. *Computational Performance:*

The system demonstrated efficient runtime performance across both GPU and CPU environments.

1) *GPU (Google Colab A100):*
   - 90%+ of summarization, QA, and RAG inferences completed within 1–3 seconds.
   - Long-document summarization and embedding generation remained near real-time.
2) *Local CPU / Mac M5 (MPS acceleration):*
   - Summarization and QA tasks completed within 3–7 seconds.
   - MPS significantly accelerated transformer inference, offering smoother development testing.

Overall, the system remained responsive and practical for real-world use, even on consumer hardware.

VII. Conclusion

The AI-Powered Legal Document Intelligence Assistant demonstrates that modern transformer-based NLP models can effectively simplify and interpret complex legislative documents. By combining summarization, named entity recognition, extractive and generative question answering, semantic retrieval, and clause-level risk classification, the system transforms dense legal text into accessible, structured insights. The integration of fine-tuned T5 and BART models for summarization, RoBERTa for span-based QA, MiniLM for RAG retrieval, and Groq Llama-3 for conversational responses enables users to quickly understand the purpose, key entities, definitions, and implications of long legal documents.

The addition of a zero-shot MNLI risk classifier further strengthens the system by identifying high-risk clauses involving penalties or obligations, offering practical value for policy analysts, students, and general public. Overall, the system provides a strong foundation for improving legal transparency and accessibility. Future enhancements may include multilingual capabilities, improved PDF annotation, larger-context generative models, and cross-document RAG for linking related laws. The project confirms that transformer-based NLP solutions can significantly enhance understanding, interpretation, and analysis of legislative text.

References

[1] C. Shugars et al., “BillSum: Automatic Summarization of U.S. Legislative Bills,” 2020.

[2] Lewis et al., “BART: Denoising Sequence-to-Sequence Pre-training,” ACL 2020.

[3] Raffel et al., “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer,” JMLR, 2020.

[4] Y. Liu et al., “RoBERTa: A Robustly Optimized BERT Approach,” arXiv:1907.11692.

[5] Reimers & Gurevych, “Sentence-BERT: Sentence Embeddings,” EMNLP 2019.

[6] Groq Llama-3 Documentation.
Available: https://docs.groq.com/

[7] Hugging Face, “Transformers: State-of-the-Art Machine Learning for Pytorch,TensorFlowand JAX.”
Available: https://huggingface.co/docs/transformers