sensors

MDPI

Article

Agentic RAG for Maritime AIoT: Natural Language Access to Structured Data †

Oxana Sachenkova [1,*], Melker Andreasson [1], Dongzhu Tan [1] and Alisa Lincke [2]

[1] Computer Science and Media Technology Department, Linnaeus University, SE-391 31 Kalmar, Sweden
[2] Computer Science and Media Technology Department, Linnaeus University, SE-352 53 Växjö, Sweden; alisa.lincke@lnu.se
* Correspondence: oxana@vild.ly
† This paper is an extended version of our paper published in Andreasson, M.; Tan, D. Lighthouse Bot: A Platform For Evaluating LLMs For Agentic Maritime Data Analysis, 2025.

Abstract

Maritime operations are increasingly reliant on sensor data to drive efficiency and enhance decision-making. However, despite rapid advances in large language models, including expanded context windows and stronger generative capabilities, critical industrial settings still require secure, role-constrained access to enterprise data and explicit limitation of model context. Retrieval-Augmented Generation (RAG) remains essential to enforce data minimization, preserve privacy, support verifiability, and meet regulatory obligations by retrieving only permissioned, provenance-tracked slices of information at query time. However, current RAG solutions lack robust validation protocols for numerical accuracy for high-stakes industrial applications. This paper introduces Lighthouse Bot, a novel Agentic RAG system specifically designed to provide natural-language access to complex maritime sensor data, including time-series and relational sensor data. The system addresses a critical need for verifiable autonomous data analysis within the Artificial Intelligence of Things (AIoT) domain, which we explore through a case study on optimizing ferry operations. We present a detailed architecture that integrates a Large Language Model with a specialized database and coding agents to transform natural language into executable tasks, enabling core AIoT capabilities such as generating Python code for time-series analysis, executing complex SQL queries on relational sensor databases, and automating workflows, while keeping sensitive data outside the prompt and ensuring auditable, policy-aligned tool use. To evaluate performance, we designed a test suite of 24 questions with ground-truth answers, categorized by query complexity (simple, moderate, complex) and data interaction type (retrieval, aggregation, analysis). Our results show robust, controlled data access with high factual fidelity: the proprietary Claude 3.7 achieved close to 90% overall factual correctness, while the open-source Qwen 72B achieved 66% overall and 99% on simple retrieval and aggregation queries. These findings underscore the need for a secure limited-context RAG in maritime AIoT and the potential for cost-effective automation of routine exploratory analyses.

Keywords: IoT; GenAI; LLMs; RAG; sensor data; maritime industry

Check for updates

Academic Editors: Emanuele Lattanzi and Chiara Contoli

Received: 24 November 2025
Revised: 2 February 2026
Accepted: 2 February 2026
Published: 13 February 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

The maritime industry faces unprecedented pressure to enhance operational efficiency, reduce environmental impact, and navigate complex regulations. While the sector generates vast amounts of data—from real-time sensor readings to operational logs—extracting

actionable insights remains a significant challenge due to fragmentation, the need for specialized knowledge, and difficulty with real-time analysis [1]. As a result, the industry is working to overcome critical issues like inefficient predictive maintenance, fuel optimization, and the risk of "out-of-the-loop familiarity," where seafarers become too reliant on automated systems, as detailed in recent studies from Det Norske Veritas (DNV) expert organisation [2]. These challenges are exacerbated by industry fragmentation, reliance on outdated methods, and increasing environmental demands [3].

Recent advances in Artificial Intelligence (AI), particularly Large Language Models (LLMs) built upon the Transformer architecture [4], offer promising avenues for addressing these challenges. LLMs excel in natural language understanding and generation, providing contextually relevant and coherent outputs [5]. However, working with sensor-based information and structured datasets using LLMs requires robust frameworks that can seamlessly bridge natural language queries into executable tasks on complex data. Existing RAG solutions often lack rigorous validation protocols for numerical accuracy—a fundamental requirement for industrial reliability where precise sensor data analysis directly impacts operational safety and efficiency.

In this paper, we present an approach that utilizes Agentic Retrieval Augmented Generation (ARAG) to democratize access to and analysis of maritime data. Our core contribution lies in the development of an ARAG system that empowers maritime stakeholders to query complex datasets using natural language. By integrating an LLM with specialized database, file system and coding generation agents, our system translates natural language queries into structured database queries and Python code, enabling rapid identification of inefficiencies and exploration of potential improvement scenarios.

We demonstrate our system's efficacy by optimizing ferry operations at Färjerederiet, a Swedish state-owned public ferry operator. Committed to achieving climate neutrality by 2045, Färjerederiet, whose operational environment reflects challenges shared across the broader maritime sector, currently relies on time-consuming manual data analysis. Our research addresses this challenge, exploring how LLM-based systems can automate and accelerate data-driven decisions to meet stringent industrial and environmental goals. This approach simplifies access to structured data and insights, helping the operator minimize emissions and maintain service efficiency on its five diverse ferry routes. Specific operational challenges shared by Färjerederiet included the need to: optimize ferry schedules by refining trip frequencies to balance peak and low demands; identify and implement effective emission reduction strategies such as minimizing excess trips and improving fuel efficiency; and rapidly enhance operational insights to uncover inefficiencies and generate actionable intelligence aligned with Sweden's national climate neutrality goals.

Beyond addressing these specific operational objectives, a key contribution of this work is the development of a robust, generalizable application designed for the rigorous evaluation and testing of such generative AI (GenAI)-enabled Internet Of Things (IoT) systems.

This application, built to handle real-world sensor data, provides a foundation not only for further maritime optimization but also serves as a framework for assessing the performance and reliability of ARAG systems in diverse IoT contexts. We highlight the system's architecture, the challenges of developing a robust test suite, and preliminary metrics showcasing its potential to enhance decision-making and provide a foundation for broader IoT applications.

This paper addresses the following research questions:

- RQ1 (System Enablement & Architecture): What architectural adaptations are required to bridge non-deterministic natural language queries with deterministic sensor data in an Agentic RAG system?

- RQ2 (Practical Impact & Case Study): What are the practical implications and demonstrable benefits of deploying such an AIoT system for optimizing operational efficiency and supporting sustainability goals, as exemplified by the Färjerederiet ferry operations case study?

The remainder of this paper is structured as follows. Section 2 provides background on large language models and their role in supporting data analysis for IoT systems, including an overview of ARAG and AI-driven data analysis in maritime applications. Section 3 describes the proposed methodology, detailing the system architecture, datasets, experimental setup, evaluation metrics, and test suite design. Section 4 presents the results, including quantitative factual correctness measures, qualitative and textual generation metrics, and an end-to-end test case scenario. Section 5 discusses the implications of the findings and their limitations, and Section 6 concludes the paper by summarizing the main contributions and outlining directions for future work.

2. Background

2.1. Large Language Models (LLMs) and Their Role in Supporting Data Analysis for IoT

LLMs offer substantial potential to transform raw data into actionable insights using natural language queries. Earlier systems, often relying on models like GPT-3.5 [6], have already demonstrated success in providing simple, predefined insights and automating steps such as data standardization over text-like data [7,8]. However, the application of these models to heterogeneous sensor data, common in industrial AIoT environments , presents a key challenge because of its structured nature and sheer volume. Addressing this problem is central to RQ1 (System Enablement & Architecture), which explores the architectural adaptations necessary to make complex sensor data accessible to AI algorithms.

For industrial AIoT applications, there are three primary methodologies for adapting an LLM to specialized knowledge: Prompt Engineering (PE), Fine-Tuning (FT), and RAG [9]. PE is constrained by the LLM's context window, making it impossible to handle large data volumes. FT is impractical because it demands continual retraining and fails to enforce data minimization and verifiability policies for critical operations [10]. Therefore, we employ RAG [11], which integrates the generative power of LLMs with dynamic data retrieval. This approach is particularly useful for analyzing evolving datasets and ensuring up-to-date information, providing robust data access and processing capabilities to the text-generation model.

This technique is already being used in various analytical contexts. For example, in finance, RAG-enhanced systems are used for evidence recommendation in financial legal resolutions, grounding recommendations in analogous past cases [12]. Similarly, in the legal domain, RAG methods improve applications that range from legal overviews to optimizing virtual legal assistants for dispute resolution [13,14]. In healthcare, RAG approaches have been shown to assist in patient triage, improving diagnostic accuracy [15]. Furthermore, RAG has been successfully applied in real-time health monitoring systems to integrate healthcare data, seamlessly processing complex medical history [16].

The architecture of the RAG system (Figure 1) involves indexing and aggregation of the input dataset. This includes transforming raw, diverse data, whether it is text documents, tabular records, or other digital assets, into organized, retrievable segments optimized for efficient search. This structured data is stored in a vector database, forming a foundational context for subsequent queries. When a user introduces a query, it undergoes query preprocessing to align with the data's structure. An embedding model then generates vector representations, enabling precise retrieval of relevant data. Techniques like hybrid search further refine this process by prioritizing the most relevant information. The retrieved contexts are then subjected to prompt optimization, guiding the LLM to generate contextually

accurate and informative output. Finally, the LLM produces a response, enriched with real-time data, ensuring contextual accuracy and relevance.

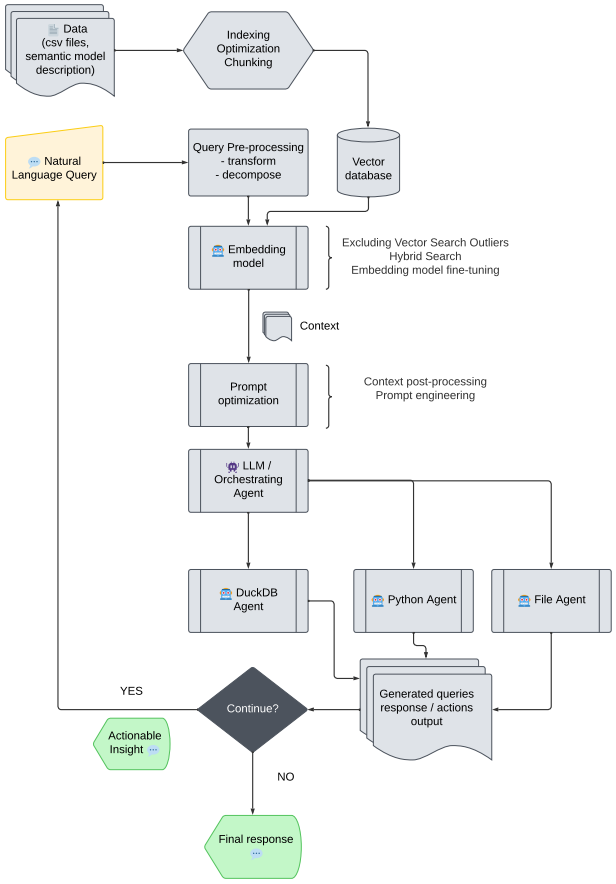


Figure 1. Proposed System Architecture for a Sensor Analysis Bot—Lighthouse Bot.

This ability to ground LLM responses in traceable, real-time data is essential for our maritime AIoT system, directly contributing to the practical benefits explored in RQ2. The utility of RAG in analyzing sensor data for real-time monitoring and control is further evidenced by recent work. For instance, RAG-LLM frameworks have been integrated with deep Q-networks for intelligent fish control systems, demonstrating superior growth rates and rapid stabilization of automation policies by leveraging IoT data [17]. Similarly, a real-time environment monitoring and response system leverages IoT sensor data, preprocessing on the edge devices, and a RAG-enhanced LLM for fast query handling and reliable environmental insights, overcoming challenges like network delays and scalability for non-expert users [18].

Despite these successes, standard RAG architectures face significant limitations when applied to the structured, numerical nature of AIoT data. It is crucial to distinguish between text-based RAG, which operates on unstructured documents, and numeric/tabular RAG, which must handle rigid time-series and relational schemas. While semantic vector similarity excels at retrieving relevant text chunks, it often fails to perform accurate aggregations or temporal reasoning over numerical data (e.g., "calculate the average fuel consumption during peak hours"). This is compounded by specific tokenization inefficiencies, where fragmented numerical values compromise arithmetic precision, and a heightened risk of digit hallucination, where models generate plausible but incorrect numbers that defy semantic detection [19]. These specific failure modes create a critical reliability gap for industrial operators, necessitating an agentic approach that leverages executable tools—such as SQL and Python—rather than relying solely on probabilistic text generation.

2.2. Agentic RAG

Agentic Retrieval-Augmented Generation (ARAG) extends the concept of RAG by equipping autonomous AI agents with the ability not only to retrieve and synthesize information but also to make informed decisions based on this data (Figure 1). These agents can perform a variety of tasks such as generating code, executing SQL queries and automating workflows.

Unlike traditional RAG, ARAG facilitates dynamic adaptation to new information, allowing the system to provide proactive insights and actionable recommendations based on real-time data.

An ARAG system can even include running verification steps or an automated test suite to further verify the generated output. This iterative process enables continuous refinement and informed decision-making, thereby enhancing the overall efficacy and adaptability of the AI system within the dynamic IoT environment.

This concept is fundamental to our research, directly informing the architecture detailed in RQ1 and enabling the implementation of RQ2. As a relatively new and rapidly evolving approach, there is growing interest in its potential application to real-time data analysis. For example, recent work has explored ARAG for industrial anomaly detection, effectively identifying anomalies in real-time pipeline sensor data with high precision and reduced training effort compared to traditional models [20]. Similarly, an Agentic Search Engine for Real-Time IoT Data has shown promise in achieving high accuracy in retrieving intent-based services and providing concise, context-aware responses, outperforming generalized systems by making real-time IoT data accessible for effective decision-making [21]. Although published peer-reviewed work on ARAG remains limited, these early applications demonstrate its significant potential in dynamic IoT environments. Current ARAG research also demonstrates a clear gap: existing frameworks lack robust validation protocols specifically designed for numerical accuracy in high-stakes industrial environments. While ARAG systems demonstrate promise in various domains, they typically rely on traditional NLP evaluation metrics (e.g., semantic similarity, faithfulness) that are inadequate for verifying precise numerical outputs from sensor data analysis—a fundamental requirement for industrial reliability where incorrect calculations can lead to operational failures, safety risks, or significant financial losses.

2.3. AI-Driven Data Analysis in Maritime Applications

Just as in other industries, the maritime domain is increasingly interested in LLMs for a wide range of data analysis applications. This is a rapidly emerging area of research, with work from various research groups exploring the use of LLMs to predict ship detention in Port State Control inspections [22], to enhance automation and decision-making in maritime operations more broadly [23], and to improve safety in maritime operations through a variety of applications [24].

Based on this growing interest, our work directly addresses key challenges in the application of LLMs to the maritime sector. Our research group with collaborators conducted a pre-study [25] exploring the use of LLM for decision support in maritime navigation. Although this work found that LLMs could potentially address the shortcomings of traditional algorithms, it also exposed critical limitations, such as the models' poor spatial reasoning and their tendency to produce authoritative but incorrect statements. This highlighted a crucial need for robust verification and validation frameworks, particularly when working with specialized maritime data.

Building upon this growing interest, our research directly addresses key challenges in applying LLMs to the maritime sector. Our group's initial pilot study [26,27] explored the application of a RAG approach to real-world maritime data. This work demonstrated

immense potential for tasks like ferry schedule optimization, but also exposed potential issues with this approach. These included the need for resilient self-hosted applications for secure data handling, robust mechanisms to mitigate LLM inaccuracies, such as "hallucinations," and rigorous testing to ensure reliability.

This current work addresses these limitations directly by proposing a system architecture with full agentic capabilities. It is specifically designed to handle real-time, structured, and sensor data, incorporating a more robust test suite, verified metrics, and a coding agent for a comprehensive and reliable evaluation of the system's performance. The framework we propose is relevant not only to the maritime industry but also to other IoT and industrial applications as a whole.

Consequently, this paper addresses three critical research needs identified in the literature:

1. From text to structured reasoning: Moving beyond semantic retrieval of documents to verifiable execution of SQL and Python code for time-series analysis.
2. Verifiability in industrial setting: Overcoming the "black box" nature of standard LLMs by implementing an agentic architecture where every analytical step is executed via auditable code.
3. Democratization of maritime data: Providing a natural language interface that allows non-technical stakeholders to access complex sensor silos without knowledge of SQL, directly supporting operational optimization goals (e.g., fuel efficiency).

3. Methods

The present study employs a design science research (DSR) [28] methodology to design, implement, and evaluate an Agentic AI system capable of providing natural language access to complex, heterogeneous maritime data. This approach was selected for its suitability in developing and rigorously assessing innovative, technology-based artifacts working with real-world data. To address RQ1 (System Enablement & Architecture), the study investigates the architectural adaptations required to bridge flexible natural language queries with deterministic sensor data, exemplified through the Lighthouse Bot system architecture that combines multiple specialized models to facilitate automated contextual understanding and reasoning over maritime tasks and AIoT data. In response to RQ2 (Practial Impact & Case Study), a controlled experimental evaluation was conducted using a structured test suite of 24 operationally relevant queries derived from the Färjerederiet ferry operations use case. This approach enabled a structured validation of the system's design and an evaluation of its real-world benefits for operational efficiency and environmental performance in maritime operations.

3.1. System Architecture

Building on the initial insights from the Lighthouse pilot study [26,27], we developed a comprehensive architecture of the ARAG system (shown in Figure 1) designed to provide robust, accurate, and explainable natural language access to complex maritime sensor data.

At its foundation, the system is designed to ingest heterogeneous maritime AIoT data from multiple sources. The architecture supports two primary data ingestion paths: (1) historical batch data from unstructured and semi-structured files (e.g., CSV files containing ferry trip records) and historical data retrieved via REST API endpoints (e.g., historical vessel operational data from PONTOS-HUB), and (2) real-time streaming data accessed via MQTT APIs (e.g., near real-time sensor readings from PONTOS-HUB). For the evaluation presented in this paper, we utilized pre-processed historical batch data that had been previously integrated and loaded into the database. This raw data is then processed and loaded into a robust structured database environment (e.g., DuckDB or PgVector) for

storage, efficient querying, and vector indexing. The data is preprocessed by indexing and optimization, followed by segmenting for efficient retrieval. A dedicated embedding model (text-embedding-3-small from OpenAI transforms textual and metadata content into vector representations stored in a vector database, enabling both hybrid search and outlier exclusion to enhance retrieval precision. In the current implementation, we utilize a pre-trained embedding model without fine-tuning; however, the modular architecture supports replacing this with a domain-specific fine-tuned embedding model should improved retrieval performance for maritime-specific terminology and concepts be required in future deployments.

When a user issues a natural language query, a pre-processing module decomposes and transforms the query before passing it to the orchestrating LLM agent. The orchestrator is implemented as a single unified agent (using the agno framework) equipped with multiple specialized tools: DuckDB tools for structured SQL retrieval and aggregation, Python tools for time-series computation and statistical analysis, Pandas tools for data manipulation, and File tools for accessing file-based records. The agent autonomously selects which tool(s) to invoke using a prompt-based routing mechanism (configured with `tool_choice=‘‘auto’’`), rather than relying on a separate classification layer or router model. The tool selection logic is explicitly defined through structured guidelines in the system message: DuckDB tools are recommended for SQL queries, aggregations, and relational data operations; Python tools are recommended for complex calculations, statistical analysis, data transformations, and custom logic; and File tools are recommended for reading/writing files and data export. The instructions further specify a combination strategy: start with SQL queries to obtain the right data subset, then use Python for complex analysis or transformations. This prompt-based routing approach ensures reproducibility, as the decision logic is fully encoded in the system message and instructions. Although a formal prompt engineering evaluation was outside the scope of this study, the iterative refinement of this design was important to achieve the high factual correctness reported in our evaluation. The extracted or computed results are then returned to the LLM for context integration, post-processing, and prompt optimization, resulting in a coherent natural language response.

The transformation of a natural language query into an executable command is facilitated by a carefully designed prompt, which guides the LLM to act as a reasoning agent. This prompt, a key component of our system's design, is available for review on our GitHub repository at [29]. Our prompt adheres to best practices in Tool-Augmented Reasoning [30,31], instructing the LLM to first adopt the persona of a "senior data analyst" and then follow a precise, multi-step Chain-of-Thought (CoT) process. The core strategy is encoded into explicit instructions, requiring the agent to: (1) Think First before generating code; (2) utilize the provided Semantic Model to identify necessary data tables and relationships; and (3) strategically select between SQL (for filtering/aggregation) and Python (for complex analysis/statistical tasks). This structured guidance is essential for ensuring the reliable and verifiable execution of analytical tasks on heterogeneous sensor data. Code execution safety is ensured through multiple mechanisms: the agent is configured with a `tool_call_limit=20` to prevent excessive tool invocations, and the system enforces request-level timeouts (120 s via Gunicorn) to prevent infinite loops or runaway processes. All generated Python code is executed within the agno framework's controlled execution environment, which provides isolation and exception handling. Although a formal prompt engineering evaluation was outside the scope of this study, the iterative refinement of this design was important to achieve the high factual correctness reported in our evaluation.

This proposed multi-agent, modular design enables the system to integrate retrieval, reasoning, and execution in a unified workflow, allowing it to dynamically generate exe-

cutable code, perform complex database operations, and deliver actionable insights across multiple maritime data modalities. The architecture enables the generation of Python code for time-series analysis of diverse sensor data streams, the execution of complex SQL queries on structured sensor databases to extract operational insights, and the analysis of datasets related to key performance indicators, such as fuel consumption and emissions. Furthermore, the system can automate workflows for route optimization and scheduling based on dynamic maritime conditions and historical trends, while offering an intuitive natural language interface that enables these analytical and operational tasks to be performed without technical or programming expertise.

3.2. Data

Our project utilized two primary datasets for analyzing maritime operations.

The Ferry Trips Dataset [32], published by the Swedish Transport Administration contains operational records from five ferries (Fragancia, Jupiter, Merkurius, Nina, and Yxlan) from 1 March 2023 to 29 February 2024. This dataset includes detailed trip information, departure times, vehicle statistics, trip types, and passenger car equivalents (PCE). Derived fields include travel distance and trip timings.

The PONTUS project [33] provides historical and near real-time ship operational data via the PONTOS-HUB, an open-access platform. Data, contributed by partners, is stored in an SQL database and accessed through REST API for historical data and MQTT API for near real-time data. This data includes extensive metrics beyond AIS, such as engine fuel consumption, battery state, and environmental conditions, all governed by the Open Database License (ODbL) v1.0. Data from the same period of time as for the Ferry Trips Dataset has been extracted to complement it with fuel consumption and environmental conditions.

3.3. Experimental Setup

The evaluation was conducted as a controlled experiment using a custom-designed test suite consisting of 24 natural language queries, each paired with a verified ground-truth answer. These test cases, developed from the Färjerederiet pilot project, cover the most common analytical operations in maritime AIoT contexts—namely data retrieval, aggregation, and analytical reasoning—and are categorized by complexity as simple, moderate, or complex. The experiment compared the performance of one leading proprietary large language model (Claude 3.7 Sonnet) against two prominent open-source models (e.g., Llama 3 70B and Qwen 72B), all integrated within the same ARAG architecture using identical tools and prompt designs. These three models—Claude 3.7 Sonnet, Llama 3.3-70b-instruct, and Qwen 72B-instruct were selected to represent a balanced comparison between state-of-the-art proprietary and high-performance open-source LLMs. This selection enables a comprehensive assessment of performance, accuracy and adaptability across different deployment scenarios, reflecting both commercial-grade reliability and the growing potential of open-source alternatives in real-world maritime analytics. All core experiments and the ARAG system architecture were deployed within a Docker environment based on the python:3.11-slim image, running on a host machine equipped with an Apple M4 Pro chip (Apple Inc., Cupertino, CA, USA) (featuring an integrated Apple GPU and 24 GB of Unified Memory). The development, data ingestion pipeline construction, and model verification (including the validation of the test suite ground truths) were performed using Python 3.11 within Jupyter Lab 4.3.6. The primary data manipulation and analytical libraries used were pandas 2.2.3, NumPy 1.26.4, and seaborn 0.13.2. The core RAG and evaluation frameworks employed included LangChain, agno, and ragas. All

dependencies and exact library versions utilized in the study are detailed in the project's GitHub repository [29].

3.4. Evaluation Metrics

To systematically assess the performance of the models, Lighthouse Bot utilizes the RAGAS (Retrieval Augmented Generation Evaluation) framework 0.2.13 [34], complemented by a custom-implemented metric for numerical precision. RAGAS offers a comprehensive suite of tools and metrics for assessing RAG system performance, primarily using an LLM as an automated judge. For our evaluation, OpenAI was used for embeddings, and DeepSeek-V3-0324 served as the LLM evaluator for RAGAS metrics.

The following metrics were employed in our evaluation, each with explicit computational formulas and defined value ranges:

Calculated Factual Correctness (Custom Metric): This custom metric addresses the critical limitation of standard LLM-based RAG evaluation frameworks, including the native Factual Correctness metric in RAGAS, which are unable to accurately and robustly assess numerical answers. These standard metrics, which rely on nuanced LLM-driven judgment of factual claims and semantic similarity, often fail to precisely verify a single numerical value, which is critical for our application. Our Calculated Factual Correctness metric works by programmatically extracting the single, most relevant numerical value from the LLM's response using regular expressions and string matching. This extracted value is then compared directly against a manually verified numerical ground truth. The computation follows the formula:

$$\text{Score} = 1 - \frac{|A - B|}{\max(|A|, |B|)} \tag{1}$$

where A represents the extracted numerical value from the model's response and B represents the ground truth value. The metric yields a range of [0.0, 1.0], where 1.0 indicates an exact match and 0.0 indicates complete disagreement. Edge cases are handled as follows: if both values are zero, the score is 1.0; if only one value is zero (either A or B), the score is 0.0.

Semantic Similarity (RAGAS Metric): This metric evaluates the semantic resemblance between the LLM's answer and the ground truth reference using embedding-based similarity. The computation uses cosine similarity between the embedding vectors:

$$\text{Semantic Similarity} = \cos(\theta) = \frac{A \cdot B}{||A|| \times ||B||} \tag{2}$$

where A and B are the embedding vectors of the response and ground truth, respectively, generated using OpenAI's text-embedding-3-small model. The metric yields a range of [0.0, 1.0], where 1.0 indicates perfect semantic alignment and 0.0 indicates no semantic similarity.

Faithfulness (RAGAS Metric): This metric measures the factual consistency of the LLM's answer with the retrieved contexts, assessing whether claims in the generated response are supported by the provided context (i.e., no hallucinations). The computation follows:

$$\text{Faithfulness} = \frac{\text{Number of supported claims}}{\text{Total number of claims}} \tag{3}$$

where claims are extracted from the response using an LLM evaluator (DeepSeek-V3-0324 in our implementation), and each claim is verified against the retrieved contexts to determine if it can be inferred from the provided context. The metric yields a range of [0.0, 1.0], where 1.0 indicates all claims are fully supported by the context and 0.0 indicates no claims are supported.

ROUGE Score (RAGAS Metric): This metric measures n-gram overlap between the generated response and the ground truth reference. We employ ROUGE-L F1, which is based on the Longest Common Subsequence (LCS) between candidate and reference. The computation follows:

$$\text{ROUGE-L F1} = \frac{2 \times \text{Precision}_L \times \text{Recall}_L}{\text{Precision}_L + \text{Recall}_L} \quad (4)$$

where $\text{Precision}_L = \frac{\text{LCS_length}}{\text{candidate_length}}$ and $\text{Recall}_L = \frac{\text{LCS_length}}{\text{reference_length}}$. The metric yields a range of [0.0, 1.0], where 1.0 indicates perfect n-gram overlap and 0.0 indicates no overlap.

While these traditional NLP metrics (Semantic Similarity, Faithfulness, and ROUGE Score) provide complementary insights and are commonly used in chatbot evaluations, they proved limited for our numerical-focused use case, as discussed in the Results section. The Calculated Factual Correctness metric remains the primary and most reliable measure for assessing numerical accuracy in analytical contexts.

3.5. Test Suite Design

For this study, we have designed 24 distinct test cases, each with a predefined natural language query, corresponding ground-truth answer, and metadata to categorize its complexity and domain. This structured approach allows for a reproducible and objective evaluation. The test cases are categorized by the following attributes:

- Query Complexity: Questions are categorized as Simple, Moderate, or Complex, based on the number of steps or data manipulations required to arrive at the answer. Simple queries might involve direct retrieval of a single value, while Complex queries require multiple data aggregations, calculations, or advanced analysis.
- Maritime Domain Category: Queries are grouped by their relevance to key operational areas, including Fuel Efficiency, Route Optimization, and Environmental Impact.
- Data Interaction Type: This categorizes the type of task the system must perform, such as simple Retrieval (e.g., fetching a specific value), Aggregation (e.g., calculating an average or sum), or Analysis (e.g., identifying anomalies or correlations).

Figure 2 illustrates the composition of the 24-query test suite, highlighting the diversity of query complexity (simple, moderate, and complex) and their distribution across key operational domains—route operations, fuel efficiency, and environmental impact.

Classification of each query by interaction type (retrieval, aggregation, comparison, and analysis) and associated data variation tags, demonstrating the comprehensive coverage of typical analytical tasks within maritime AIoT operations, is also represented in Figure 2.

This query assesses the system's advanced reasoning and analytical capabilities, where the ARAG must identify an outlier by processing a statistical analysis, going beyond simple retrieval or aggregation (see Appendix A.1 for examples).

Ground Truth Generation Methodology

To create a set of ground truths for our test cases, we developed a systematic methodology to identify and prioritize anomalous operational days and other analytical insights. This process was implemented in Python using the Pandas library, starting with the aggregation of raw trip data into daily summaries and the engineering of performance metrics (e.g., fuel per trip). By calculating the Z-score $Z = \frac{x-\mu}{\sigma}$ for these metrics, our script objectively flags the most statistically significant anomalies, producing a ranked list of candidate days or events. A Pearson correlation analysis is then used to provide initial context, helping to distinguish anomalies that correlate with operational demand (e.g., high passenger loads) from those that do not. The final output is a targeted list of

high-priority events, which forms the basis for our subsequent effort to establish ground truth by comparing these specific instances against known operational outcomes or further investigation. This systematic approach was applied consistently across all 24 test cases with the aim of identifying whether the Lighthouse Bot System would accurately mimic programmatic exploratory data analysis of the given data.

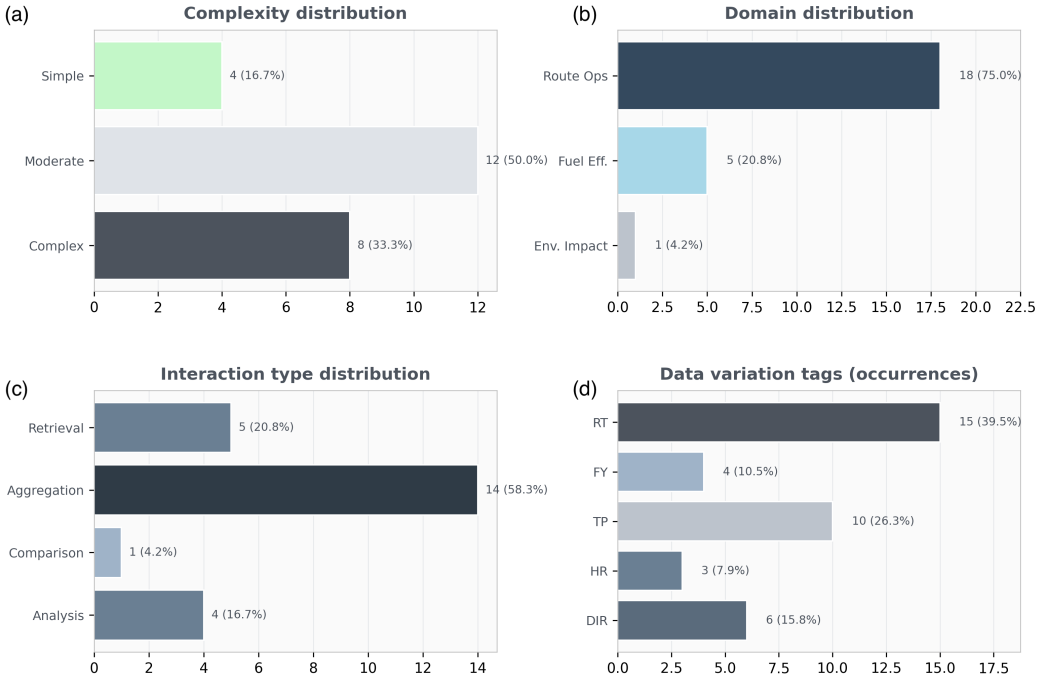


Figure 2. Test Suite Distribution: (**a**) Complexity distribution (Simple, Moderate, Complex); (**b**) Domain distribution (Route Optimization/Operations, Fuel Efficiency, Environmental Impact); (**c**) Interaction type distribution (Retrieval, Aggregation, Comparison); (**d**) Data variation tags showing the frequency of RT, FY, TP, HR, and DIR across test cases. The X-axis in panels (**a**–**c**) represents the number of test cases (count), with each segment corresponding to a specific category within the test suite.

By employing this structured and comprehensive test suite, we can not only quantitatively measure the system's performance but also qualitatively understand its strengths and weaknesses in handling diverse maritime data analysis tasks. The results of this evaluation will be discussed in the following section. The complete set of designed test suites is available in the complementary materials (Appendix A.2).

4. Results

4.1. Calculated Factual Correctness Metric

Figure 3 provides a visual overview of the performance of the three models on the Calculated Factual Correctness metric across all 24 test cases. The heatmap shows a clear performance ladder, with the proprietary model demonstrating superior overall accuracy compared to the open-source models. Claude 3.7 achieved an average Calculated Factual Correctness of 0.896 with only six test cases that had incorrect answers. The model overall has a robust ability to handle a wide range of queries, from simple data retrieval to complex analytical tasks. The open-source models, Llama and Qwen, showed more variability in their performance, as evidenced by the lower overall average score of 0.35 and 0.667.

A closer examination of the results, as presented in Table 1 and visualized in the heatmap shown in Figure 3, reveals a more nuanced picture of model performance, particularly when broken down by query complexity. Table 1 shows that on simple queries (e.g., Test Cases 1–2, 6, 10 in the heatmap), both Claude and Qwen demonstrated exceptional

performance, achieving near-perfect factual correctness (0.999 and 0.997 respectively), proving that some open-source models are highly reliable for basic data lookup and simple calculations. However, the performance on moderate and complex queries varied significantly. For moderate tasks, Claude maintained a high level of accuracy with a score of 0.984, while Qwen's performance dropped considerably to 0.690. This trend was even more pronounced with complex analytical requests, where Claude scored 0.713 and Qwen only managed 0.466, indicating a current limitation in the reasoning and multi-step execution capabilities of the open-source models within the Agentic RAG framework.

Table 1. Average calculated factual correctness by query complexity.

Query Complexity	Claude	Qwen	Llama	Number of Tests
Simple	0.999	0.997	0.568	4
Moderate	0.984	0.690	0.339	12
Complex	0.713	0.466	0.259	8

The performance gap between Claude and the open-source models widens substantially with complexity—while Qwen trails Claude by only 0.002 on simple queries, this gap expands to 0.294 on moderate queries and 0.247 on complex queries. Specific challenges involving statistical analysis, such as identifying anomalies or correlations (e.g., Test #21 and #24 in the heatmap), were particularly challenging for the open-source models. For example, on Test Case #20 (CO_2 emission calculation), Llama reported 2024.0 kg instead of the ground truth 43,843.88 kg—a 95.4% error—incorrectly extracting the year from the query rather than performing the required fuel-to-CO_2 conversion calculation. These findings showcase the advanced reasoning capabilities of Claude required for complex tasks and suggest that open-source models are currently a more viable alternative for routine exploratory analyses. Their shortcomings might also be accounted for with prompt engineering and more advanced pre-processing of the data.

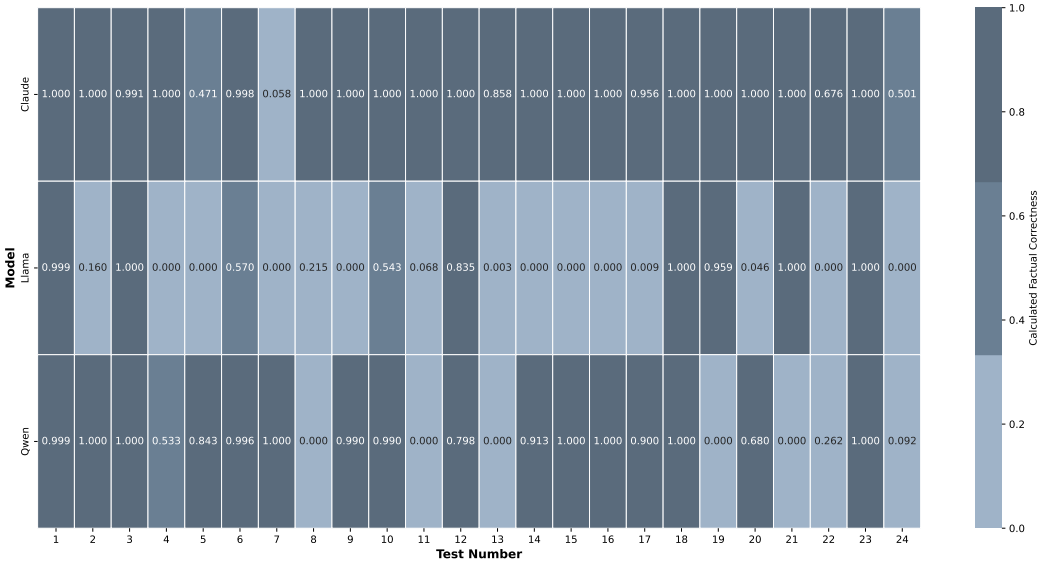


Figure 3. Heatmap of model performance (Calculated Factual Correctness) across test cases.

Notably, Figure 3 reveals several test cases where models achieved scores of 0.000, representing complete failure to produce a correct answer. These zero scores predominantly occur in the open-source models (Llama and Qwen), particularly on complex analytical queries, and are primarily attributable to fundamental misunderstandings of the query

intent or requirements. Upon detailed examination of these failure cases, we found that models either misinterpreted the question's core objective, generated queries that retrieved incorrect data subsets, or produced answers that were entirely unrelated to the ground truth. For instance, in some cases, models would correctly execute SQL queries but misinterpret which metric or aggregation was being requested, while in others, they would generate syntactically correct code that addressed a different analytical question than the one posed. These complete failures (0.000 scores) contribute to the overall performance degradation observed in Table 1, particularly for Llama, which shows the highest frequency of such failures. Importantly, while these zero scores impact the aggregate performance metrics, they do not invalidate our experimental conclusions; rather, they highlight the critical importance of query clarity, domain-specific prompt engineering, and the need for robust error detection mechanisms in production deployments.

4.2. Qualitative and Textual Generation Metrics

While our primary evaluation focused on Calculated Factual Correctness due to the analytical nature of the queries, we also assessed the linguistic quality and contextual coherence of the responses generated using the metrics provided by RAGAS (Table 2).

Table 2. Secondary Metrics: ROUGE-L F1, Semantic Similarity, and Faithfulness. All metrics are on a 0–1 scale.

Model	ROUGE-L F1	Semantic Similarity	Faithfulness
Claude 3.7	0.08	0.85	0.21
Qwen 72B	0.13	0.81	0.18
Llama 70B	0.15	0.78	0.14

The results demonstrate the limited utility of traditional NLP metrics for numerical data analysis tasks. All three models achieved very low ROUGE-L F1 scores (0.08–0.15). This is expected, as ROUGE measures lexical overlap, which is inherently low when comparing a full-sentence model response (e.g., "The most fuel-efficient ferry is Merkurius with a consumption of 6.08 L/NM") against a concise numerical ground truth.

In contrast, the Semantic Similarity scores were considerably higher (0.78–0.85) in all models. This confirms that the generated natural language content conceptually aligns with the expected answers, with Claude 3.7 achieving the highest semantic alignment.

In particular, the faith scores were uniformly low (0.14–0.21), with the best-performing model achieving only 0.21. This finding is critical, as it suggests that standard faithfulness evaluation—which measures an answer's adherence to retrieved text contexts—is ill-suited for evaluating ARAG systems. In our case, the correct answer is often a newly calculated synthesis (e.g., a statistical value) based on the context, not a direct restatement of it. These results underscore that while Semantic Similarity provides a useful secondary signal for linguistic coherence, our Calculated Factual Correctness metric remains the only reliable measure for assessing numerical accuracy in this analytical context.

4.3. An End-to-End Test Case Scenario

To further illustrate the practical application and value of our test suite, we examine test case 23: "What is the top fuel consumption spike day in ferry operations?" The Lighthouse Bot, leveraging its Agentic RAG architecture, processed this query and identified 22 January 2024 as the highest peak day, with a total fleet consumption of 3869.53 L across 214 trips (18.08 L per trip). The per-ferry breakdown showed ferry Yxlan leading with 1084.29 L, followed by Jupiter (1063.24 L) and Nina (904.63 L). The system further identified contributing factors, including high operational activity, high Passenger Car Equivalent

(PCE) loads on multiple ferries, and a mix of long-distance legs and challenging winter season conditions.

The ground truth for this test case, established using the systematic methodology described in Section 3.5, is: "The top fuel consumption spike day is 22 January 2024" with an extracted true value of "22 January 2024". This end-to-end scenario demonstrates how a natural language query is translated into data access, analysis, and actionable output that could be surfaced in an operator-facing platform or other stakeholder interface.

5. Discussion

The near-perfect performance on simple queries validates the core architecture and prompt design of the Lighthouse Bot. It confirms that an ARAG system is a highly effective approach for providing natural language access to complex, structured data in a verifiable manner. This means that we are with relatively straightforward set-up are able to fully automate a significant portion of a data analyst's work. This opens the door for developing cost-effective, self-hosted solutions for less complex, but high-volume, analytical tasks.

While impressive factual correctness on most questions, even those requiring more complex aggregation and code generation, the proprietary Claude 3.7 model misinterpreted seemingly simpler question, while Qwen performed better. Both of the questions involved the PCE metric, which may be a point of insight into the failure. The model's misinterpretation was twofold: first, its analysis correctly generated a query to find the cumulative PCE for the peak hour across the month, yet the final response reported a sub-total—specifically, the PCE for the single busiest day (86.0)—instead of the required overall sum. This indicates a failure to correctly synthesize and report the primary result of its own analysis. Second, the model identified the incorrect peak hour (9 AM), a conclusion that contradicts the ground truth (12 PM) and the results from another model. This suggests a fundamental flaw in the underlying data or a critical misstep in its initial data aggregation logic, leading to an entirely incorrect premise for the rest of its detailed analysis. The comparison highlights a vulnerability in handling basic data summarization and synthesis, even when the model can generate complex, multi-step queries. This case serves as a clear example of why a human-in-the-loop approach is crucial. For such ambiguities, the model should have the capacity to ask clarifying questions, such as "Are you looking for the total PCE for the entire month's peak hour, or the PCE for the busiest day within that peak hour?" This ability to seek clarification would prevent such errors and ensure the accuracy of the final analysis.

The clear performance hierarchy observed between Claude 3.7 and the open-source models on complex queries aligns with broader industry findings regarding the superior multi-step reasoning capabilities of large proprietary models in agentic AI systems [35,36]. However, the specific failure mode in the PCE calculation—where the agent correctly executed the analysis but reported an incorrect synthesis of the final result—underscores the challenge of "action grounding" in complex LLM systems [37]. This highlights that even with auditable code execution, the final interpretive layer remains a critical source of error. Furthermore, the strong performance of Qwen on simple queries validates the growing potential for cost-effective open-source deployments in industrial settings, where simpler retrieval and aggregation tasks comprise a majority of routine exploratory analyses. This suggests a compelling path toward decentralized, self-hosted AIoT solutions, provided the use case is carefully constrained.

While Claude 3.7 demonstrates superior accuracy, the deployment decision in maritime safety-critical contexts must critically weigh the trade-off between model performance and data sovereignty. Some proprietary models like OpenAI GPT offer enterprise-grade security features, including SOC 2 [38] and ISO 27001 [39] certifications, data processing

agreements (DPAs), and data residency guarantees that ensure data remains within specified geographic regions [40]. Some providers also offer private cloud or on-premises deployment options that provide enhanced control over data flow. However, proprietary models still operate as "black boxes" where users cannot inspect model weights, training data, or hosting infrastructure—creating significant risks in safety-critical maritime operations where regulatory compliance, data residency requirements, and operational transparency are paramount. In contrast, open-source models such as Qwen and Llama enable full model inspection, on-premises deployment, and complete control over the inference infrastructure, ensuring that sensitive operational data never leaves the organization's infrastructure. Our results show that Qwen achieved 99.7% accuracy on simple queries and 69.0% on moderate queries, demonstrating that for routine exploratory analyses open-source models provide a viable path toward self-hosted solutions that balance acceptable accuracy with critical data sovereignty requirements. This trade-off becomes particularly important in industrial contexts where operational data may be subject to national security regulations, port state control requirements, and the need for fully auditable decision-making processes.

Addressing the risk of 'out-of-the-loop familiarity' [2], the ARAG system transforms data exploration from passive consumption into active inquiry. Unlike black-box automation, our architecture preserves operator engagement through transparent execution (generated SQL and Python code is available for verification, ensuring operators maintain familiarity with underlying data structures) and active inquiry (compelling users to formulate hypotheses and iteratively refine questions rather than passively consuming static reports). This design ensures the system enhances rather than replaces human expertise, aligning with research showing that active engagement with decision support tools preserves operator competencies effectively [41,42].

It is also important to note the distinction between real-time data access and real-time analysis. While this paper emphasizes the system's capability to access and analyze real-time sensor data, it is important to clarify that the current Agentic RAG architecture is not designed for time-critical control applications requiring sub-second response times. The system's workflow typically requires 30–60 s for complex analytical queries, depending on query complexity, model selection, and data volume. This latency profile is appropriate for exploratory data analysis and strategic decision support tasks, such as identifying fuel consumption patterns, optimizing schedules, or analyzing historical trends. However, for real-time operational control scenarios—such as immediate collision avoidance, emergency response, or dynamic route adjustments during active voyages—this latency would be unacceptable. Future work should explore hybrid architectures that combine fast, deterministic control systems with slower but more flexible Agentic RAG systems for strategic analysis, or investigate optimization techniques such as query caching, pre-computed aggregations, or specialized lightweight models for time-sensitive queries to bridge this latency gap. In production environments, such systems can also be scaled through optimized database indexing, parallelized agent execution, and robust cloud infrastructure. Deploying ARAG systems in maritime industrial environments requires even addressing identified failure modes: query misinterpretation, synthesis errors (as demonstrated in the PCE calculation case), and complete failures on complex queries. For production deployment, we recommend a phased approach: initial deployment for exploratory analysis with human verification, gradual expansion to routine tasks with confidence thresholds triggering human review, and continuous monitoring through systematic evaluation. Critical safety considerations include ensuring that the system never directly controls operations without human oversight, implementing query logging and audit trails for regulatory compliance, and maintaining clear boundaries between the ARAG system's analytical

capabilities and deterministic control systems. The system's architecture—with auditable code generation, tool call limits, and execution timeouts—provides foundational safety mechanisms, but production deployment requires additional layers: role-based access control, query result validation workflows, and integration with existing maritime operational systems that enforce safety-critical constraints independently.

6. Conclusions

In this paper, we introduced Lighthouse Bot, a novel Agentic RAG system designed to address the critical need for secure, verifiable, and autonomous data analysis in maritime operations. We demonstrated that despite advances in LLMs, RAG remains essential for industrial AIoT applications, as it ensures data minimization, privacy, and auditable tool use. Through a comprehensive case study, we presented a detailed system architecture where an LLM, integrated with specialized agents, successfully transformed natural-language queries into actionable insights from complex sensor data. Our rigorous evaluation validates this approach, with a proprietary model achieving 89.6% overall factual correctness (99.9% on simple queries). Furthermore, the strong performance of an open-source model, which achieved 99% accuracy on all simple retrieval and aggregation queries, highlights the immense potential for developing cost-effective, self-hosted solutions for automating routine exploratory analyses.

This research is valuable for maritime operators, data analysts, and AIoT system developers seeking to democratize access to complex sensor data through natural language interfaces. However, several limitations should be acknowledged: our evaluation is constrained to 24 queries from a single case study, and the system's robustness in dynamic, continuously updating environments remains to be validated. Future work will focus on expanding the test suite to include a wider range of queries and data types, integrating the system with live sensor data streams, and exploring advanced agentic capabilities such as automated dashboard generation and proactive anomaly detection and incorporating static code analysis for enhanced security validation of generated code.

Author Contributions: Conceptualization, O.S.; methodology, O.S.; software, O.S., D.T. and M.A.; validation, O.S. and A.L.; formal analysis, O.S., D.T. and M.A.; investigation, O.S.; resources, O.S. and A.L.; data curation, O.S., D.T. and M.A.; writing—original draft preparation, O.S.; writing—review and editing, O.S. and A.L.; visualization, O.S.; supervision, A.L.; project administration, O.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Swedish Transport Administration (Trafikverket) through the Lighthouse industry program, grant number 309002 (Lighthouse pre-study: Retaining Swedish seafarers in the profession at sea [43]).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study, including raw data, source code, and the processed CSV datasets used to generate the figures, are openly available in the GitHub repository at https://github.com/vildly/lighthouse-sensor-bot (accessed on 1 February 2026).

Acknowledgments: During the preparation of this manuscript/study, the author(s) used Claude 3.7 Sonnet (Anthropic) via the Cursor IDE for the purposes of generating and refining synthetic test data for RAG testing, reformatting data and manuscript text into specific LaTeX templates, and performing minor linguistic refinements. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
AIoT	Artificial Intelligence of Things
API	Application Programming Interface
CoT	Chain of Thought
CSV	Comma-Separated Values
DNV	Det Norske Veritas
DSR	Decision Support System (or Data Science Research, depending on context)
FT	Fine-Tuning
GB	Gigabyte
GenAI	Generative Artificial Intelligence
GPU	Graphics Processing Unit
GPT	Generative Pre-trained Transformer
IoT	Internet of Things
LLM	Large Language Model
MQTT	Message Queuing Telemetry Transport protocol
OpenAI	Open Artificial Intelligence
PCE	Passenger Car Equivalents
RAG	Retrieval-Augmented Generation
RAGAS	Retrieval Augmented Generation Evaluation framework
PE	Prompt Engineering
REST	Representational State Transfer
SQL	Structured Query Language

Appendix A

Appendix A.1. Lighthouse Test Suite Examples

To illustrate the range and complexity of our test suite, we provide several examples below.

Example 1: Simple Retrieval Query

Query: "What is the average speed of ferry Jupiter? (in km/h)"
Ground Truth: "The average speed of ferry Jupiter in km/h is: 11.55 km/h"
Data Interaction: Retrieval

This query tests the system's ability to perform a direct lookup and unit conversion, ensuring the foundational data retrieval and transformation layers are functioning correctly.

Example 2: Complex Aggregation Query

Query: "How many vehicles are on average unable to board (left behind at the terminal) per trip for ferry Jupiter during summer months (6, 7, 8)?"
Ground Truth: "The average number of vehicles left at the terminal per trip is: 3.17"
Data Interaction: Aggregation

This example requires the system to filter data by specific criteria (ferry, months) and then perform a multi-step aggregation to calculate a final average, demonstrating its ability to handle complex, nested requests.

Example 3: Advanced Analysis Query

Query: "What is the correlation between total passengers and total fuel consumption in ferry operations?"
Ground Truth: "The correlation between total passengers and total fuel consumption is −0.501"
Data Interaction: Analysis

Appendix A.2. Test Cases Classification

Table A1. Test Cases Classification.

No.	Query	Complexity	Domain	Interaction Type
1	What is the total fuel cost for ferry Jupiter in January 2024 assuming that the price per liter is 20 sek?	Simple	Fuel Efficiency	Retrieval
2	What is the average speed of ferry Jupiter? (in km/h)	Simple	Route Optimization/Operations	Retrieval
3	How many vehicles are on average left at the terminal for inbound trips?	Moderate	Route Optimization/Operations	Aggregation
4	How many vehicles are on average unable to board (left behind at the terminal) per trip for ferry Jupiter during summer months (6, 7, 8)?	Complex	Route Optimization/Operations	Aggregation
5	What is the highest average daily PCE (Passenger Car Equivalent) value for Furusundsleden route? (outbound and inbound combined)?	Complex	Route Optimization/Operations	Aggregation
6	What is the average distance travelled on Ljusteröleden (in nautical miles)?	Simple	Route Optimization/Operations	Retrieval
7	What is the total Passenger Car Equivalent (PCE) during the peak hour for route Aspöleden in February, combining both outbound and inbound directions?	Complex	Route Optimization/Operations	Aggregation
8	What is the average fuel consumption on the route Ljusterö to Östanå, combining both outbound and inbound?	Moderate	Fuel Efficiency	Aggregation
9	What is the average trip duration for Oxdjupsleden (in minutes)?	Moderate	Route Optimization/Operations	Retrieval
10	What is the fuel consumption (liters per nautical mile) for ferry Merkurius?	Simple	Fuel Efficiency	Retrieval
11	What is the average utilization percentage for Oxdjupsleden (average PCE vs the ferry's maximum capacity)?	Complex	Route Optimization/Operations	Aggregation
12	What's the average number of vehicles left at terminals in June for Furusundsleden (combine outbound and inboud)?	Moderate	Route Optimization/Operations	Aggregation

Table A1. *Cont.*

No.	Query	Complexity	Domain	Interaction Type
13	What is the average daily vehicle throughput per day for Vaxholmsleden (only outbound)?	Moderate	Route Optimization/Operations	Aggregation
14	What's the average PCE for Oxdjupsleden on Thursdays (calculated as the mean of the average inbound PCE and average outbound PCE)?	Moderate	Route Optimization/Operations	Aggregation
15	Which hour of the day has the highest average passenger car equivalent (PCE) for route Ljusteröleden (with inbound and outbound trips combined)?	Moderate	Route Optimization/Operations	Aggregation
16	What is the average passenger car equivalent (PCE) during June, July, August for Vaxholmsleden route (calculated as the mean of the average inbound PCE and average outbound PCE)?	Moderate	Route Optimization/Operations	Aggregation
17	What is the difference in percent of pedestrians to vehicles, comparing weekdays to weekends for Vaxholmsleden?	Moderate	Route Optimization/Operations	Comparison
18	At which hour does Oxdjupsleden on average have the highest passenger car equivalent (PCE)?	Moderate	Route Optimization/Operations	Aggregation
19	What was Aspöleden's ratio of pedestrians to vehicles in 2023?	Moderate	Route Optimization/Operations	Aggregation
20	What is the total CO_2 emission from Ljusteröleden in February 2024, assuming 2.68 kg CO_2 per liter of fuel?	Moderate	Environmental Impact	Aggregation
21	What is the top most anomalous day in terms of passenger demand for ferry operations?	Complex	Route Optimization/Operations	Analysis
22	What is the percent deviation of total passengers on 22 June 2023 compared to the average daily passenger demand?	Complex	Route Optimization/Operations	Analysis
23	What is the top fuel consumption spike day in ferry operations?	Complex	Fuel Efficiency	Analysis
24	What is the correlation between total passengers and total fuel consumption in ferry operations?	Complex	Fuel Efficiency	Analysis

References

1. Raza, Z.; Woxenius, J.; Vural, C.A.; Lind, M. Digital transformation of maritime logistics: Exploring trends in the liner shipping segment. *Comput. Ind.* **2023**, *145*, 103811. [CrossRef]
2. DNV; Singapore Maritime Foundation. The Future of Seafarers 2030: A Decade of Transformation. 2023. Available online: https://www.smf.com.sg/wp-content/uploads/2023/04/The-Future-of-Seafarers-2030-A-Decade-of-Transformation.pdf (accessed on 1 November 2025).
3. Zeng, F.; Chen, A.; Xu, S.; Chan, H.K.; Li, Y. Digitalization in the Maritime Logistics Industry: A Systematic Literature Review of Enablers and Barriers. *J. Mar. Sci. Eng.* **2025**, *13*, 797. [CrossRef]
4. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *arXiv* **2017**, arXiv:1706.03762. [CrossRef]
5. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language Models are Few-Shot Learners. In *Proceedings of the Advances in Neural Information Processing Systems*; Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2020; Volume 33, pp. 1877–1901. Available online: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf (accessed on 1 November 2025).
6. OpenAI. OpenAI Website. 2025. Available online: https://www.openai.com (accessed on 1 November 2025).

7. Ma, P.; Ding, R.; Wang, S.; Han, S.; Zhang, D. InsightPilot: An LLM-Empowered Automated Data Exploration System. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Singapore, 6–10 December 2023*; Feng, Y., Lefever, E., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2023; pp. 346–352. [CrossRef]
8. Qi, D.; Miao, Z.; Wang, J. CleanAgent: Automating Data Standardization with LLM-Based Agents. In Proceedings of the VLDB 2025 Workshop on Data Driven AI (DDAI), 51st International Conference on Very Large Data Bases (VLDB 2025), VLDB Workshops, London, UK, 1–5 September 2025. Available online: https://github.com/sfu-db/CleanAgent (accessed on 1 November 2025).
9. Bouchard, L.F.; Peters, L. *Building LLMs for Production: Enhancing LLM Abilities and Reliability with Prompting, Fine-Tuning, and RAG*; Towards AI Inc.: Wilmington, DE, USA, 2024.
10. IBM. RAG vs. Fine-tuning: Enterprise AI Strategy Guide. IBM Blog/Resource. In *Stresses that RAG Is Preferred for Compliance and Governance*; IBM: Armonk, NY, USA, 2024. Available online: https://www.ibm.com/think/topics/rag-vs-fine-tuning (accessed on 1 November 2025).
11. Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.T.; Rocktäschel, T.; et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. *arXiv* **2020**, arXiv:2005.11401. [CrossRef]
12. Lee, H.H.; Chen, C.C.; Yen, A.Z. RAG-Enhanced Evidence Recommendation in Financial Legal Resolutions. In Proceedings of the Companion Proceedings of the ACM on Web Conference 2025, WWW '25, New York, NY, USA, 28 April–2 May 2025; pp. 1096–1099. [CrossRef]
13. Hindi, M.; Mohammed, L.; Maaz, O.; Alwarafy, A. Enhancing the Precision and Interpretability of Retrieval-Augmented Generation (RAG) in Legal Technology: A Survey. *IEEE Access* **2025**, *13*, 46171–46189. [CrossRef]
14. Huhtanen, M. AI-Powered Legal Virtual Assistant: Utilizing RAG-Optimized LLM for Housing Dispute Resolution in Finland. 2024. Available online: https://www.theseus.fi/handle/10024/859826 (accessed on 1 November 2025).
15. Sinha, K.; Singh, V.; Vishnoi, A.; Madan, P.; Shukla, Y. Healthcare Diagnostic RAG-Based Chatbot Triage Enabled by BioMistral-7B. In Proceedings of the 2024 International Conference on Emerging Technologies and Innovation for Sustainability (EmergIN), Greater Noida, India, 20–21 December 2024; pp. 333–338. [CrossRef]
16. Srivastava, A.K.; Lohia, A.; Raj, A.; Mishra, A.K.; Sagar, H.; Lakhotia, S. HealthHub: An IoT-Enabled Real-Time Health Monitoring System Using Retrieval-Augmented Generation. *Strad Res.* **2025**, *12*, 20–24. [CrossRef]
17. Danvirutai, P.; Charoenwattanasak, S.; Tola, S.; Thaiso, K.; Yuangsoi, B.; Minh, H.T.; Srichan, C. An integrating RAG-LLM and deep Q-network framework for intelligent fish control systems. *Sci. Rep.* **2025**, *15*, 21377. [CrossRef] [PubMed]
18. Oh, S.; Kum, S.; Moon, J. Real-Time Environment Monitoring and Response Through IoT and Retrieval-Augmented Generation. In Proceedings of the 2024 15th International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Republic of Korea, 16–18 October 2024; pp. 1658–1659. [CrossRef]
19. Barnett, S.; Kurniawan, S.; Thudumu, S.; Brannelly, Z.; Abdelrazek, M. Seven Failure Points When Engineering a Retrieval Augmented Generation System. In *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering—Software Engineering for AI, CAIN '24, Lisbon, Portugal, 14–15 April 2024*; Association for Computing Machinery: New York, NY, USA, 2024; pp. 194–199. [CrossRef]
20. Soto, M. *Agentic Retrieval-Augmented Generation for Industrial Anomaly Detection*; Zenodo: Geneva, Switzerland, 2025. [CrossRef]
21. Elewah, A.; Elgazzar, K.; Elnaffar, S. Agentic Search Engine for Real-Time Internet of Things Data (with Full Text). *Sensors* **2025**, *25*, 5995. [CrossRef] [PubMed]
22. Jin, J.; Fu, X.; Gao, X.; Cheng, T.; Yan, R. MSD-LLM: Predicting Ship Detention in Port State Control Inspections with Large Language Model. *arXiv* **2025**, arXiv:2505.19568. [CrossRef]
23. Sun, S.; Zhao, L.; Deng, M.; Fu, X. VTS-LLM: Domain-Adaptive LLM Agent for Enhancing Awareness in Vessel Traffic Services through Natural Language. *arXiv* **2025**, arXiv:2505.00989. [CrossRef]
24. Miller, T.; Durlik, I.; Kostecka, E.; Łobodzińska, A.; Łazuga, K.; Kozlovska, P. Leveraging Large Language Models for Enhancing Safety in Maritime Operations. *Appl. Sci.* **2025**, *15*, 666. [CrossRef]
25. Sánchez-Heres, L.; Weber, R.; Ahlgren, F.; Olsson, F.; Sachenkova, O. *COLREG3—Exploring the Potential of Large Language Models in Marine Navigation Systems*; Technical Report; Lighthouse: Seattle, WA, USA, 2024. [CrossRef]
26. Lundström, O.; Nilsson, A.; Johansson, M. Large Language Models (LLMs) in Maritime Data Analysis and Decision Support. In *A Pre-Study Initiated by Lighthouse*; Lighthouse: Seattle, WA, USA, 2025. [CrossRef]
27. Andreasson, M.; Tan, D. Lighthouse Bot: A Platform for Evaluating LLMs for Agentic Maritime Data Analysis. 2025. Available online: https://www.diva-portal.org/smash/record.jsf?pid=diva2:1969025 (accessed on 1 November 2025).
28. Peffers, K.; Tuunanen, T.; Rothenberger, M.A.; Chatterjee, S. A design science research methodology for information systems research. *J. Manag. Inf. Syst.* **2007**, *24*, 45–77. [CrossRef]
29. Sachenkova, O. *Lighthouse Bot: Agentic RAG System for Maritime Sensor Data Analysis*; GitHub Repository; Linnaeus University: Kalmar, Sweden, 2025. Available online: https://github.com/vildly/lighthouse-sensor-bot (accessed on 1 November 2025).

30. Zhang, B.; Zhou, K.; Wei, X.; Zhao, X.; Sha, J.; Wang, S.; Wen, J.R. Evaluating and improving tool-augmented computation-intensive math reasoning. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 23570–23589.
31. Lu, H.; Li, X.; Ji, X.; Kan, Z.; Hu, Q. ToolFiVe: Enhancing Tool-Augmented LLMs via Tool Filtering and Verification. In *Proceedings of the ICASSP 2025—2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Hyderabad, India, 6–11 April 2025*; IEEE: New York, NY, USA, 2025; pp. 1–5. [CrossRef]
32. Research Institutes of Sweden. *Hack-A-Fleet v2.0: Repository Containing Python Tools and Datasets*; GitHub Repository; Research Institutes of Sweden: Göteborg, Sweden, 2024. Available online: https://github.com/RISE-Maritime/hack-a-fleet (accessed on 1 November 2025).
33. Research Institutes of Sweden. *PONTOS Datahub: An Open Datahub of Ship Operational Data*; Website/Data Resource; Research Institutes of Sweden: Göteborg, Sweden, 2024. Available online: https://pontos.ri.se/ (accessed on 1 November 2025).
34. Es, S.; James, J.; Anke, L.E.; Schockaert, S. Ragas: Automated evaluation of retrieval augmented generation. In Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations, St. Julians, Malta, 17–22 March 2024; pp. 150–158.
35. Patel, S.; Dholakiya, R.A. Large Language Models: Evolution, Architecture, Applications, and Future Horizons. In Proceedings of the 2025 5th International Conference on Soft Computing for Security Applications (ICSCSA), Salem, India, 4–6 August 2025; pp. 1922–1929. [CrossRef]
36. Chang, Y.; Wang, X.; Wang, J.; Wu, Y.; Yang, L.; Zhu, K.; Chen, H.; Yi, X.; Wang, C.; Wang, Y.; et al. A Survey on Evaluation of Large Language Models. *ACM Trans. Intell. Syst. Technol.* **2024**, *15*, 39. [CrossRef]
37. Rivera, C.; Byrd, G.; Paul, W.; Feldman, T.; Booker, M.; Holmes, E.; Handelman, D.; Kemp, B.; Badger, A.; Schmidt, A.; et al. ConceptAgent: LLM-Driven Precondition Grounding and Tree Search for Robust Task Planning and Execution. In Proceedings of the 2025 IEEE International Conference on Robotics and Automation (ICRA), Atlanta, GA, USA, 19–23 May 2025; pp. 8988–8995. [CrossRef]
38. AICPA. *SOC 2 Reporting on an Examination of Controls at a Service Organization Relevant to Security, Availability, Processing Integrity, Confidentiality, or Privacy (Guide)*; American Institute of Certified Public Accountants (AICPA): Durham, NC, USA, 2022. Available online: https://www.aicpa-cima.com/cpe-learning/publication/soc-2-reporting-on-an-examination-of-controls-at-a-service-organization-relevant-to-security-availability-processing-integrity-confidentiality-or-privacy (accessed on 3 November 2025).
39. *ISO/IEC 27001:2022*; Information Security, Cybersecurity and Privacy Protection—Information Security Management Systems—Requirements. International Organization for Standardization: Geneva, Switzerland, 2022. Available online: https://www.iso.org/standard/82875.html (accessed on 3 November 2025)).
40. OpenAI. Expanding Data Residency Access to Business Customers Worldwide. 2025. Available online: https://openai.com/index/expanding-data-residency-access-to-business-customers-worldwide/ (accessed on 1 November 2025).
41. Turner, C.J.; Ma, R.; Chen, J.; Oyekan, J. Human in the Loop: Industry 4.0 Technologies and Scenarios for Worker Mediation of Automated Manufacturing. *IEEE Access* **2021**, *9*, 103950–103966. [CrossRef]
42. Natarajan, S.; Mathur, S.; Sidheekh, S.; Stammer, W.; Kersting, K. Human-in-the-loop or AI-in-the-loop? Automate or Collaborate? *Proc. AAAI Conf. Artif. Intell.* **2025**, *39*, 28594–28600. [CrossRef]
43. Swedish Transport Administration (Trafikverket). Lighthouse: Collaboration Platform for Maritime Research and Innovation. Available online: https://lighthouse.nu/en/ (accessed on 1 February 2026).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.