ChatGPT

SensibLaw: Open Legal Knowledge Graph & Reasoning Platform

SensibLaw is an open-source toolkit that integrates natural language processing (NLP), knowledge graphs, and interactive analysis tools to make legal documents and reasoning more accessible. It addresses the challenge of understanding complex legislation and court judgments by extracting their logical structure and presenting it in machine-readable form. Traditional AI approaches – even advanced language models – often miss the mark in the legal domain: they fail to identify relevant context, cannot trace how facts map to laws, and may misrepresent the layered logic of judicial decisions [1]. SensibLaw takes a different approach. It parses legal text into a structured *logic tree* of actors, obligations, conditions, and references, then stores this information in a knowledge graph. This explicit reasoning graph makes implicit legal logic **explicit** and auditable, enabling more accurate retrieval of relevant laws from facts than black-box methods [2]. The platform includes both a command-line interface (CLI) and a web-based console, aiming to serve a broad audience – from lawyers and policy analysts to non-profit advocates and curious citizens – by automating legal text analysis, scenario simulation, and visualization of legal reasoning.

Features and Capabilities

- **Advanced Legal NLP Integration:** SensibLaw employs state-of-the-art NLP for **deterministic text parsing**. It leverages the spaCy library (a proven industry-standard toolkit) for tokenization, part-of-speech tagging, lemmatization, and dependency parsing, replacing brittle regex-based approaches with robust linguistic analysis. Each sentence is segmented and every token is enriched with grammatical attributes and offsets, providing a foundation for higher-level interpretation. SpaCy's accuracy and efficiency make it well-suited for legal text, and it can be augmented with domain-specific models. For example, legal-focused NLP models like **Blackstone** extend spaCy to recognize case names, legal citations, provisions, courts, judges, and more [3] – categories critical for parsing statutes and case law. By using a flexible NLP pipeline, SensibLaw ensures that citations like *"[2002] 2 Cr App R 123"* or references like *"section 1 of the Act"* are correctly identified as structured entities, rather than just strings of text. This **token-aware** approach enables downstream components (like rule extractors and graph builders) to work with clean, annotated data. *(While some enterprise solutions favor heavy-weight frameworks like Spark NLP – reportedly the most popular NLP library in industry surveys* [4] *– SensibLaw opts for spaCy's lightweight, extensible design to better accommodate custom legal rules and integration into a Python-based workflow.)*

- **Rule Extraction and Logic Tree Building:** On top of the NLP engine, SensibLaw implements a rule-based extraction layer to capture the **logical structure** of legal sentences. Modal verbs (e.g. *"must", "may not"*), conditional clauses (e.g. *"if X, then Y"*), and cross-references (e.g. *"under section 5 of the Act"*) are detected using a combination of spaCy's **Matcher/DependencyMatcher** patterns and custom logic. Each token or phrase is tagged with a semantic role – such as *Actor* (subject), *Action* (predicate), *Condition*, *Exception*, *Reference*, or *Penalty*. These tags feed into the **logic tree builder**, which constructs a hierarchical representation of the rule: for example, a prohibition like "A person **must not** sell spray paint **unless** they have a retailer license" would be parsed into a tree with a root (*obligation not to sell paint*), a condition branch (*unless licensed*), and identified actor (*a person*). By making every clause in the text an explicit node in a graph, SensibLaw enables users to see how each requirement or exception is logically connected. This deterministic logic extraction is auditable and reproducible – a sharp contrast to the opaqueness of AI-generated summaries. Importantly, every step is **traceable to the source text** with character offsets, so users can click on a logical element and see exactly where it came from in the document.

- **Legal Knowledge Graph Backend:** All extracted elements – concepts, rules, cases, provisions – are stored in a **knowledge graph** that captures relationships across documents. SensibLaw adopts a property-graph model (nodes and edges with attributes) to represent legal knowledge. Legislation and case law are broken down into nodes like *Provision#ActName:Section*, *Case#Name*, *Concept#LegalPrinciple*, etc., with edges representing relations such as *"cites", "applies", "overrules", "defines"*. This graph-centric design aligns with modern efforts in legal informatics to use graph databases for managing complex legal connections. For instance, cloud-based solutions using AWS Neptune (a graph database) combined with NLP have demonstrated the ability to organize intricate legal relationships and even uncover hidden patterns via graph neural networks [5]. In SensibLaw, the knowledge graph allows cross-document queries – you can ask *"what other cases were applied by Case X?"* or *"show all obligations that refer to Concept Y"*. It also supports **versioning** and evolution of the law: each node can carry metadata like effective dates or amendment history, and the system's built-in versioned storage can retrieve any document *as it was* at a given time. This provides a foundation for temporal reasoning (vital in law where statutes change over time).

- **Interactive Reasoning and Visualization Tools:** SensibLaw comes with an interactive **Streamlit web console** that puts these capabilities into a user-friendly interface. Users can upload a legal document (e.g. a PDF of legislation or a court judgment) and let the system parse it end-to-end. The console then provides multiple views to explore the results:

- *Documents View:* shows the original text with interactive highlights for entities and references. Detected citations or terms are hyperlinked – clicking a section reference scrolls to that section, clicking a case citation can fetch that case's summary if available.
- *Text & Concepts:* allows free-text input to identify legal concepts via a trigger glossary (e.g., typing *"permanent stay"* might return `Concept#StayOfProceedings`). This helps map layman input to formal legal concepts.
- *Rules & Logic:* visualizes the extracted logic tree from a provision or clause. Users can see a tree or outline of conditions and consequences, and even export it as a graph (DOT or JSON) for further analysis. A **reasoning viewer** highlights how conclusions are reached: for example, in a proof tree format often used to explain why a certain outcome follows from given facts and rules.
- *Knowledge Graph Explorer:* enables browsing the subgraph of related legal nodes. If you seed it with a concept (e.g. *TerraNullius*) or a leading case (e.g. *Mabo 1992*), it generates a graph of connected cases and concepts within a few "hops". This is particularly useful to discover precedents and principles: for instance, seeing all cases that cite a particular section of an Act, or visualizing how a landmark case influences others through citations.
- *Case Comparison:* a side-by-side comparison tool for factual scenarios. SensibLaw can take a structured summary of a case's key factors (a "silhouette") and compare it to a user's input story or another case to find overlapping issues or distinguishing facts. The output lists which factors are *common* (overlaps) and which are *missing* in one story versus the other, complete with references to paragraphs where those factors appear. This helps lawyers quickly **distinguish** a new case from precedents or find analogies.

- *Reading-Fatigue Killers:* a suite of utilities to help users digest voluminous legal text quickly. This includes a *“pin-cite navigator”* that extracts all references (e.g. case citations, section numbers) from a document and lets you jump to their contexts – like an auto-generated table of pinpoint references. Another is a *duplicate detector* that identifies and collapses repetitive paragraphs across document revisions or bundles (useful when analyzing multiple drafts or annexes that repeat content). There’s also a *focus mode* that dims extraneous text and highlights key clauses (e.g. obligations and exceptions), helping readers concentrate on crucial parts of a 50-page contract or legislation bundle. These tools are aimed at achieving the holy grail of *“50 pages to first decision in under ten minutes”* – allowing users to navigate and comprehend the core of long documents with speed.

- **Automation & Scenario Simulation:** Beyond analysis, SensibLaw is geared towards **automation of legal workflows**. It can transform natural-language statements into structured data for decision support. For example, negotiators can input a free-form statement of a proposal or concession, and SensibLaw will automatically identify legal concepts and obligations in it, potentially populating a negotiation template or checklist. The platform envisions a *“what-if” simulator* where users adjust parameters (via sliders or inputs) – such as the severity of a contract clause or the win/loss threshold in a dispute – and the system projects outcomes or fairness metrics based on historical data. By cross-referencing against its knowledge graph of past cases and agreements, the tool can flag if a proposed deal falls outside the *“historical compromise range”* or if certain combinations of concessions have led to unfair results in the past. This forward-looking capability is supported by the structured nature of SensibLaw’s data: because legal rules and outcomes are encoded in a machine-readable way, the system can reason about them or feed them into predictive models. While still exploratory, this **neuro-symbolic** approach (combining symbolic legal rules with statistical learning) aligns with emerging research that hybrids knowledge graphs with language models to get the best of both worlds [6]. SensibLaw prioritizes determinism and transparency – for critical functions like scenario evaluation, it uses explicit rule checks and documented metrics, resorting to generative AI only in controlled ways (e.g. to generate a plain-language summary of a set of known facts, not to invent reasoning). This ensures any recommendations or insights are backed by traceable data rather than opaque AI intuition.

- **Provenance and Auditability:** Every output of SensibLaw is accompanied by detailed **provenance** metadata. If the system says “Case A *distinguishes* Case B,” it will provide the source (e.g. which paragraph or citation led to that conclusion). Under the hood, the platform generates cryptographic *receipts* for key processing steps – from text extraction to graph building – enabling a form of *deterministic replay*. This means given the same input and version of the software, the same outputs will be produced, which is crucial for legal reliability. All documents ingested are stored in a versioned repository (e.g., an internal SQLite or graph store with history), so users can retrieve a document as it was on a certain date, and all analyses can be tied to specific data versions. This level of audit trail is often missing in typical AI tools, and it’s a gap SensibLaw intentionally fills to meet the standards expected in legal proceedings and government applications. By design, the system aligns with open standards where possible – for instance, it can ingest Akoma Ntoso (AKN) XML from official legislative sources. (AKN is an open XML standard for legal documents that many governments, including the EU and Italy, have adopted to ensure consistency across legislative data [7].) Embracing such standards and providing full transparency makes SensibLaw attractive for public sector and non-profit use, where trust and interoperability are paramount.

Target Audience and Use Cases

SensibLaw is built to serve a **multi-faceted audience** across the legal and civic spectrum, each benefiting in different ways:

- **Government & Policy Makers:** Legislative drafters and public policy analysts can use SensibLaw to evaluate the *structure and impact of laws*. By mapping entire Acts into a graph of provisions and references, officials can identify redundant or conflicting clauses, trace how a proposed amendment would ripple through related laws, and ensure consistency in language. The platform's ability to analyze linguistic complexity (e.g. sentence length, readability metrics) and highlight cross-references supports efforts to simplify legislation. In fact, governments are exploring similar knowledge-graph approaches for their legislation – for example, Italy's official legislative platform models all national laws as a property graph and uses AI to assess law quality over time [8] [9]. SensibLaw brings such capabilities to a wider user base, enabling any jurisdiction or organization to adopt a data-driven approach to managing legal texts. Moreover, the open-source nature (licensed under MPL 2.0) means public institutions can adopt and customize the tool without prohibitive costs, fostering innovation in legal tech for the public good.

- **Non-Profits & Advocacy Groups:** Civil society organizations and NGOs often need to dissect dense legal documents – be it a new environmental regulation, a human rights treaty, or a draft policy – under tight timelines. SensibLaw's reading fatigue killers and concept mapping are tailored for this scenario. An advocacy group can quickly upload a 100-page bill and get an outline of all the "must" and "must not" obligations it contains, see which existing laws it references, and identify key provisions that warrant further scrutiny. The knowledge graph can reveal hidden connections (e.g., *"This proposed law amends a clause that has been the subject of litigation in these 3 cases…"*), empowering advocates with context that might otherwise take weeks of research. By lowering the barrier to understanding legalese, SensibLaw helps level the playing field for non-profits and community activists who lack large legal teams. It essentially acts as a **legal reasoning assistant**, translating complicated text into an interactive map of consequences and precedents that activists can navigate. This not only saves time but also provides confidence – every highlighted point in the logic tree links back to the exact line in the source, so they can double-check and quote original language accurately when preparing reports or public comments.

- **Lawyers & Legal Professionals:** For attorneys, judges, and academics, SensibLaw accelerates legal research and argument development. Lawyers can use the case comparison feature to quickly spot how a fact pattern in a new case differs from a leading precedent – an exercise that traditionally involves reading dozens of pages and manually noting distinctions. The automated *treatment query* tool can summarize how later courts have treated a case (e.g., counting how many times it was followed, distinguished, overruled, with weightings for court hierarchy), which helps in assessing a precedent's strength. These are akin to features offered by premium legal research services, but SensibLaw provides them in an open, extensible platform. Additionally, the logic tree of a statute can aid in compliance analysis: a lawyer advising a client can extract all conditions and exceptions of a regulation to ensure the client meets each one, or generate a checklist of obligations from a contract. The **graph visualization** is useful for litigation strategy as well – for example, mapping out all cases related to a particular principle might uncover less-known authorities to cite. Academics and students, on the other hand, can use SensibLaw to study how legal reasoning is structured, experiment with scenario tests (the *story testing* feature allows writing expected outcomes and verifying them against a scenario), and even contribute new logic patterns for emerging legal concepts. Since the platform is extensible (with Python hooks and a modular pipeline), law school programs could customize it to local law and use it as a teaching aid in courses on legal reasoning or AI & law.

- **Individuals & Civic Tech Enthusiasts:** SensibLaw also embraces **open justice** and citizen empowerment. Individuals who want to understand a law or judgment that affects them can use the platform to break down the text. For example, a small business owner could upload a lengthy regulation and extract just the parts that say "you must do X" or "you cannot do Y," cutting through the legal jargon. Because the interface is visual and does not assume legal training, it can serve as a bridge between formal law and public understanding. Journalists and civic tech developers might similarly use SensibLaw to analyze legal documents as data – generating interactive explainer visuals or tracking changes in law over time. By providing an open API/CLI in addition to the UI, SensibLaw allows integration into other tools and workflows (such as civic websites that auto-annotate legislation, or chatbots that answer questions about rights and duties by consulting the logic graph). This flexibility ensures that the platform's benefits extend to anyone interested in **making sense of the law**, not just professionals.

Roadmap (Upcoming Features)

SensibLaw is under active development, continuously evolving to incorporate cutting-edge techniques and user feedback. The roadmap focuses on **features** (not timelines) that will expand the platform's capabilities and robustness. Key upcoming enhancements include:

- **Full spaCy NLP Pipeline Integration:** Completing the migration from regex-based parsing to a spaCy-powered pipeline. This involves **Named Entity Recognition (NER)** for legal terms (using spaCy's built-in NER augmented by custom **EntityRuler** patterns for laws, cases, etc.), **Dependency Parsing** for complex sentence structures, and assigning custom token attributes (like `token._.class_` tags for roles in the logic tree). This upgrade will deliver more accurate and maintainable text processing. The target state is a tokenization module that yields sentences with tokens annotated by lemma, POS, dependency, and linked to domain concepts, providing a solid backbone for all higher-level reasoning. *(This effort aligns with the broader industry trend of adopting mature NLP frameworks for legal tech, rather than reinventing basic text processing – a lesson learned from projects like Blackstone and others.)*

- **Multilingual and Jurisdictional Support:** Extending the pipeline to handle legal texts in multiple languages and formats. SensibLaw plans to incorporate language detection and route text to appropriate models (for instance, using **Stanza** or spaCy models for languages other than English). The architecture will accommodate jurisdiction-specific plugins – e.g., different patterns for civil code articles vs. common law cases. This will allow analysis of, say, European directives in French or Japanese court decisions, using the same core platform. It also involves mapping non-English legal terminology to the common ontology of concepts so that the knowledge graph can unify information across jurisdictions. Ultimately, this feature aims to make SensibLaw a **cross-border legal reasoning tool**, useful in comparative law studies or international policy work.

- **Text Extraction & Provenance Stack:** Improving the ingestion of raw documents, especially PDFs and scans. A containerized **text extraction microservice** is in the works, likely built on Apache Tika for parsing PDFs and OCR (Optical Character Recognition) tools for scanned pages. This will be paired with a *provenance sidecar* that logs how each text was obtained (e.g., which pages were OCRed, any content skipped like images or signatures) and attaches checksums for integrity. The deliverable will include a CLI command (e.g. `sensiblaw extract`) that takes a PDF and produces a structured JSON document (sections, paragraphs, etc.) ready for NLP, along with a *receipt* of the extraction process. By containerizing this, deploying SensibLaw's ingestion pipeline in cloud or on-premises environments (where large volumes of documents need processing) becomes easier. Ensuring deterministic text extraction is crucial – it means if two different users ingest the same PDF, they get the exact same text and section breakdown, which is important for collaboration and verification.

- **Gremlin-Compatible Pipeline & Node SDK:** Refactoring the internal processing pipeline into a series of **interoperable nodes** that can run in Apache Gremlin or similar workflow engines. This is a step toward scalability and integration: each stage (e.g., extraction, normalization, concept tagging, rule extraction, graph loading) will conform to a standard interface so it can plug into a distributed dataflow or be swapped out if needed. As part of this, a **Node SDK** will be provided to allow developers to write custom pipeline nodes or extend existing ones in a consistent way. Each node will produce outputs and metrics in a standardized JSON format, making the pipeline *gremlin-friendly* and easier to debug. The motivation is to allow SensibLaw's core logic to run in different contexts – whether as a simple local script, a cloud function sequence, or within a larger case management system – without modification. This modular design also means organizations can insert their proprietary steps (like a classifier or an external knowledge base lookup) at defined hook points. The roadmap includes publishing a template Gremlin DAG (Directed Acyclic Graph) that mirrors the default SensibLaw processing flow, and documentation for deploying it on platforms like Apache NiFi or AWS Step Functions.

- **Deterministic Logic Tree & Auditable Reasoning:** Formalizing the logic extraction into an **explicit state machine or graph** that can be examined and tested independently. Currently, the logic tree emerges from pattern matches and code – the goal is to have it defined in a declarative way (for example, a set of logical inference rules or a small DSL that describes how to go from text tokens to a logic graph). By doing so, we can better verify that every possible clause type is handled and that the process is **100% deterministic**. This also involves classifying and filtering out "junk" or non-normative text (like introductory phrases or examples) in a consistent manner, so the logic tree focuses only on enforceable content. The deliverable here will be a documented specification of the logic-building algorithm, tests with *golden examples* (where the expected logic graph for a given sentence is codified and used to validate the pipeline), and possibly a visualization tool that shows the step-by-step construction of a logic tree from a sentence. The end result will give users and developers confidence that the *same inputs always yield the same reasoning structure*, an essential property for legal automation and one that sets SensibLaw apart from probabilistic AI approaches.

- **Enhanced Reasoning Viewer (UX/UI):** Building out the Streamlit dashboard into a richer **reasoning interface**. Upcoming features include: interactive proof trees where users can click on any node to see details (e.g., definitions of legal terms, or links to evidence supporting that node); a timeline view for versioned documents (to compare how a section's text changed between revisions); and a graph neighborhood explorer that can animate the traversal (so one can *play* a sequence like "from this case, jump to cited case, then to that case's cited statute", visualizing the path taken). Another planned addition is an **annotation and collaboration layer** – for example, allowing a user to attach a note or tag to a node in the logic tree (like "needs verification" or "client says this condition is met") which can be exported or shared. This transforms the UI from a purely read-only analysis tool to a workspace where legal teams can perform tasks and mark insights. Given the sensitive nature of legal data, the interface will remain client-side and stateless by default (i.e., no data leaves the user's machine unless explicitly saved), but features to export a session (all derived data and graphs) as a report or bundle are on the roadmap. The guiding vision is a **"reasoning cockpit"** for lawyers and analysts: a single screen where they can see the text, the logic breakdown, related precedents, and their own notes, all aligned together.

- **Graph-Based Machine Learning & Insights:** With the accumulation of a rich legal knowledge graph, SensibLaw will incorporate **graph analytics and machine learning** to surface deeper insights. One exciting area is applying Graph Neural Networks (e.g., **Relational Graph Convolutional Networks, R-GCN**) to the legal graph. For instance, by embedding nodes (cases, provisions, concepts) in a vector space, the system could suggest similarities – *“Case X is structurally similar to Case Y in reasoning”* – or predict likely outcomes – *“a case with these factors has an 80% chance of being decided in favor of the defendant”* – based on patterns learned from the graph. Initial steps in this direction (already prototyped) include generating node embeddings and providing a **graph embedding explorer** in the UI. This could help users discover non-obvious connections: perhaps a tenancy law case from one jurisdiction shares a reasoning pattern with a consumer contract case from another, hinting at a general principle. Moreover, ML can assist in automating some of the manual rule tagging by learning from the annotated corpus; for example, a model could learn to label sections as *“penalty clause”* vs *“obligation”* by looking at enough human-tagged examples, thereby speeding up the processing of new document types. These features will be introduced carefully, keeping the focus on explainability – any ML-driven suggestion will be accompanied by a rationale or a set of supporting data points from the graph, to avoid the “black box” pitfall. In line with the project’s philosophy, such **neuro-symbolic** techniques augment rather than replace the core rule-based logic, providing a layer of insights on top of the solid deterministic foundation.

- **Extended Domain Coverage & Ontology Expansion:** SensibLaw’s knowledge model will continue to expand to cover more areas of law and types of documents. The roadmap includes incorporating **glossaries and ontologies** for specialized domains (e.g., finance, health, environment) so that domain-specific terms and concepts are recognized. For example, a finance regulation might mention “KYC requirements” or “Basel III standards” – hooking in an ontology of finance terms would allow the system to tag those and perhaps link to reference definitions. We are exploring integration with existing open legal ontologies and linked data – such as the LKIF (Legal Knowledge Interchange Format) ontology for legal concepts, or vocabularies published by sources like OpenLegislation or the EU’s Publications Office – to give SensibLaw a rich semantic understanding. In practical terms, this means **more powerful concept matching**: the triggers dictionary (used in concept matching) will grow, and it will be backed by a hierarchy (so the system knows, for instance, that “the Crown” is an Actor of type Government, or that “decide an appeal” implies a judicial action in appellate context). This expansion goes hand-in-hand with community engagement: as an open project, SensibLaw welcomes contributions of pattern files or glossaries for different legal systems. By broadening the ontology and patterns, we aim to make the platform useful for *all* of those stakeholders – from criminal law in Australia to international treaties to local bylaws – with minimal configuration.

Key Deliverables and Technologies

The development approach of SensibLaw emphasizes delivering concrete, usable components backed by modern technologies. Here are the key deliverables (current and upcoming) and *how* we’re achieving them:

- **1. Comprehensive NLP Processing Module:** *Tech:* **spaCy** (for English and other languages via models/adapters), with possible integration of **Stanza** for languages where spaCy models are less available. We implement a custom spaCy pipeline (`spacy_adapter.py`) that outputs tokens with legal-specific annotations. This module also uses **Regex and Matcher** rules where necessary (for fine-tuned legal patterns). The deliverable includes robust unit tests to ensure tokenization aligns exactly with the original text (no lost or altered characters). By building on spaCy, we stand on the shoulders of an NLP giant – benefiting from its speed and accuracy – while injecting our domain knowledge (via vocabulary lists, legal NER labels, etc.) to push it to state-of-the-art for legal text. The result will be a self-contained Python module that other projects could even reuse for general legal text parsing. *(Notably, this approach contrasts with purely statistical NLP on general corpora – by customizing spaCy's pipeline, we incorporate the domain semantics that generic models often miss, closing the gap identified in research where domain knowledge plus NLP outperforms raw AI on law [10] .)*

- **2. Legal Knowledge Graph Schema & Storage:** *Tech:* **Property Graph** model implemented via Python data classes and stored in **SQLite** (with JSON columns) or serialized to **GraphML/JSON-LD** for interoperability. We chose a property graph over a pure RDF triple store to allow more natural graph traversals and to easily attach metadata to nodes/edges (e.g., each edge can carry a citation weight, each node can have a text excerpt). However, the design doesn't preclude using an RDF or a graph database – in fact, the data schema is compatible with standard ontologies and could be loaded into Neo4j or Neptune if needed. A forthcoming deliverable is a **Gremlin traversal template** that demonstrates how the graph can be queried in Gremlin syntax (for those who choose to load it into a Gremlin-capable system). We also provide conversion tools (e.g., export the graph to CSV or Cypher scripts). The graph forms the **backbone** of SensibLaw's reasoning capability – it's what connects everything together. Our approach here is informed by state-of-the-art implementations like the Italian "Legis Graph" project, which showed that using a property graph with legislative XML inputs yields a highly queryable and consistent dataset [7] . By delivering an open schema and storage solution, we fill a niche for an **open legal knowledge base** that others can build on (whereas most legal knowledge graphs are proprietary or ad-hoc).

- **3. Text Ingestion & OCR Pipeline:** *Tech:* **Apache Tika** for parsing PDFs and extracting text (including metadata like author, titles, etc.), **pytesseract/OCRmyPDF** or similar for handling scanned documents, and a custom **PDF structure parser** for things like table of contents or page headers/footers. We wrap these in a **Docker Compose** stack, meaning one command can spin up the necessary services (OCR engine, etc.) and connect them to SensibLaw. The deliverable includes a CLI tool (`sensiblaw pdf-fetch`) that not only extracts text but also structures it into sections and subsections (using layout cues and regex heuristics for numbered headings). This is crucial for lengthy Acts or reports where we need to preserve the hierarchy (Part > Section > Subsection). We're also implementing **cover page detection** and **table of contents skipping** (to avoid treating a TOC as content), as indicated by recent updates. This extraction pipeline will produce a *provenance report* – for example, a JSON that lists each page's status (parsed or OCRed), any sections inferred, and any potential anomalies (like missing fonts or unrecognized characters). By using established tools like Tika (used widely in enterprise content management) and Tesseract, we align with best practices and ensure robustness. The added value from SensibLaw is the legal-specific handling on top (e.g., knowing that "Schedule 1" in a PDF likely starts an appendix we should capture as a unit). This deliverable is about reliability and ease of use: making sure that whether the input is a clean HTML, a PDF scan, or a bundle of documents, users can ingest them with confidence that nothing important is lost.

- **4. Modular Processing Pipeline & SDK:** *Tech:* **Python (FastAPI)** for any service wrappers, **Apache Gremlin** (planning phase) for orchestrating the pipeline, and containerization via **Docker**. SensibLaw's pipeline will be broken into distinct modules – e.g., `extract_text`, `normalize_text`, `match_concepts`, `extract_rules`, `build_graph`, `evaluate_tests` – each of which can run independently. The **Node SDK** will likely be a Python base class or interface each module implements, plus a set of schemas for inputs/outputs. For example, every node will accept a JSON payload with a certain schema (say, the output of the previous stage) and produce a JSON payload as result, along with logging. We will also provide a test harness for nodes to ensure they meet the deterministic requirements. In practice, this means an organization could swap out, say, the `extract_rules` node with their own enhanced version, and as long as it adheres to the interface, the rest of SensibLaw will accept it. The deliverable includes **documentation (docs/roadmap.md and contributing guidelines)** that describe how to add new nodes or run the pipeline in custom environments. By structuring the project this way, we future-proof it – if a new AI technique emerges for a given stage, it can be slotted in without rewriting the whole system. This design echoes patterns in large-scale data engineering and is inspired by the need for **interoperability**; much like how one can plug different storage engines under a graph API, we allow plugging different logic engines under the SensibLaw workflow. Ultimately, this modular pipeline will enable deployment in enterprise contexts (e.g., as microservices in a cloud architecture) and collaboration between teams on different parts of the system.

- **5. Deterministic Logic Tree Engine:** *Tech:* **Custom rule engine** in Python (using spaCy's matcher callbacks, perhaps a variant of Rete algorithm for rule matching) and formal grammars. We are developing the logic extraction as a rigorous engine where each inference (like "identify condition clause beginning with 'if'") is a defined rule. The key deliverables here are (a) a **specification** document detailing each rule/pattern the engine uses (for transparency and academic value), and (b) a **suite of golden tests** with expected logic outputs for various tricky sentences (e.g., multiple conditions, nested conditions, exceptions without explicit "if", etc.). The engine will also expose an **API** (possibly in the CLI and UI) to trace its decision-making. For example, a developer or power user could run a clause through the engine in debug mode to see: *"Matched pattern X at token 5-7 → labeled as ACTOR; Matched pattern Y at token 2 → labeled as MODAL; Constructed tree node for prohibition with children..."*. This trace is invaluable for debugging and for users to trust the system (especially if a clause isn't parsed as expected – they can pinpoint why). Compared to machine learning approaches that might output a parsed structure without explanation, our rule-based engine's *explainability* is a major deliverable. We will also integrate an **Open Policy Agent (OPA)** or similar policy logic to enforce any high-level constraints (for example, to ensure the logic tree always has at least one Actor and one Action for a well-formed rule, or to flag if something seems off). By making the logic tree generation explicit and testable, SensibLaw strives to be **provably correct** (or at least clearly wrong in known cases) rather than *probably correct*. This is critical in legal tech where a single missed "not" can invert the meaning of a rule.

- **6. Streamlit Dashboard & Reasoning Viewer:** *Tech:* **Streamlit** for the web app, with integrations of **Graphviz** (for graph rendering), **Plotly or D3.js** (for interactive visuals), and possibly **NetworkX** for on-the-fly graph algorithms. The deliverable is a polished, multi-page web application (packaged in `sensiblaw_streamlit`) that can be launched with a simple `streamlit run` command. We are focusing on a responsive UI that can handle large texts and graphs without choking (for instance, using pagination or progressive loading for documents with hundreds of sections). Each feature tab (Documents, Knowledge Graph, Case Comparison, etc.) is being developed with user feedback loops. There will be an **embed mode** for the viewer, so its visualizations can be embedded into other applications or reports (imagine an iframe that shows a mini proof tree for a specific clause, which could be included in an online article). The tech stack also considers *accessibility* – ensuring the color schemes and layouts are legible (important for long sessions reading text) and possibly adding options like dark mode or text-to-speech for statutes. The UI deliverable isn't just about looks; it's about bringing all the underlying tech to the forefront for end-users. We want someone with zero knowledge of AI or graphs to be able to use SensibLaw for practical tasks. Achieving this means lots of thoughtful UI/UX design, which we treat as a key deliverable alongside the code. In terms of technology, Streamlit has served well for rapid development, but we keep the architecture flexible so that a future dedicated web app (React/TypeScript) could consume the same backend – i.e., the logic is separated so the UI could be swapped without redoing functionality. In summary, this deliverable ensures **all the power of SensibLaw is accessible** to users through a modern, intuitive interface.

- **7. Integration of Graph Analytics & ML (Insight Layer):** *Tech:* **PyTorch Geometric** or **DGL (Deep Graph Library)** for training graph neural networks on the legal graph, and **scikit-learn** or **TensorFlow** for any additional models (like classification or clustering on text embeddings). While core parsing is symbolic, we recognize the value of machine learning in detecting patterns that rules might not catch (especially across documents). One deliverable in this vein is an **"AI assistant"** mode: a feature where the system can answer questions or suggest insights using a combination of the knowledge graph and a language model. For example, a user might ask in natural language, *"What are the key differences between the Native Title Act 1993 and its 1998 amendment?"*, and the system would utilize its structured data to produce an answer (perhaps by retrieving the relevant provisions and using a GPT-style model to summarize differences). This is essentially a **Graph+LLM** application – something recent research calls Graph-Retrieval-Augmented Generation [11]. The deliverable would be a proof-of-concept QA system with a constrained, fact-based generation (to avoid hallucinations by grounding the model with graph facts). Another deliverable is **case outcome prediction** on known datasets – using the factors extracted by SensibLaw as inputs to a model that tries to predict a decision (guilty/not guilty, win/lose) or a legal issue classification. The aim here isn't to replace human judgment but to highlight patterns (like "cases with Factor A, B, and C usually result in outcome D"). These AI-driven features are at the experimental end of the roadmap, but the groundwork is being laid now: by structuring data and ensuring provenance, we make it feasible to safely introduce ML on top. Users can always drill down from a model's suggestion back into the knowledge graph to see *why* that suggestion might make sense, a critical feedback loop that keeps the AI **sensible** (true to our name).

In sum, SensibLaw is ambitiously bringing together **state-of-the-art** techniques in NLP, knowledge representation, and human-computer interaction to create a **comprehensive legal reasoning platform**. It stands on the principles of openness, transparency, and domain-centric design, differentiating it from many black-box legal tech products. By comparing and incorporating the best from existing solutions – from Blackstone's legal NLP insights to Graph-based AI methods in academia – SensibLaw identifies gaps in current tools (like lack of explainability, or limited scope of automation) and targets those niches. The roadmap and deliverables above demonstrate *how* we plan to reach a feature-complete, impactful system: by using proven technologies (spaCy, Tika, Graph databases, Streamlit) in innovative combinations tailored to legal workflows. As we implement these features, we continue to engage with a broad community of stakeholders (government tech units, legal aid organizations, law firms, and civic hackers) to ensure SensibLaw "just makes sense" for real-world use cases – much like a good coleslaw, it aims to be a **refreshingly sensible mix** of ingredients that together enhance the experience of digesting even the densest legal information.

Sources:

- Kondo et al., *Capturing Legal Reasoning Paths from Facts to Law in Court Judgments using Knowledge Graphs* (2025) – identifying limitations of LLMs in legal reasoning [1] [2].
- Dhanushkodi, *AI-Augmented Graph Databases for Judicial Case Management* (2025) – on using cloud graph databases, NLP, and GNNs to reveal hidden legal patterns [5].

- John Snow Labs, *Legal NLP* – industry survey noting Spark NLP as a popular library for legal text processing [4].
- Open Source Legal "Blackstone" NLP – demonstrating spaCy-based legal entity recognition (case names, citations, etc.) [3].
- Colombo et al., *Legislative Knowledge Graph & LLMs (LegisAI)* – combining knowledge graphs with language models for legislative analysis [6] and leveraging standards like Akoma Ntoso for cross-jurisdictional consistency [7].

---

[1] [2] [10] [2508.17340] Capturing Legal Reasoning Paths from Facts to Law in Court Judgments using Knowledge Graphs
https://www.arxiv.org/abs/2508.17340

[3] Open Source Legal: Blackstone
https://opensource.legal/projects/BlackstoneNLP

[4] Legal NLP - Natural Language Processing for Legal Documents - John Snow Labs
https://www.johnsnowlabs.com/legal-nlp/

[5] AI-Augmented Graph Databases for Judicial Case Management: A Scalable AWS-Powered Framework for Relationship Analysis and Decision Support by Raja Mohan Dhanushkodi :: SSRN
https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5283555

[6] [7] [8] [9] [11] Leveraging Knowledge Graphs and LLMs to Support and Monitor Legislative Systems
https://re.public.polimi.it/retrieve/80d9c68c-5482-4a07-aaa8-f2c43832364e/cikm_paper.pdf