From Unstructured PDF Files to Contract Intelligence: A Local LLM RAG Pipeline for Extracting Demurrage Terms and Conditions

by

Stephen C. Day III

and

Wathila Ekanayake

Submitted on May 8, 2026, in Partial Fulfillment of the
Requirements for the Degree of Master of Applied Science in Supply Chain Management
at the Massachusetts Institute of Technology

ABSTRACT

Large energy companies depend on complex maritime shipping contracts to govern the transportation of commodities; however, these agreements are typically stored as unstructured text documents that are not easily searchable or analyzable. As a result, firms face significant challenges in proactively managing contractual financial and operational exposure such as shipping delay penalties, as manual review of lengthy contractual language creates substantial operational bottlenecks. To address this problem, we developed a localized and secure artificial intelligence system capable of automatically processing, organizing, and analyzing unstructured contract text without reliance on cloud-based or internet-hosted generative AI tools. The system transforms raw contractual language into a structured business intelligence asset, enabling subject-matter experts to rapidly identify weakened contractual protections and latent financial risks with a high degree of accuracy. Based on these outcomes, we recommend that the firm pursue a guided implementation of the technology to shift from reactive dispute resolution toward proactive contractual risk assessment prior to deal execution.

Advisor: Dr. Elenna Dugundji
MIT Research Director

Table of Contents

1. Introduction

The oil, gas, and chemicals industry operates through a globally integrated value chain in which profitability depends on the margin between commodity prices and operational efficiency across extraction, transportation, and refining activities. Marine transportation is a critical element of this value chain due to the scale of liquid-bulk movements required to support global sourcing, refining, and product distribution. Chartering agreements define the commercial and operational conditions under which vessels execute these maritime movements and therefore play a central role in supply chain reliability and cost performance.

This project examines spot voyage charter agreements, which bind a vessel to a single, discrete voyage. This contractual service obligation covers the transportation of cargo from the agreed load port(s) to the specified discharge port(s). Upon completion of discharge, the vessel is released from the charter and becomes available for redeployment.

Spot charter agreements are executed between the employing party and the commercial operator, that is, the entity that is authorized to commit the vessel to a commercial transaction. The commercial operator may be a time-charterer, who has secured the vessel via a long-term charter for a specified period and markets its capacity on the spot market, or the vessel owner, when ownership and commercial operations are carried out by the same entity. Regardless of the ownership structure, the contractual relationship for the voyage is always formed with the commercial operator.

1.1 Motivation

Spot chartering gives the sponsor company the flexibility to rapidly respond to evolving market conditions, refinery scheduling changes, and short-term commercial opportunities. Depending on fleet positioning and optimization requirements, the sponsor company may participate in these agreements either as the spot charterer (hiring a vessel for a voyage) or as the commercial operator (chartering out a vessel it controls under ownership or time-charter commitments).

The sponsor company maintains a large repository of maritime chartering documents, including four key types that collectively form a contractual hierarchy governing a single agreement:

1. Charter Party – The foundational template containing standardized terms and conditions.
2. Recapitulation (Recap) – The authoritative record of negotiated commercial terms for the voyage; it overrides conflicting provisions in all other documents.
3. Riders – Amendments that modify specific clauses within the Charter or Recap..
4. Addendums – Supplements that introduce additional terms without altering existing ones.

This project primarily focuses on charter parties and recapitulations. Riders and addendums were not included in the scope of work.

Like many companies involved in marine transportation, the sponsor company typically stores its chartering documents in PDF format. Because this information remains largely unstructured and difficult to query, it is effectively dormant, forcing operational teams to spend substantial time manually extracting insights from historical contracts to support routine commercial decisions.

For the sponsor company, the primary catalyst for addressing this issue is the need to better anticipate and manage demurrage exposure. Demurrage is the penalty assessed when loading or unloading operations exceed the contractually allotted time for loading and unloading operations, known as “allowable laytime.” One of the largest and most contested line items in voyage economics (Blue-C, n.d.), it compensates the commercial operator or vessel owner for lost earnings when delays impede vessel turnaround. Contractually negotiated demurrage provisions typically define laytime rules, penalty rates, exceptions, documentation requirements, and dispute-resolution procedures. Effective interpretation of these clauses is essential, as demurrage costs can materially impact voyage economics and overall supply chain performance.

At present, answering even a basic demurrage-related question requires an analyst to manually review multiple PDF files, navigate extensive redlined contractual language, and reconcile inconsistent or superseded terms across voyage agreements that routinely span 40 to 60 pages. This process can take anywhere from 30 minutes to several hours per contract, creating a significant operational bottleneck. As a result, demurrage management remains largely reactive: discrepancies often are identified only after a vessel has sailed. In such cases (whether the company is responding defensively to a demurrage claim or failing to promptly invoice a counterparty for incurred delays) the commercial opportunity to negotiate or enforce contractual rights from a position of strength typically has already passed.

1.2 Problem Statement and Key Questions

The archived PDF format makes it difficult for the sponsor to efficiently access, interpret, and standardize contractual terms. These challenges hinder the company's ability to quantify demurrage exposure, ensure consistent contractual compliance, and optimize voyage decisions. The core problem of this project, therefore, is how to utilize artificial intelligence (AI) and machine learning (ML) to convert a massive collection of contracts stored as unstructured PDF files into a structured data asset that accurately respects the complex hierarchy of maritime agreements.

Addressing this requires solving two primary technical challenges:

- Data Ingestion and Pre-Processing: Accurately extracting text while respecting PDF file layouts, preserving crucial legal defined terms, and deliberately dropping strikethrough redlines (which often remain as artifacts in the final PDF contract from negotiations) so deleted text does not interfere with the knowledge base.
- Large Language Model (LLM) Inference: Building a local Retrieval-Augmented Generation (RAG) system that allows users to query the factual, source-based contract data with natural language and receive synthesized answers that respect contract hierarchy without relying on external cloud application programming interfaces (APIs) that would forfeit data privacy.

1.3 Project Goals and Expected Outcomes

The primary goal of this project is to build an ingestion and analysis pipeline that distills voyage agreement documents into a structured, machine-readable format. This system will be accessed via a centralized user interface and will deliver the following core capabilities to the project sponsor:

- Fact Retrieval: A natural-language interface that provides accurate answers regarding contract terms, supported by in-line citations that link directly to the exact page and chunk (a segmented passage of cleaned contract text) of the source PDF file.
- Contract Compare: A tool that compares any proposed contract against a baseline template to systematically identify and describe missing protections, weakened obligations, added risks, and altered financial thresholds.
- Executive Summarize: A tool that condenses an entire agreement into a scannable, seven-section executive summary.

The resulting data-driven methodology for extracting and organizing demurrage-related information from unstructured contract documents will support improved contract governance, operational efficiency, and commercial optimization across the company's marine logistics activities. By achieving these outcomes, the project will transform static PDF files into a strategic decision-support asset, enabling pre-voyage risk mitigation and providing a foundation for fleet-wide commercial intelligence.

2. State of the Practice

The architecture required to automate the analysis of unstructured voyage agreements relies on synthesizing several advanced technological domains. The current state of the practice combines document processing,

semantic search, and on-device LLMs to power local, privacy-preserving question-answering, contract comparison, and executive summarization.

State-of-the-art pipelines now combine document chunking, hybrid retrieval, and RAG to parse unstructured PDFs and generate precise, cited answers. The advent of efficient, on-device LLMs allow organizations handling sensitive commodity contracts to execute the entirety of the intelligence-gathering process securely and entirely on-premises.

2.1 Document Parsing, Layout Analysis, and Semantic Chunking

Modern systems must use layout-aware extraction to convert unstructured PDFs (Omdena, n.d.) into RAG-ready text chunks equipped with precise metadata. This foundational step dictates the success of the entire pipeline, directly impacting downstream recall and fidelity.

Typically, text is extracted and cleaned using pdfplumber (Saravanan, n.d.), a Python library designed for deep document inspection that reads both text and underlying geometric page shapes. The cleaned text is then divided using recursive character splitting. This technique progressively breaks down the text using a hierarchy of separators, such as paragraphs and sentences, to preserve meaningful context. The process targets roughly 1,500 characters per chunk with a 300-character overlap.

In practice, this approach is operationalized through extraction libraries that capture character bounding boxes (a set of coordinate points defining a precise rectangle around a specific text chunk or object) and detect strikethrough redlines. By recognizing exactly where vector lines intersect character midlines, the system filters out deleted or obsolete clauses prior to chunking. This critical step ensures that strikethrough contract (Omdena, n.d.) text never pollutes the knowledge base.

2.2 Semantic Retrieval and Vector Databases

Extracting relevant provisions from complex, unstructured legal documents is a foundational challenge in modern contract management. To address this, the current industry standard relies on hybrid information retrieval, which fuses contextual semantic understanding with precise exact-match search. Achieving this capability typically requires synthesizing several core technologies into a unified pipeline:

- Traditional Lexical Indexing (Elastic, n.d.-a): Foundational layer of search, which relies on keyword-based methodologies to match exact words or phrases between a user's query and a document.
- BM25 (Elastic, n.d.-b): Highly effective, specific algorithmic implementation of lexical search. BM25 ranks document relevance based on term frequency (how often a word appears) and inverse document frequency (how rare the word is across the whole corpus).
- Embeddings (Lewis et al., 2020): High-dimensional mathematical vector representations of text. Unlike lexical indexing, embeddings capture the underlying semantic meaning and context of a passage, allowing a system to recognize that, for example, "penalty" and "demurrage" are related even if the exact keyword is not used.
- Reciprocal Rank Fusion (RRF): (Cormack et al., 2009) A mathematical sorting algorithm used to bridge the gap between traditional indexing (Lexical, BM25) and embeddings. It takes disparate ranked lists from multiple search methods and combines them into a single, unified ranking without requiring complex, manual weight calibration
- ChromaDB (AltexSoft, n.d.): Specialized, open-source vector database designed specifically to store dense embeddings and execute fast, scalable nearest-neighbor semantic lookups (a search method that finds the most relevant information by mathematically identifying the stored data points whose underlying meaning is closest to your query, rather than relying on exact keyword matches).

- SQLite (SQLite, n.d.): Lightweight, serverless database engine. Unlike traditional databases that require a separate background process, SQLite stores its entire dataset in a single local file. The system is designed to operate exclusively on device, providing secure handling of structured, non-vector data while eliminating the need for external network connections.
- Cross-Encoder Reranking (Reimers & Gurevych, 2019): A secondary neural processing stage that evaluates the user query and retrieved document chunks simultaneously, rather than independently, to generate a highly accurate relevance score. This final filter ensures the system only feeds the highest-quality evidence to the LLM, drastically reducing hallucinations (made up information) caused by poor initial retrieval.

2.3 RAG, LLM Integration, and Verifiable Citations

In current industry practice, the RAG (Retrieval-Augmented Generation) framework serves as a foundational architecture for grounding large language models. By querying an external knowledge base prior to generating text, RAG bolsters domain expertise, diminishes hallucinations, and ensures that AI outputs remain strictly tied to source documents.

Within this framework, the Large Language Model (LLM) synthesizes retrieved data to produce credible answers. A key focus in the field is the generation of verifiable outputs. To achieve this, LLMs are guided by strict citation protocols. Techniques such as prompt engineering and logit bias (a method used to mathematically penalize the probability of a model generating specific characters or phrases) are utilized across the industry to compel models to emit consistent citation tags when referencing source material.

To bridge the gap between AI generation and human verification, these citations must be actively viewable. The state of the practice frequently relies on robust rendering tools like PDF.js. (Mozilla Foundation, n.d.) As an open-source JavaScript library developed by Mozilla, PDF.js parses and renders PDF files natively within web browsers. In enterprise document applications, libraries like PDF.js are integrated to translate LLM-generated citation tags into interactive, rendered document views. This capability allows end users to instantly verify extracted information against the original text, establishing trust and ensuring output quality assurance.

2.4 Efficient LLM Inference on Edge Devices

Running large parameter models locally demands aggressive model optimization. Current industry evaluations reveal that heavy quantization and compression (mathematical techniques that reduce the memory footprint of large AI models by lowering the precision of their weights) make inference highly feasible on constrained hardware. In practice, organizations are now successfully running multi-billion-parameter models, such as concurrent chat and embedding LLMs, on single, unified-memory edge workstations.

By leveraging execution frameworks like llama.cpp and accelerated hardware backends, the industry has proven that complex AI pipelines can be economically deployed on commodity hardware, eliminating the reliance on cloud GPU access. Integral to this shift is the GGUF binary format, which is highly optimized for fast loading and execution. GGUF supports advanced quantization and has been widely adopted across the field because it drastically shrinks the memory footprint of massive neural networks. This standardization allows enterprise-grade models to fit securely within a local machine's memory rather than requiring distributed cloud computing clusters.

2.5 Research Synthesis

The intersection of these distinct research areas is transforming how organizations handle unstructured data. Rather than operating in silos, industry leaders increasingly deploy these tools in tandem to create cohesive, automated intelligence workflows that solve the operational bottlenecks of traditional manual review.

At a high level, the state of the practice begins with layout-aware document parsing and semantic chunking. Together, these tools ensure that complex legal agreements are accurately digitized, contextualized, and filtered of obsolete information. This clean, structured text is then ingested into vector databases, setting the stage for hybrid information retrieval. By fusing traditional exact-match keyword searches with advanced semantic understanding, modern systems can instantly identify and retrieve highly relevant clauses across vast document repositories.

These retrieved components are then fed into a RAG framework. This integration is what makes the technology viable for enterprise use. Instead of relying on a language model's general, pre-trained knowledge, RAG forces the AI to synthesize answers strictly anchored to the retrieved text, outputting verifiable facts complete with inline citations and visual provenance.

A major shift in how these tools are utilized today is the push toward secure, edge-based deployments. By leveraging heavily quantized LLMs and local databases on unified memory hardware, organizations can now achieve the analytical power of advanced AI without exposing sensitive corporate contracts to external cloud networks.

3. Methodology

The overarching methodology of this project was to design and deploy a completely local, privacy-preserving RAG pipeline. This system is architected to ingest unstructured, highly sensitive maritime voyage agreement PDF files and transform them into a structured, queryable data asset without relying on external cloud APIs. The methodology encompasses a rigorous sequence of hardware configuration, meticulous data pre-processing, localized LLM inference, and a highly constrained generative framework designed to produce strictly cited, verifiable outputs. In the following sections, we discuss the data, hardware, software, and LLM specifications that this methodology utilizes. Then, we discuss the methodology's two stages: data ingestion/preprocessing and LLM inference and tool use. Figure 1 presents a high level overview of the methodology.

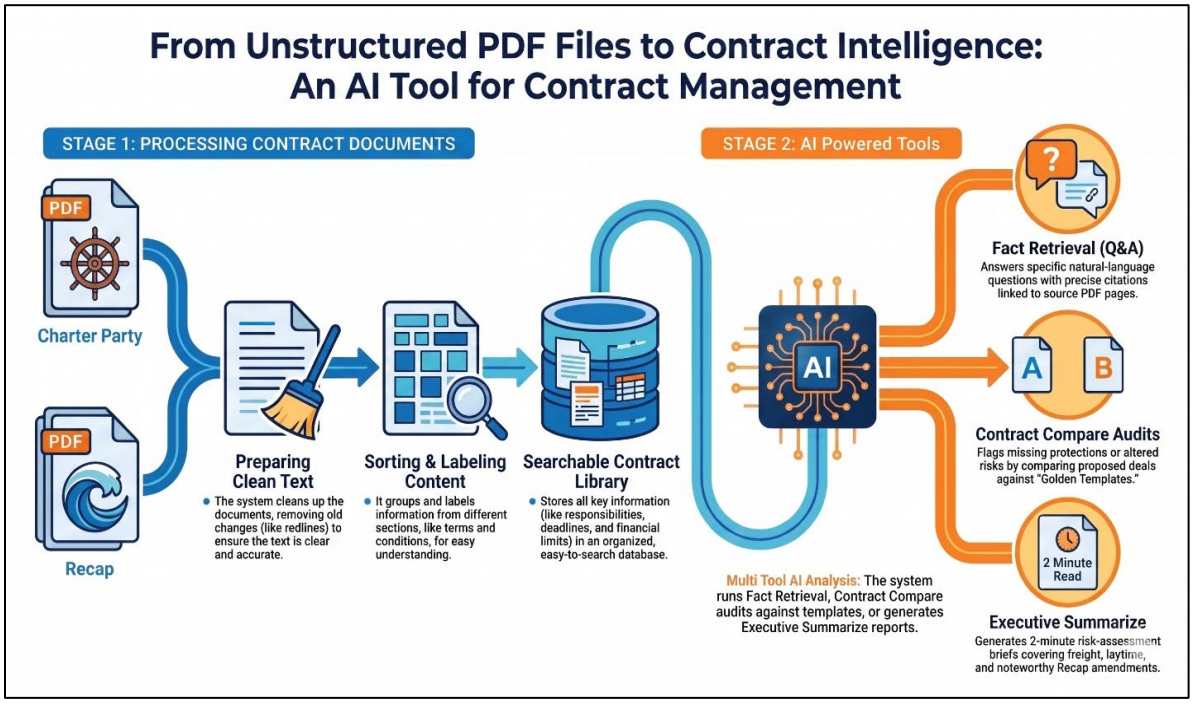

Figure 1. Process Flow Diagram

3.1 Data Collection

All data utilized in this project was provided by the sponsor company. The dataset consists of 10 CP and Recap combinations (five voyage agreements), which collectively form five historical, unstructured maritime agreements stored in PDF file format. This corpus serves as the “truth anchor” for both the ingestion pipeline and the subsequent evaluation of the RAG system's retrieval and synthesis capabilities. This methodology was constructed with scale in mind so that the sponsor company can utilize a larger corpus of data in the future.

3.2 Hardware Selection

The foundation of this system is a specialized hardware architecture designed to deliver data-center AI performance directly into a standard office environment without relying on the cloud. By utilizing a compact Minisforum workstation (ServeTheHome, n.d.) equipped with an advanced AMD processor and a substantial 128 GB pool of unified memory, the setup solves the biggest bottleneck in local AI: memory limits. Because 96 GB of this memory is permanently dedicated to graphics and compute workloads, large-scale AI models can be executed entirely locally with high performance. This approach guarantees complete data privacy, eliminates the need for expensive external server rooms, and proves that enterprise-grade AI can be run securely on-premise. Table 1 breaks down the hardware specifications used in this project:

Hardware	Specifications	Application
Workstation	Minisforum MS-S1 Max A fanless-class, compact desktop computer.	Allows the system to run quietly in a standard office environment. Because it cools itself passively, the project does not require an expensive, climate-controlled server room.
Integrated Processor	AMD Ryzen AI Max+ 395 ("Strix Halo") A System-on-Chip that physically combines the CPU, GPU, and memory controllers onto a single silicon chip.	Maximizes data transfer speeds by keeping core components intimately connected. This allows the system to process complex AI tasks entirely locally, requiring zero cloud reliance.
Graphics Processor (GPU)	Integrated RDNA 3.5 GPU (iGPU) A graphics processor built directly into the main chip that shares physical memory with the rest of the system.	Handles the heavy mathematical lifting of the AI models. By utilizing shared system memory instead of isolated memory, it bypasses the physical limits of standard plug-in graphics cards.
Total System Memory (RAM)	128 GB Total Unified Memory (UMA) The high-speed, temporary workspace the computer uses to actively hold data.	This architecture provides a substantially larger memory pool, ensuring that large AI models and the operating system can run concurrently without resource contention. This is an outcome that would be infeasible on a standard laptop.

Video Memory Partition (VRAM)	96 GB UMA Allocation	Enabling the system to operate with capabilities comparable to an ultra-high-end enterprise server. It ensures that both the 26-billion-parameter language model (approximately 16 GiB) and the embedding model (approximately 4.5 GiB) remain resident in fast local memory, delivering data-center-level performance without degradation or instability.
BIOS Firmware (Pre-boot Firmware Configuration)	The foundational program that initializes the hardware before the operating system loads.	Used to permanently lock in the 96 GB VRAM partition. Whereas standard systems dynamically share memory across workloads, this fixed partitioning guarantees that large, contiguous blocks of memory are reserved exclusively for AI workloads.

Table 1. Hardware Information

3.3 Local LLM Inference Infrastructure

To execute these operations securely and efficiently on the chosen hardware, the system utilizes a specific, highly optimized software stack:

A native Linux installation (Ubuntu 24.04 LTS) is deployed rather than Windows. Linux offers superior resource management and significantly better developer tooling local LLM inference (Linux Kernel Documentation, n.d.). Utilizing Linux eliminates the excess software that is often bundled with Windows. This grants the generative AI stack "bare-metal," unthrottled access to the GPU and memory controllers, maximizing overall system stability and text generation performance.

The core LLM generative engine is built on llama.cpp, an open-source inference framework originally developed by Georgi Gerganov and actively maintained by a large global open-source community (Gerganov, n.d.). This C/C++ software library is explicitly designed to run LLMs locally with minimal resource consumption. In contrast to heavyweight, Python-based enterprise inference frameworks (such as vLLM or PyTorch-based alternatives) which introduce substantial dependency overhead and frequently require dedicated data-center-grade GPUs, llama.cpp is deliberately lightweight and broadly portable. Its support for advanced quantization techniques enables the deployment of large language models on consumer-grade hardware while maintaining strong inference performance and minimal degradation in model capability.

A low-overhead, cross-platform 3D graphics and computing API is required to compile llama.cpp. Vulkan (Khronos Group, n.d.) accomplishes this by allowing the system to bypass the hardware's CPU and offload the heavy neural network matrix multiplications directly to the AMD iGPU. For the stack's performance, this parallelization is transformative: it drastically accelerates inference times, ensuring the natural-language chat interface remains fluid and responsive for the end user rather than stalling between words.

Like the LLM engine compiler Vulkan, ONNX Runtime (Microsoft, n.d.) is an open-source machine learning accelerator utilized specifically to run the system's cross-encoder reranking model. During the retrieval phase, the cross-encoder must rapidly score and sort dozens of extracted document chunks against the user's query. By leveraging specialized software to offload computationally intensive tasks to the graphics processing unit (GPU), the system achieves highly parallel data processing rather than sequential execution. This keeps the project tools from getting jammed up, ensuring the user get answers significantly faster.

3.4 LLM Selection

Deploying a state-of-the-art RAG pipeline entirely on-premises requires carefully balancing a model's capabilities against the strict memory limits of local hardware. To achieve this, both the embedding model and the generative language model in this architecture are deployed using the GGUF (GPT-Generated Unified Format) file format.

3.5 Embedding Model

The system's retrieval layer utilizes Alibaba's open source Qwen3-Embedding-8B-Q4_K_M local embedding model.

The primary mission of the embedding model is to map unstructured text (both the PDF file contract chunks and the user's natural language queries) into a shared mathematical vector space. By converting text into high-dimensional numerical vectors, the system can perform semantic similarity searches. This allows the system to retrieve contract clauses based on their underlying meaning and intent, rather than failing when a user's query lacks the exact keywords printed in the document.

This specific 8-billion-parameter model was selected because it offers state-of-the-art semantic understanding for dense retrieval while maintaining a highly compact memory footprint of approximately 4.4 GiB. This compact size is critical, as it allows the embedding model to reside entirely in the VRAM simultaneously alongside the larger generative model, eliminating the latency of swapping models in and out of memory during a chat turn. (Applied Sciences, 2026)

3.6 Cross-Encode Model

In the retrieval pipeline, the cross-encoder (BAAI/bge-reranker-v2-m3) serves as the final, high-precision reranking stage. While the initial BM25 and dense ChromaDB searches focus on broad recall to ensure no potentially relevant text is missed, the cross-encoder acts as the quality backbone by strictly optimizing for precision. It achieves this by scoring the fused shortlist of search results as (query, chunk) pairs, which allows the model to evaluate the user's prompt and the contract excerpt together simultaneously, rather than relying on their separate, isolated vectors. The classification model produces a single relevance logit for each input pair, and its extensive multilingual vocabulary is particularly well suited to this domain, as maritime charter parties frequently reference foreign port names and non-English governing-law clauses. Ultimately, only the top-K chunks that survive this rigorous cross-encoder scoring are passed to the chat LLM as retrieved context.

3.7 Generative Language Model

The generative core of the system is powered by Google's open-source Gemma-4 26B A4B Instruct (Cloudflare, 2026), also utilized in a dynamically quantized Q4_K_M GGUF format.

The mission of the generative model is to act as the reasoning engine. It evaluates the exact semantic chunks retrieved by the embedding model, actively resolves legal precedence conflicts (e.g., enforcing that the Recap supersedes the CP), and synthesizes that evidence into highly structured, plain-language outputs complete with inline citations.

This model employs a Mixture-of-Experts (MoE) architecture, in which a large pool of parameters is selectively activated during inference. Although the model contains approximately 26 billion total parameters, which supports deep reading comprehension and legal reasoning, only around 4 billion parameters are engaged for any given inference step. This architectural design combines the representational capacity of a large-scale neural network with the token-generation efficiency of a significantly smaller model, enabling high-performance operation within an approximate 16 GiB VRAM footprint (Cloudflare, 2026).

3.8 Data Ingestion and Preprocessing

Transforming raw PDF files into a machine-readable, highly structured data asset requires a specialized, multi-stage ingestion pipeline. This preprocessing pipeline is executed once per uploaded agreement pair, operates entirely on device, and distributes the resulting outputs across multiple local storage formats to support a range of downstream analytical tasks.

The first stage of the ingestion pipeline is document storage and archiving. Before text extraction begins, the uploaded PDF files are copied in local directory using a standardized naming convention. Archiving the

documents in this persistent location is a prerequisite for the system's Visual RAG citation capability (discussed below).

The second stage of the ingestion pipeline is layout-aware extraction and strikethrough handling. Within this stage, the first active processing step utilizes pdfplumber. The system uses this library to extract text alongside its exact character coordinates.

Strikethrough text often remains in final CP agreements as an artifact of the negotiation and drafting process. The system identifies strikethrough text by detecting where vector lines in the PDF file’s drawing layer intersect a character's midline. Contiguous redacted runs are either dropped entirely or wrapped in tags prior to cleanup. This prevents obsolete language from polluting the system's semantic embeddings or the LLM's context window, which acts as the model's short-term memory and dictates the maximum volume of text it can hold during a single interaction.

The third stage of the ingestion pipeline is semantic chunking. Once the chunks are created, they are enriched with a robust metadata dictionary. This metadata captures the agreement id, document type (CP or Recap), and an explicitly assigned hierarchy level to enforce legal precedence later in the pipeline. The system also captures spatial metadata for every chunk, including the page number and a specific bounding box. To ensure citation accuracy, this bounding box capture utilizes four tiers of precision, ranging from a tight line match around specific sentences down to a page estimated location if precise coordinates cannot be resolved. All of this information is then stored in memory as a highly structured JSON file (a lightweight, text-based format used for organizing and transporting data) which acts as the immutable, foundational source of truth for the entire text corpus.

The fourth stage of the ingestion pipeline is vector embedding and indexing. Once the JSON source of truth is generated, the pipeline executes the embedding phase. Rather than embedding chunks one by one (computationally inefficient) the system sends the chunks in consolidated batches to the local Qwen3 embedding model via an API call. The resulting high-dimensional vectors are then "upserted" (a database operation that inserts new records or updates existing ones) into ChromaDB. This persistent, local vector database maps the embeddings to their respective chunk IDs, establishing the infrastructure required for the system's rapid semantic search capabilities to come in the LLM inference and tool usage stage.

The fifth and final stage of the ingestion pipeline is the clause inventory distillation. This stage occurs in parallel with the embedding process. The system executes a four-way LLM map-reduce operation (LangChain, n.d.). Map-reduce is a computational paradigm that splits a massive task into smaller parallel operations (the "map" phase) and then synthesizes the results (the "reduce" phase).

During ingestion, the generative LLM reads through the document pair to extract every discrete clause, obligation, liability cap, and financial threshold. It distills the dense legal language into a structured database table called the "clause inventory," mapping each extracted fact to its original chunk reference. This inventory is saved locally in a SQLite database, enabling the system's compare and summarize tools to execute fleet-wide audits instantly without having to repeatedly retrieve raw text chunks.

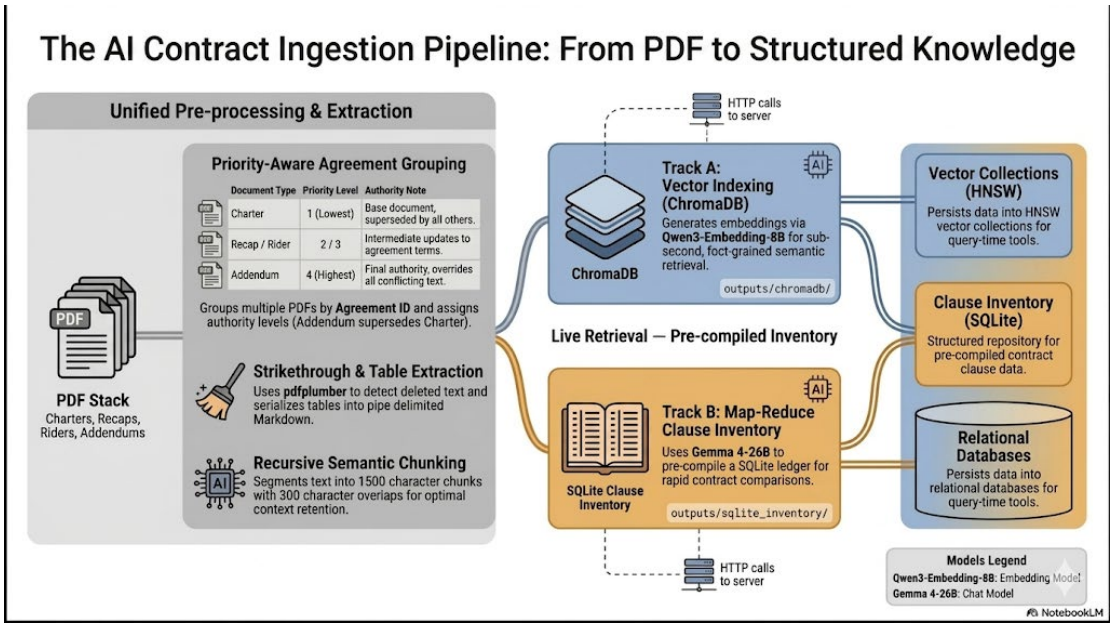

Figure 2. Data Ingestion and Preprocessing Overview

3.9 The UI, Hierarchy Enforcement, and Tool Usage

The sponsor company's access to this project is managed through a single, secure web dashboard that handles all the technical heavy lifting. It automatically runs the AI servers and processes contract data, hiding the complex infrastructure from the user. By simply uploading a PDF, the system automatically reads the document, organizes the information, and saves it into a permanent legal database.

Through this interface, users access three distinct tools supported by a smart context-assembly module. This module actively manages the AI's 32,768-token limit to prevent system failure during complex operations, using "budget drops" (removing irrelevant data) and "shrink logic" (compressing information) to avoid prompt overflow.

3.1.1 Hierarchy Enforcement: Recap Overrules Charter Party

Our sponsor company's legal hierarchy dictates that Recap terms always supersede base Charter Party terms. This logic is strictly enforced in all Fact Retrieval, Contract Compare, and Executive Summarize through a defense-in-depth strategy spanning seven distinct architectural layers. At the data and ingestion levels, chunks are tagged with numeric hierarchy scores and embedded semantic context strings that state the Recap's priority, ensuring that any conflicting financial or operational terms are seamlessly collapsed in favor of the Recap when the foundational Clause Inventory is compiled. During the live chat phase, the hybrid retrieval pipeline applies a tie-breaking RRF score boost to Recap chunks, while the document organizer physically orders the final retrieved excerpts, so the LLM reads the Recap before the Charter. Finally, the system prompts inject mandatory hierarchy rules across all analytical modes, explicitly directing the model to favor the Recap so that the operational guarantee is preserved even if previous systemic layers fail.

3.1.2 Tool One: Fact Retrieval

The primary goal of this tool is to answer ad-hoc, natural-language questions regarding contract terms, ensuring every answer is strictly grounded in the source text. To accomplish this, the tool executes hybrid retrieval. Step one combines BM25 lexical search with dense vector search via Reciprocal Rank Fusion. This fused list is then processed by an ONNX-accelerated cross-encoder reranker to isolate the top 10 most relevant chunks for the LLM's context window. To address a common RAG limitation where broad

questions miss short tabular data, this tool also utilizes an "inventory-augmentation layer." This layer cross-encoder-scores the user's prompt against the pre-compiled clause inventory from the voyage agreement's ingestion map-reduce, injecting up to six highly relevant structured database rows directly into the prompt.

Driven by a specific system prompt (a foundational set of background instructions that defines a LLM's role, constraints, and operational behavior), this tool relies heavily on injected hierarchy rules (which instruct the model to prioritize Recap values over CP values in the event of a conflict) and strict citation formatting rules. The resulting output is a natural-language answer where every factual claim is anchored by an in-text citation.

Subsequent prompts are routed via a multi-turn intent classifier. A drill-down intent triggers an advanced retrieval step that actively hydrates the prompt with previous citations and executes highly targeted topic retrieval based on the ongoing conversation, while a new topic intent clears the previous context to retrieve fresh evidence.

The report generated by this capability fundamentally accelerates the day-to-day workflow of maritime analysts and traders. Instead of spending hours manually cross-referencing a base CP against superseding Recaps to find a term, users can ask a natural language question and receive an instant, verifiable answer. This drastically reduces the time spent investigating disputes, limits human error in contract interpretation, and equips the commercial team to resolve contract opacity with unprecedented speed and accuracy.

3.1.3 Tool Two: Contract Compare

This tool is designed to systematically evaluate a proposed voyage agreement against a pre-approved baseline to describe potential commercial risk and flag specific portions of the voyage agreement for negotiation. It operates in two stages. Stage One streams the entirety of both documents' Clause Inventories to the LLM to outline the audit. During this stage, the user's actual chat input is demoted to a low-priority "hint," and a system prompt is injected to ensure the model executes the full audit rather than providing a short answer. Stage Two executes a targeted coverage retrieval step. Instead of a general search, it specifically fetches raw text chunks from the proposed agreement corresponding to up to four currently uncited baseline categories, ensuring maximum audit coverage without exceeding the context window.

Contract Compare stage one and stage two both utilize unique system prompts that drive the tool's highly structured, five-section Markdown (a lightweight text formatting language) report output. This specific output details missing protections, weakened obligations, newly added risks, altered financial thresholds, and an overall risk verdict, complete with bilateral citations mapping to both the baseline and proposed documents. For follow-on behavior, a DRILL intent runs dual topic retrieval across both the baseline and proposed agreements. It retains the exact same five-section headings but generates only highly specific, on-topic bullets related to the user's follow-up question.

The compare tool's output report shifts the organization's demurrage management strategy from reactive to proactive. By auditing a proposed contract against the standard baseline before it is signed, traders can instantly visualize where a counterparty has attempted to weaken obligations, strip away standard protections, or alter financial thresholds. This arms the commercial team with a precise, targeted list of clauses to renegotiate, preventing unfavorable terms from ever entering the active fleet and mitigating millions of dollars in potential downstream demurrage exposure.

3.1.4 Tool Three: Executive Summarize

The goal of this final tool is to condense an extensive voyage agreement into an easily scannable executive brief. The summary is generated directly from the structured SQLite clause inventory created during the initial ingestion map-reduce phase.

Using a specialized system prompt, the tool outputs a consistent, seven-section brief covering key parties and vessel details, freight rates, laytime and demurrage, key obligations, liability caps, noteworthy recap

amendments, and a summary risk assessment. Upon completion of the brief, the system automatically migrates the chat state to the Fact Retrieval tool. This ensures that any subsequent drill-down questions from the user naturally revert to deep hybrid retrieval and rigorous inline citations.

Executive Summarize is designed to provide high-level commercial visibility and enable rapid triage of contractual risk. The generated report relieves portfolio managers and senior traders from the need to review lengthy charter-party documents *(*often exceeding 50 pages of dense legal language*)* simply to understand the economic fundamentals of a proposed deal. By delivering a standardized executive-level brief that explicitly highlights contractual outliers and material risks, the tool enables leadership to quickly and consistently assess fleet-wide contractual exposure.

3.1.5 Citations and Verification with PDF File Viewer

The value of a descriptive, legal-domain AI system is fundamentally tied to its verifiability. While the generation of accurate insights is essential, such outputs are only operationally viable when they are fully auditable and transparently verifiable by human experts. In the context of commodity trading, analysts must be able to directly inspect the underlying contractual language and confidently reference the exact source text when engaging with counterparties, particularly in the event of a commercial dispute.

The AI system is governed by strict requirements to cite verifiable sources for every factual assertion it produces. To enforce this constraint, the architecture incorporates a background filtering mechanism that actively suppresses improper formatting and internal system artifacts. This design prevents the accidental exposure of intermediate or non-human-readable data, ensuring that all outputs consist of clean, consistent, and directly navigable citation links suitable for expert validation.

To close the loop on verification, the system connects these technologies into a Visual RAG PDF Viewer using PDF.js. Because every chunk of text captures precise spatial metadata (including page number, page width, and coordinate bounding boxes) during the initial ingestion phase, the web interface dynamically transforms the model's in-text citation tags into interactive buttons. Clicking a citation instantly opens the air-gapped PDF.js viewer. The system programmatically calculates the necessary scaling factors and overlays a translucent yellow bounding box directly onto the exact paragraph from which the model derived its answer. This allows a human analyst to immediately verify the model's response in its original typographical context.

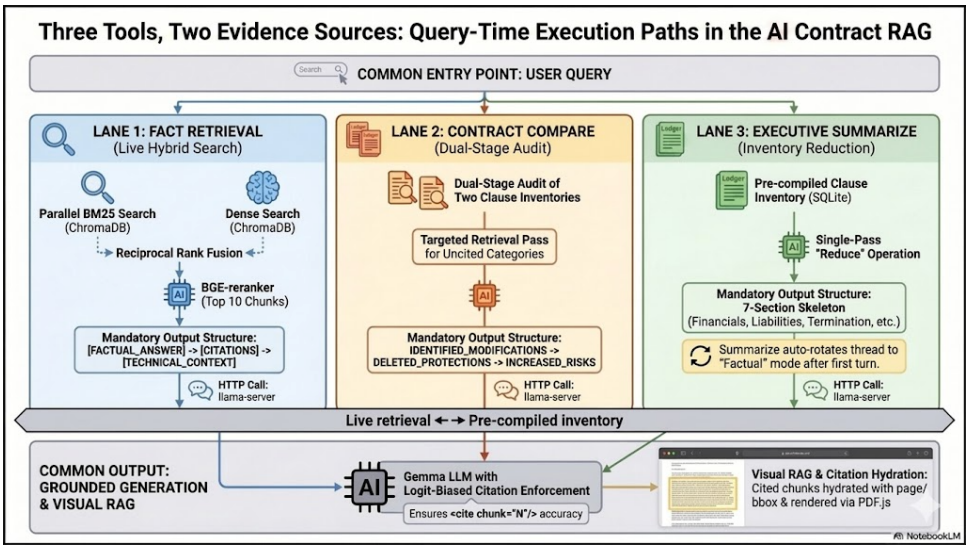

Figure 3: Hardware Information Table

Together, these architectural elements form a fully local, end-to-end pipeline that ingests unstructured maritime contracts, enforces hierarchy-aware retrieval, and exposes its reasoning through verifiable, source-anchored outputs. The following section evaluates how this pipeline performed across the corpus of voyage agreements and discusses the analytical insights it produced.

4. Results and Discussion

All three tools were tested throughout the duration of the project. Our results are included in the next section. It is important to note that the sponsor company took particular interest in evaluating the performance of the Contract Compare tool and, as such, the results section focuses heavily on this portion of the project.

4.1 System Performance and Analytical Outputs

The completely localized RAG pipeline successfully transformed the sponsor company's unstructured maritime voyage agreements PDF files into a structured, highly queryable data asset without relying on cloud-based compute. Operating natively on a single Minisforum MS-S1 Max workstation, the system demonstrated reliable performance across its core objectives.

The ingestion pipeline reliably processes voyage agreement PDF files, accurately identifying and stripping out strikethrough redlines so that deleted text does not corrupt the knowledge base. Through the parallel LLM map-reduce operation, the system successfully distills each contract pair into a highly structured SQLite "clause inventory," typically generating 80 to 250 distinct lines of extracted obligations, financial thresholds, and liability caps per agreement.

In operational testing, the Fact Retrieval tool successfully resolved complex legal hierarchies, prioritizing overriding Recap values over base CP values when conflicts arose. The cross-encoder reranking mechanism ensured that the top 10 most relevant chunks were retrieved and processed well under the strict 30-second timeout threshold.

4.2 Sponsor Validation and Stress Testing of the Contract Compare Tool

To evaluate the Contract Compare tool, a maritime trader with in-depth operations experience from the sponsor company conducted a manual analysis of a baseline and proposed contract. The employee then compared the results of his manual analysis to the output of the Contract Compare tool and concluded that the Contract Compare tool successfully captured many material exposure points. The employee awarded the Contract Compare report an overall performance score of 90% (13.5/15) across three key dimensions:

- Clause-by-Clause Analysis (5/5): The system demonstrated highly literal and exceptionally strong capabilities in isolating individual clauses.
- Highlighting Differences (5/5): The tool consistently and accurately located and presented the structural differences between the contracts.
- Commercial Reasoning (3.5/5): While the model's high literalism occasionally caused it to miss standard commercial workarounds, it demonstrated a strong, emerging capability to flag implicit risks arising from non-explicit wording. For instance, the tool successfully identified the absence of explicit termination rights regarding Russian-origin cargo warranties as a distinct commercial exposure point.

The sponsor confirmed that the tool accurately navigated the complexities of the dissimilar charter parties, successfully isolating and structuring key commercial exposures across eight critical domains:

- Demurrage & Laytime: Accurately contrasting explicit voyage laytime definitions against time charter structures that required reliance on overriding recap wording to intervene.
- Operating Speed: Identifying the shift from contractually fixed voyage speeds to time charter speed and consumption tables with performance reporting obligations.

- Payment Terms: Highlighting the difference between lump-sum or voyage-based pricing and daily hire rates.
- Trading Limits & Safe Berthing: Noting the transition from clearly defined safe berthing parameters to broad, less restrictive trading ranges that necessitate contradictory recap protections.
- Tank Preparation: Flagging potential exposure stemming from the lack of clarity in longer-duration time charters compared to specific voyage readiness outlines.
- Claims Handling & Agency: Identifying shifts in standard claim provisions and the transfer of agency appointment rights from the Charterer to the Owner.

Similarly, the Executive Summarize tool successfully generated standardized seven-section executive briefs directly from the pre-compiled clause inventory, accurately flagging outsized liability exposures.

Finally, the system successfully enforced its strict citation protocol. The generative model consistently anchored its factual claims with in-text citations, and the UI successfully translated these into clickable elements that opened an air-gapped PDF.js viewer, accurately highlighting the exact bounding box of the source text.

4.3 Key Learnings

Developing this pipeline yielded several critical technical and operational insights regarding the application of Generative AI in highly sensitive domains.

A prevailing assumption in enterprise AI is that advanced generative reasoning requires large-scale, expensive cloud infrastructure or data-center-grade discrete GPUs. This project demonstrates that such assumptions are overstated. By combining advanced quantization techniques with highly optimized, open-source LLM inference libraries, it is feasible to deploy a 26-billion-parameter Mixture-of-Experts (MoE) language model alongside an 8-billion-parameter embedding model on commodity hardware. By statically allocating a 96 GB unified-memory VRAM partition, the system effectively emulates enterprise-grade compute performance while maintaining strict data-residency and counterparty-confidentiality requirements, all without sacrificing inference speed or analytical depth.

In legal contract analysis, standard text extraction is insufficient because negotiated agreements frequently contain obsolete, strikethrough clauses that remain visually present in the final PDF file. If extracted blindly, these deleted terms would be embedded into the vector database, systematically biasing the LLM toward obsolete facts. A key learning was that deep PDF file inspection, specifically detecting strikethrough text (where vector drawing lines intersect character midlines) is a mandatory requirement for accurate document ingestion.

Initially, executing a comprehensive contract comparison relying solely on semantic chunk retrieval proved difficult due to context window limits and "needle-in-a-haystack" retrieval failures. A major breakthrough was decoupling the data structure from the text itself. By forcing the LLM to execute a map-reduce distillation during the initial ingestion phase, the system creates a structured "clause inventory" database. Bypassing standard vector retrieval and running the Contract Compare and Executive Summarize tools directly against this pre-compiled inventory proved significantly faster and provided vastly superior audit coverage.

Even when the AI produces highly accurate insights, commercial traders and legal analysts exhibit a high barrier to trust. The realization that an AI-generated answer is useless if it cannot be instantly verified led to the development of the Visual RAG PDF Viewer. We learned that providing raw citation text is insufficient; the system must physically bridge the gap back to the original document. Overlaying a translucent bounding box directly onto the cited paragraph within a native PDF viewer was the decisive factor in transforming the tool from experimental to a trusted, auditable operational asset.

5. Limitations and Mitigation Strategies

While the completely localized RAG pipeline successfully operationalizes the analysis of unstructured maritime voyage agreements, several constraints shaped its development. These limitations were carefully mitigated through rigorous architectural design and present clear pathways for future scalability.

The foundational limitation of this methodology is the hardware. The reliance on a single Minisforum MS-S1 Max mini-workstation, utilizing a 96 GB VRAM partition, fundamentally limits the maximum size and precision of the generative models that can be deployed. Consequently, the system relies on a quantized, open-source model (Gemma-4 26B) rather than a scalable, unconstrained proprietary frontier model (such as Claude Opus 4.7). While the quantized open-source model is highly efficient, aggressive quantization inevitably sacrifices some reasoning capability. To mitigate this quality gap, the system employs highly constrained prompt engineering and robust context routing. For instance, the compare tool uses a dual-stage setup with task injection to forcefully guide the model's output and prevent it from drifting off-task. As the sponsor company advances implementation of this system, a transition toward local server-rack deployments that leverage either large unified-memory architectures or discrete data-center-grade GPUs could further extend the pipeline's capabilities. Such infrastructure would support the deployment of substantially larger local language models, on the order of 70-billion to 400-billion parameters, at higher numerical precision. This evolution would materially narrow the reasoning and comprehension gap between open-source models and frontier proprietary, cloud-hosted systems, such as Claude Opus 4.7, while preserving on-premises execution and stringent data-confidentiality requirements.

This localized hardware footprint also introduces context window limitations, creating potential "needle-in-a-haystack" challenges during deep document retrieval. The current architecture actively manages a 32,768-token capacity limit using structured budget drop and shrink logic to prevent memory overflow. When searching for highly specific, deeply buried financial thresholds across dozens of pages, standard chunk retrieval can sometimes fail to retrieve the most relevant data. The system mitigates this limitation by utilizing an inventory-augmentation layer and an ONNX-accelerated cross-encoder, which rigorously re-ranks the semantic search results to guarantee the highest-quality evidence is prioritized within the token budget. Future scalability with context window capacities would allow the system to safely open the context window to 128,000 tokens or more, enabling the model to cross-reference multiple full-length voyage agreement PDF files simultaneously without relying as heavily on chunk-based retrieval filtering.

Additionally, the initial development and testing of this methodology were based on a focused foundational dataset. The scope of this project included Charter Party agreements and Recaps, but Addendums and Riders were not part of the available data and were therefore excluded. While this initial corpus was fully sufficient for proof-of-concept validation, working with a specific set of historical agreements naturally restricts the observation of highly heterogenous edge cases, particularly regarding complex document hierarchies. To mitigate the risk of the model failing to accurately resolve these hierarchies at scale, the system rigidly enforces precedence rules across three distinct, interlocking layers: explicit system prompts, targeted retrieval ordering tie-breaks, and controlled context assembly. Furthermore, because the ingestion pipeline autonomously distills all agreements into a structured SQLite database, the architecture is functionally decoupled from the initial data volume, rendering it ready to seamlessly scale horizontally to fleet-wide analytics once a broader dataset is introduced.

Finally, a limitation exists in the project's manual validation scope. Due to time and domain-expertise constraints, the authors did not conduct exhaustive, line-by-line manual legal reviews of every single contract pair to establish a perfect, human-graded baseline for evaluation. Instead, accuracy verification was primarily limited to evaluating the system's generated citations. To mitigate the lack of a comprehensive manual baseline, the methodology leans heavily on the Visual RAG PDF Viewer and its strict, mathematically enforced citation protocol. By programmatically linking every factual claim to an interactive bounding box overlaid on the original PDF file, analysts can instantly and visually verify the

source text in real-time. This limitation was also mitigated by working directly with the sponsor company's contracting experts to validate the quality of the project's output as compare to manual review.

6. Recommendations

The sponsor company possesses decades of valuable commercial intelligence embedded within dense legal PDF documents. Because this information is not natively searchable or analyzable, operational teams expend significant time manually reviewing historical contracts to support routine decision-making. To address this inefficiency, we recommend that the sponsor company adopt this secure artificial intelligence engine as a standard component of its demurrage contract review process. The system converts dormant legal PDF files (specifically maritime shipping agreements) into a structured, queryable intelligence repository, enabling teams to interrogate historical contracting data using natural-language queries. Importantly, this capability is delivered entirely within the firm's local computing environment, ensuring that no proprietary or sensitive data ever leaves company-controlled hardware.

We recommend starting with a structured pilot program. While fully manual reviews are inefficient, keeping a "human in the loop" remains essential during this transition phase. To verify the system's reliability, analysts should conduct their standard manual reviews alongside the AI tool. The pilot should test a diverse mix of voyage types, counterparties, and document formats, measuring success based on extraction accuracy, proper contract hierarchy, and precise citations.

While the initial scope of this project was intentionally limited to supporting demurrage analytics, claims coordination, and marine operational workflows, its potential applications extend well beyond these domains. As the capstone project matured, it became evident that, if scaled and broadly deployed, such a tool could serve as an active, day-to-day decision-support capability for commercial trading and contract negotiation. By converting static contract documents into a structured intelligence asset (rather than merely a structured database) the most significant value unlock emerges in the analysis of commercial voyage economics.

This capability enables a strategic shift away from reactive management of financial penalty disputes arising from suboptimal contractual outcomes, toward proactive contractual risk audits conducted before a vessel ever commences a voyage. Traders and commercial operators can evaluate proposed agreements against approved baseline templates prior to execution, entering negotiations informed by empirically grounded historical evidence rather than relying primarily on institutional memory. Ultimately, adoption of this workflow has the potential to conserve substantial capital, reduce operational friction, and systematically protect profit margins by identifying hidden contractual risks early. In doing so, the system transforms decades of previously inactive data into a compounding and defensible competitive advantage.

6.1 Future Work and Strategic Extensions

While the current tool successfully renders powerful insights securely within the local UI, it establishes a foundation for several high-value, follow-on or adjacent projects. Because the core asset is the engine itself, the exact same architecture can be leveraged to optimize future contract negotiations far beyond demurrage. To extend this project's impact, we recommend pursuing the following strategic extensions:

Currently, the system requires users to pull contracts ad hoc and push them through the ingestion pipeline individually. Future work could include creating a centralized, digital repository of all historical and active contracts within the sponsor company. The AI engine could then plug directly into this centralized database, automatically ingesting new deals as they are signed and keeping the data asset continually up to date without manual uploading.

To further enhance this ingestion pipeline, future iterations should incorporate an Optical Character Recognition (OCR) module. Adding robust OCR capabilities will allow the tool to seamlessly process

scanned voyage agreements, legacy physical contracts, and non-searchable PDFs, ensuring that older or alternatively formatted data is not left siloed.

On the processing side, a strategic upgrade would involve adding a LLM fine-tuned specifically for legal and maritime contracts. A specialized model of this kind would need to be custom-built so that it would inherently understand the nuanced phrasing of maritime law and could review the Gemma4 LLM's work as a judge. This would benefit the sponsor company by increasing extraction accuracy and reducing the need for extensive prompt engineering when handling highly complex terms and conditions.

Similarly, the system's contextual understanding can be deepened by systematically teaching the LLM the unique acronyms and internal shorthand utilized in the sponsor company's specific voyage agreements. Developing this domain-specific lexicon within the model's architecture will reduce parsing errors and ensure highly accurate interpretations of company-specific contract language.

Additionally, the tool's foundation can easily be expanded to track other critical contract conditions. For example, future versions could be adapted to help manage areas like war-risk compliance, tariff responses, fleet liability, or payment terms across the active fleet. This was already proven by the sponsor company's review (previously discussed in results).

A highly impactful next step would be configuring the system to produce a structured, machine-readable record alongside each Contract Compare and Executive Summarize report. This acts as a bridge from in-app insights to workflow-integrated agentic intelligence, allowing downstream enterprise tools to consume the findings directly.

Utilizing the machine-readable feed, the company can implement automated alerting infrastructure. By extracting payment deadlines, laytime expiration dates, and overall risk verdicts, the tool can be wired directly into enterprise email, calendar, or approval workflows. For example, every proposed agreement could automatically trigger a compare audit, attaching a one-page risk summary directly to a manager's pre-signing approval request.

To maximize strategic value, terms extracted from all historical contracts should be pulled into a centralized database and compared against actual voyage and demurrage outcomes. By conducting a comparative analysis between the negotiated terms and the final financial realities of the voyages, the company can quantitatively identify hidden risk vectors and adjust future contracting strategies to proactively mitigate losses.

Ultimately, this structured data asset can be joined with the sponsor's existing voyage-economics and itinerary-tracking systems. This integration would allow the company to price contractual risk in actual dollar terms, calculating the exact expected cost of a counterparty altering laytime based on the current vessel's real-time itinerary and port congestion data.

References

AltexSoft. (n.d.). The good and bad of ChromaDB for RAG. https://www.altexsoft.com/blog/chroma-pros-and-cons/

Applied Sciences. (2026). From dataset to model: Cross-lingual embeddings for text and tabular retrieval. MDPI Applied Sciences, 15(22). https://www.mdpi.com/2076-3417/15/22/12219

Blue-C. (n.d.). Making voyage decisions count: Real-time commercial visibility is vital to deliver value with optimization. https://blue-c.no/making-voyage-decisions-count-why-real-time-commercial-visibility-is-vital-to-deliver-value-with-optimization/

Cloudflare. (2026, April 4). Google Gemma 4 26B A4B now available on Workers AI. https://developers.cloudflare.com/changelog/post/2026-04-04-gemma-4-26b-a4b-workers-ai/

Cormack, G. V., Clarke, C. L. A., & Buettcher, S. (2009). Reciprocal rank fusion outperforms Condorcet and individual rank learning methods. In Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 758–759). ACM. https://dl.acm.org/doi/10.1145/1571941.1572114

Dettmers, T., Lewis, M., Shokri, R., Zettlemoyer, L., & Shen, S. (2022). The case for 4-bit precision: k-bit inference scaling laws. arXiv. https://arxiv.org/abs/2212.09720

Elastic. (n.d.-a). Lexical vs. semantic search. https://www.elastic.co/blog/lexical-vs-semantic-search

Elastic. (n.d.-b). BM25: The next generation of Lucene — BM25 scoring. https://www.elastic.co/blog/found-understanding-scoring-in-lucene-part-2

GeeksforGeeks. (n.d.). Introduction to ChromaDB. https://www.geeksforgeeks.org/nlp/introduction-to-chromadb/

Gerganov, G. (n.d.). llama.cpp [Software]. GitHub. https://github.com/ggerganov/llama.cpp

Khronos Group. (n.d.). Vulkan API overview. https://www.khronos.org/vulkan/

LangChain. (n.d.). Map-reduce document processing. LangChain Concepts. https://python.langchain.com/docs/concepts/#map-reduce

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. arXiv. https://arxiv.org/abs/2005.11401

Linux Kernel Documentation. (n.d.). Memory management in Linux. Kernel.org. https://www.kernel.org/doc/html/latest/admin-guide/mm/index.html

Microsoft. (n.d.). ONNX Runtime. https://onnxruntime.ai/

Mozilla Foundation. (n.d.). PDF.js. GitHub. https://github.com/mozilla/pdf.js

Omdena. (n.d.). Document parsing for RAG: Handling multi-column documents. https://www.omdena.com/blog/document-parsing-for-rag

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. arXiv. https://arxiv.org/abs/1908.10084

Saravanan, S. (n.d.). Unlocking PDF data the smart way: A practical guide to pdfplumber. Medium. https://sangeethasaravanan.medium.com/unlocking-pdf-data-the-smart-way-a-practical-guide-to-pdfplumber-3c80b5d9d491

ServeTheHome. (n.d.). Minisforum MS-S1 Max review: The best Ryzen AI Max mini-PC yet. https://www.servethehome.com/minisforum-ms-s1-max-review-the-best-ryzen-ai-max-mini-pc-yet/

SQLite. (n.d.). SQLite as an application file format. https://sqlite.org/appfileformat.html