

Efficient Large Language Model Application Development: A Case Study of Knowledge Base, API, and Deep Web Search Integration

Xiangyu Wang¹, Yan Tan², Tao Yang², Meng Yuan², Shaohan Wang¹, Min Chen¹, Feiyang Ren¹, Zijian Zhang¹, Yuqi Shao¹

¹COSCO SHIPPING Technology Co., Ltd., Shanghai, China

²COSCO SHIPPING Specialized Carriers Co., Ltd., Guangzhou, China

Email: wang.xiangyu@coscoshipping.com, tan.yan@coscoshipping.com, yang.tao2@coscoshipping.com, yuan.meng@coscoshipping.com, chen.min9@coscoshipping.com, ren.feiyang@coscoshipping.com, zhang.zijian@coscoshipping.com, shao.yuqi@coscoshipping.com

How to cite this paper: Wang, X.Y., Tan, Y., Yang, T., Yuan, M., Wang, S.H., Chen, M., Ren, F.Y., Zhang, Z.J. and Shao, Y.Q. (2024) Efficient Large Language Model Application Development: A Case Study of Knowledge Base, API, and Deep Web Search Integration. *Journal of Computer and Communications*, 12, 171-200.

<https://doi.org/10.4236/jcc.2024.1212011>

Received: December 5, 2024

Accepted: December 27, 2024

Published: December 30, 2024

Copyright © 2024 by author(s) and Scientific Research Publishing Inc.

This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).

<http://creativecommons.org/licenses/by/4.0/>



Open Access

Abstract

This paper presents a reference methodology for process orchestration that accelerates the development of Large Language Model (LLM) applications by integrating knowledge bases, API access, and deep web retrieval. By incorporating structured knowledge, the methodology enhances LLMs' reasoning abilities, enabling more accurate and efficient handling of complex tasks. Integration with open APIs allows LLMs to access external services and real-time data, expanding their functionality and application range. Through real-world case studies, we demonstrate that this approach significantly improves the efficiency and adaptability of LLM-based applications, especially for time-sensitive tasks. Our methodology provides practical guidelines for developers to rapidly create robust and adaptable LLM applications capable of navigating dynamic information environments and performing effectively across diverse tasks.

Keywords

Large Language Model, Knowledge Base, API Integration, Web Retrieval, Application Development

1. Introduction

1.1. Background and Motivation

Large Language Models (LLMs) have become increasingly important in a wide range of real-world applications, from natural language processing tasks such as

text summarization, translation, and sentiment analysis, to more complex tasks like code generation, scientific research, and personalized content creation [1] [2]. These models, trained on vast amounts of textual data, exhibit remarkable capabilities in understanding and generating human-like text. However, despite their impressive performance, LLMs face several significant challenges when it comes to developing efficient and effective applications.

One of the primary challenges is the lack of structured knowledge. While LLMs are adept at handling unstructured text, they often struggle with tasks that require access to specific, well-organized information. For example, in domains such as healthcare, finance, and legal, where accuracy and precision are critical, LLMs need to be augmented with structured knowledge to ensure reliable and contextually accurate responses [3].

Another challenge is the need for real-time data. Many applications, especially those in dynamic environments, require up-to-date information. For instance, news aggregation, maritime market analysis, and weather forecasting all benefit from real-time data [4]. LLMs, which are typically trained on static datasets, may not be able to provide the most current information without access to external data sources.

Finally, there is the issue of adaptability. As new information emerges and existing knowledge evolves, LLMs must be able to adapt and update their understanding. This is particularly crucial in fields where information changes rapidly, such as technology, science, and public health [5]. Without the ability to incorporate new data, LLMs risk becoming outdated and less effective over time.

To address these challenges, it is essential to integrate external information sources into LLM-based applications. By combining knowledge bases, API integration, and web retrieval, we can enhance the reasoning capabilities of LLMs, extend their functionality, and ensure they stay updated with the latest information. This integrated approach not only improves the performance of LLMs but also makes them more adaptable and versatile in a variety of tasks [6].

1.2. Objective and Contributions

The primary objective of this paper is to introduce an innovative integrated approach that amalgamates knowledge bases, API integration, and web retrieval to streamline and expedite the development of Large Language Model (LLM)-based applications. Our work presents several key contributions:

- **Comprehensive Integrated Framework:** We develop a holistic framework that seamlessly combines structured knowledge from various knowledge bases, real-time data accessed through APIs, and the most recent information retrieved from the web. This integration is designed to bolster the reasoning capabilities of LLMs, enabling them to execute complex tasks with greater efficacy.
- **Enhanced Performance in Dynamic and Time-Sensitive Tasks:** Our approach is tailored to elevate the performance of LLMs, particularly in environments that demand up-to-date and accurate information. By leveraging real-time

data and the latest web-derived content, our framework ensures that LLMs generate responses that are both precise and timely.

- **Empirical Validation through Real-World Case Studies:** We demonstrate the practical effectiveness of our integrated approach through detailed case studies in real-world scenarios. These case studies illustrate how the synergy of structured knowledge, API integration, and web retrieval can significantly enhance the efficiency, adaptability, and overall performance of LLM-based applications.
- **Actionable Insights and Best Practices:** The paper offers valuable guidelines and best practices for developers aiming to build robust and versatile LLM applications. This includes strategies for selecting and integrating knowledge bases, designing and implementing APIs, and optimizing web retrieval processes to maximize the capabilities of LLMs.

1.3. Paper Structure

The remainder of this paper is organized as follows:

- **Section 2: Related Work:** This section reviews existing literature on Large Language Models, knowledge bases, API integration, and web retrieval. It identifies current research gaps and contextualizes our proposed integrated approach within the broader academic discourse.
- **Section 3: Methodology:** We detail the proposed integrated framework, encompassing the selection and integration of knowledge bases, the design and implementation of APIs, and the utilization of web retrieval techniques. This section also elaborates on the system architecture and key algorithms that underpin our approach.
- **Section 4: Experimental Setup and Case Studies:** This section outlines the experimental design, including the environments, datasets, and evaluation metrics employed. It presents real-world case studies that illustrate the application and effectiveness of our integrated approach, followed by a discussion of the experimental results.
- **Section 5: Results and Discussion:** We provide a comprehensive analysis of the experimental outcomes, highlighting the significance of our findings. This section compares our approach with existing methodologies, discusses its advantages and limitations, and explores potential avenues for future research.
- **Section 6: Conclusion:** The concluding section summarizes the main contributions of the paper, underscores the practical implications of our integrated framework, and suggests directions for subsequent research and development efforts aimed at further enhancing LLM-based applications.

2. Related Work

2.1. Existing Approaches in LLM Application Development

Many methods and frameworks have been developed to help build applications using Large Language Models (LLMs). One popular framework is LangChain [7].

LangChain provides tools to call LLMs, create prompts, and parse responses. It also helps manage memory and integrate with external data. Another approach is Retrieval-Augmented Generation (RAG) [8]. RAG uses a retriever to find relevant documents and then generates responses based on this information. This improves the accuracy and relevance of the LLM's output. Tools like Hugging Face's Transformers library and OpenAI's GPT-3 API are also widely used [9] [10]. These tools offer pre-trained models and APIs that can be easily integrated into various applications.

2.2. Use of Knowledge Bases in LLMs

Structured knowledge bases can enhance the reasoning capabilities of LLMs. Researchers have explored how to use knowledge bases to improve LLM performance [11]. For example, some studies have shown that integrating knowledge graphs with LLMs can make the models more interpretable and perform better [12]. Knowledge graphs provide structured information, such as facts, concepts, and relationships, which can help LLMs understand and generate more accurate text. Other research has focused on extracting symbolic knowledge from LLMs and using it to build knowledge bases [13]. This can help LLMs handle tasks that require specific and detailed information, such as answering questions in specialized domains.

2.3. API Integration in LLMs

API integration is another way to extend the functionalities of LLMs. By connecting LLMs to external services through APIs, developers can access real-time data and additional features. For example, integrating LLMs with RESTful APIs allows them to interact with web services [14]. This can be useful for tasks that need up-to-date information, such as maritime analysis or weather forecasting. Some studies have shown that fine-tuning LLMs on API specifications can improve their performance in both in-distribution and out-of-distribution tasks [15]. This means that LLMs can learn to use APIs more effectively and handle a wider range of tasks.

2.4. Web Retrieval in LLMs

Web retrieval techniques help LLMs stay updated with the latest information. By searching the web, LLMs can access new and relevant data. For example, LLMs can be used to search the web for the latest news or scientific articles [16]. Some researchers have proposed frameworks that combine web retrieval with LLMs to improve their performance in tasks like question answering [17]. These frameworks use web search results to provide LLMs with the most current and relevant information, making their responses more accurate and up-to-date.

2.5. Embedding Models in Retrieval-Augmented Generation

Vector embeddings are central to RAG applications, capturing semantic information

of data objects (e.g., text, images) and representing them as numerical arrays [18]. The selection of appropriate embedding models is crucial for the performance of RAG systems. Factors influencing this choice include the specific use case, modality, domain specificity, and computational resources [19]. Recent benchmarks like the Massive Text Embedding Benchmark (MTEB) provide comprehensive evaluations of embedding models across various tasks and metrics [20]. However, it's imperative to evaluate embedding models on domain-specific datasets to ensure optimal performance [21].

3. Methodology

3.1. Knowledge Base Integration

To enhance the reasoning and decision-making capabilities of Large Language Models (LLMs), we integrate structured knowledge from databases and ontologies. This process involves several key steps. First, we select relevant knowledge bases that contain structured information, such as Wikipedia, DBpedia, or domain-specific ontologies. These knowledge bases provide a rich source of factual and contextual information.

The selected knowledge bases undergo data preprocessing to extract and format the data for easy integration into the LLM. This includes cleaning the data, normalizing it, and converting it into suitable formats like JSON or PDF. Following preprocessing, the data is embedded using techniques such as word embeddings (e.g., Word2Vec, GloVe) or more advanced methods like BERT embeddings. The embedded data is then indexed to allow efficient retrieval.

The indexed and embedded knowledge is integrated into the LLM using a dual-decoder approach. One decoder generates text based on the LLM's internal knowledge, while the other utilizes the external knowledge base. The outputs from both decoders are combined to produce a final, more accurate response. This integration offers several benefits, including enhanced reasoning, improved accuracy, and better contextual understanding, which collectively contribute to more relevant and coherent responses.

3.2. API Integration and Design

API integration and design are crucial steps in extending the functionalities of LLMs. By connecting LLMs to external services through APIs, we can access real-time data and additional features, thereby enhancing the overall performance and versatility of the application.

The process begins with selecting open APIs that provide relevant and reliable data, such as weather APIs, maritime tracking APIs, and news APIs. Once APIs are selected, their specifications are parsed to understand available endpoints, request formats, and response structures. This ensures that the LLM can interact with the APIs correctly.

When designing APIs, several key principles are followed to ensure clarity and efficiency. Each API should have well-defined endpoints that specify the exact

functionality they provide, helping the LLM understand what actions can be performed and how to interact with the API effectively. APIs must have clearly defined input parameters and output responses, allowing the LLM to know exactly what data to send and what results to expect, thereby reducing ambiguity and enhancing reliability.

APIs should perform specific, well-defined tasks. By narrowing the scope of each API, developers make it easier for the LLM to understand and utilize the API correctly, improving the accuracy of the responses generated. Additionally, APIs should adhere to established standards such as Swagger 2.0 and OpenAPI 3.0. Compliance with these standards ensures that APIs are documented consistently, making them easier to integrate and maintain. Finally, APIs should be designed to allow modifications without significant disruptions. This flexibility ensures that APIs can evolve with the application's needs, maintaining their relevance and effectiveness.

The LLM generates API requests based on user input or task requirements, formatting these requests according to the API specifications. Upon receiving responses from the APIs, the system processes and integrates the data into the LLM's output. This integration can involve embedding the response data directly into the generated text or using the data to inform the LLM's reasoning process.

In our experimental case studies, we integrated several APIs categorized into two main domains: Vessels and Ports. For Vessels, we included APIs to retrieve vessel details by MMSI, batch retrieve the latest AIS information, obtain current vessel status, fetch meteorological details, gather port efficiency statistics, retrieve voyage and leg efficiency statistics, obtain refueling and tug operations efficiency statistics, compile vessel operational performance data, and access PSC data records by MMSI. For Ports, we included APIs to retrieve port details by code, calculate average distances between ports, batch retrieve current weather for multiple ports, predict future port dynamics and congestion indicators, collect periodic port data statistics, retrieve lists of ships at port, and access port repair details.

To illustrate the practical application of our API integration and design principles, we implemented these APIs into our RAG application. Each API endpoint was defined with clear parameters and responses, adhering to Swagger 2.0 and OpenAPI 3.0 specifications. This standardization facilitated seamless integration and interaction with the LLM. The LLM identifies the required information and selects the appropriate API endpoint based on the query. For instance, to retrieve vessel details, the LLM uses the `/v1/vessels/{mmsi}/detail` endpoint with the specific MMSI as the parameter. Clear input parameters ensure accurate API requests, and the LLM processes the API responses by extracting relevant information to formulate accurate and contextually appropriate answers to user queries. In cases where existing APIs did not fully meet the application's requirements, modifications were made to narrow their capabilities, ensuring that each API performed specific tasks. For example, if an API initially provided broad weather data, it was refined to return only essential meteorological details

pertinent to vessel operations.

The key benefits of API integration and design include access to real-time data, extended functionality, and improved accuracy by ensuring that responses are current and relevant. This integration allows the LLM to perform a wider range of tasks, such as maritime analysis, weather forecasting, and news aggregation, thereby enhancing the overall utility and effectiveness of the application.

3.2.1. Experimental API Integration

In our experimental case studies, we integrated the following APIs to demonstrate the practical application of our API design and selection principles. The APIs were categorized into two main domains: Ship and Port.

A) Ship APIs

1) Retrieve Vessel Details by MMSI

Endpoint: `/v1/vessels/{mmsi}/detail`

Function: Fetches basic information about a vessel using its Maritime Mobile Service Identity (MMSI).

2) Batch Retrieve Latest AIS Information by MMSI

Endpoint: `/v1/vessels/status/location`

Function: Retrieves the latest Automatic Identification System (AIS) data for multiple vessels based on their MMSI numbers.

3) Retrieve Current Vessel Status by MMSI

Endpoint: `/v1/vessels/status/current/{mmsi}`

Function: Obtains the current dynamic status of a specific vessel using its MMSI.

4) Fetch Meteorological Details by Time and Location

Endpoint: `/v2/weather/point`

Function: Provides detailed weather information for a specific time and geographical location.

5) Batch Retrieve Port Efficiency Statistics for Vessels

Endpoint: `/vessels/analytics/efficiency/port/batch`

Function: Gathers operational efficiency statistics for vessels across different ports.

6) Batch Retrieve Voyage Efficiency Statistics

Endpoint: `/v1/vessels/analytics/efficiency/batch`

Function: Collects efficiency data related to voyages and distances covered by vessels.

7) Batch Retrieve Leg Efficiency Statistics

Endpoint: `/v1/vessels/analytics/efficiency/leg/batch`

Function: Retrieves efficiency metrics for specific legs of vessel journeys.

8) Batch Retrieve Refueling Efficiency Statistics

Endpoint: `/v1/vessels/analytics/efficiency/refuel/batch`

Function: Obtains efficiency data related to vessel refueling operations.

9) Batch Retrieve Tug Operations Efficiency Statistics

Endpoint: /v1/vessels/analytics/efficiency/tug/batch

Function: Provides efficiency statistics for tugboat operations assisting vessels.

10) Batch Retrieve Vessel Operational Performance Data

Endpoint: /v1/vessels/analytics/performance

Function: Compiles performance data related to vessel operations.

11) Retrieve PSC Data Records by MMSI

Endpoint: /v1/vessels/psc

Function: Accesses Port State Control (PSC) data records for vessels using their MMSI.

B) Ports APIs

1) Retrieve Port Details by Code

Endpoint: /v1/ports/code/{code}

Function: Fetches detailed information about a port using its unique code.

2) Calculate Average Distance Between Two Ports (Nautical Miles)

Endpoint: /v1/ports/distance

Function: Computes the average nautical distance between two specified ports.

3) Batch Retrieve Current Weather for Multiple Ports

Endpoint: /v1/ports/weather/batch/current

Function: Provides current weather data for multiple ports based on their codes.

4) Predict Future Port Dynamics and Congestion Indicators

Endpoint: /v1/ports/analytics/dynamic/future/count

Function: Forecasts future port dynamics and congestion levels.

5) Port Data Periodic Statistics (Number of Ships and Deadweight Tonnage)

Endpoint: /v1/ports/analytics/repair/stats

Function: Collects periodic statistics on the number of ships and their deadweight tonnage at ports.

6) Retrieve Lists of Ships at Port (Docking, Anchoring, Arriving, Repairing)

Endpoint: /v1/ports/analytics/dynamic/current/count

Function: Provides lists of ships currently docking, anchoring, arriving, or undergoing repairs at ports.

7) Retrieve Port Repair Details

Endpoint: /v1/ports/analytics/repair/detail

Function: Accesses detailed repair information for ships at ports.

3.2.2. API Implementation Example

To illustrate the practical application of our API design and selection principles, we integrated the aforementioned APIs into our RAG application. Below is an example of how these APIs were implemented:

- **API Endpoint Definition:** Each API endpoint was defined with clear parameters and responses, adhering to Swagger 2.0 and OpenAPI 3.0 specifications. This standardization facilitated seamless integration and interaction with the LLM.
- **Parameter Specification:** Clear input parameters are provided to ensure

accurate API requests. For example, the `/v1/ports/code/{code}` endpoint requires a specific port code to fetch the corresponding port details.

- **Response Handling:** The LLM processes the API responses, extracting relevant information to formulate accurate and contextually appropriate answers to user queries.

3.2.3. API Modification and Optimization

In cases where existing APIs did not fully meet the application's requirements, modifications were made to narrow their capabilities. This approach ensured that each API performed specific tasks, making it easier for the LLM to understand and utilize them effectively. For example, if an API initially provided broad weather data, it was refined to return only essential meteorological details pertinent to vessel operations. Additionally, to manage updates or changes in API specifications over time, we implemented the following strategies:

- **Versioning and Documentation:** Each API was versioned systematically, and comprehensive documentation was maintained. This allows the system to handle multiple API versions concurrently and facilitates smoother transitions when updates occur.
- **Automated Testing and Monitoring:** We established automated testing pipelines to regularly verify API functionality and response structures. Continuous monitoring tools were deployed to detect any changes or anomalies in API behavior promptly.
- **Flexible Integration Layers:** An abstraction layer was introduced between the LLM and the APIs. This layer handles discrepancies in API specifications, enabling the system to adapt to changes without requiring extensive modifications to the core application logic.
- **Fallback Mechanisms:** In scenarios where an API update causes temporary incompatibility, fallback mechanisms were implemented to switch to previous stable versions or alternative APIs, ensuring uninterrupted service and maintaining system reliability.

3.3. Deep Web Retrieval Integration

In addition to integrating knowledge bases and APIs, incorporating web retrieval is essential for accessing up-to-date information from the internet. Our approach involves a deep internet retrieval process that enables the system to gather, process, and utilize the most relevant online content to enhance the responses generated by the Large Language Model (LLM). The flowchart of deep Internet retrieval is shown in **Figure 1**.

The web retrieval process begins when a user enters a query into the system. This query is sent to the Bing Web Search API, which searches the internet and returns the top 20 web results related to the query. Using the Bing API ensures that the system has access to a wide range of current and relevant web pages.

After receiving the list of top web results, the system retrieves the HTML

content of each website by sending HTTP requests to access each webpage directly. Once the HTML content is obtained, the system cleans and processes it to extract meaningful text. This cleaning process involves removing HTML tags, scripts, advertisements, and any other non-text elements that are not useful for understanding the content.

The extracted text from each website is then divided into smaller pieces, such as paragraphs or sentences. This segmentation helps in managing the content more effectively and allows the system to analyze the information in detail. By breaking down the text, the system can focus on specific pieces of information that are most relevant to the user's query.

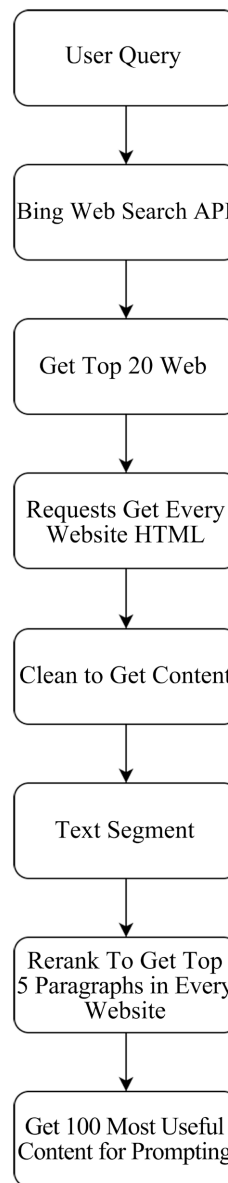


Figure 1. Deep web retrieval flowchart.

To identify the most important pieces of information, the system evaluates each

text segment and ranks them based on how closely they relate to the original user query. This reranking process involves analyzing the content of each segment to determine its relevance. For each website, the system selects the top five paragraphs that are most relevant to the query.

From all the processed websites, the system collects these top segments, resulting in approximately 100 of the most useful pieces of content. These selected pieces of information are then used to enhance the LLM's responses. By incorporating these relevant and up-to-date content pieces into the prompt, the LLM can generate answers that are more accurate, informative, and aligned with the latest information available on the internet.

This deep internet retrieval process ensures that the LLM-based application can provide users with current and relevant information, even in rapidly changing fields. It improves the system's ability to access a wide range of data, process it efficiently, and utilize it effectively to enhance the quality of the generated responses.

By implementing this methodology, we enable the system to stay updated with the most recent information, which is crucial for applications that require real-time data. It also allows the LLM to understand and incorporate diverse perspectives from multiple sources, leading to more comprehensive and reliable answers.

3.4. Embedding Model Selection for RAG Applications

Selecting the appropriate embedding model is pivotal for the effectiveness of Retrieval-Augmented Generation (RAG) applications. The embeddings capture the semantic information of data objects and are essential for tasks like semantic search and similarity computations. Our methodology for selecting embedding models involves the following steps:

3.4.1. Determining the Specific Use Case

We begin by analyzing the specific requirements of the RAG application:

- **Modality Consideration:** Determine whether the application involves text, images, or multimodal data. For text-based applications, text embeddings suffice, while multimodal embeddings are necessary for applications involving multiple data types [22].
- **Domain Specificity:** Assess if the application operates within a specialized domain (e.g., legal, medical). Domain-specific embedding models may provide better performance due to their training on relevant data [23].

In most scenarios, a general-purpose embedding model is adequate for text-based RAG applications.

3.4.2. Selecting a General-Purpose Embedding Model

To select an appropriate general-purpose embedding model, we consider:

- **Performance Benchmarks:** Utilize resources like the Massive Text Embedding Benchmark (MTEB) leaderboard, which lists various embedding models along with their performance metrics across tasks such as retrieval and summarization [20].

- **Task Alignment:** Focus on models that perform well on retrieval tasks, as this aligns with the needs of RAG applications.
- **Language Compatibility:** Ensure the embedding model supports the language(s) used in the application dataset.
- **Model Size and Computational Resources:** Balance the trade-off between model performance and computational efficiency. Larger models may offer higher accuracy but at the cost of increased latency and resource consumption [24].
- **Embedding Dimension:** Consider the dimensionality of the embeddings. Higher dimensions may capture more nuanced details but also increase computational complexity and storage requirements. A balance must be struck to ensure efficiency without compromising performance [25].

3.4.3. Evaluating Embedding Models on Application Data

While benchmarks provide a general performance overview, it's crucial to evaluate embedding models on the application's specific dataset to ensure they meet the unique requirements of our case studies. The evaluation process involves several key steps:

- **Dataset Preparation:** We curated representative datasets from our shipping-related data, including labeled examples for tasks such as vessel tracking accuracy and port status retrieval. This ensures that the evaluation reflects real-world scenarios encountered in our case studies.
- **Embedding Generation:** Embeddings were generated for the dataset using different candidate models, including BERT, FastText, and specialized maritime-domain embeddings. This step ensures that each model's performance is assessed in the context of our specific application needs.
- **Performance Metrics:** Models were evaluated using metrics such as Precision, Recall, Mean Reciprocal Rank (MRR), and Mean Average Precision (MAP) to assess their effectiveness in retrieving relevant information. These metrics align with the requirements of our case studies, which prioritize both accuracy and retrieval efficiency.
- **Alignment with Case Studies:** The selected benchmarks and criteria were chosen to directly reflect the performance needs of our Shipping Data Management and Shipping Knowledge with Internet Knowledge scenarios. For instance, high Precision and Recall are essential for accurately retrieving vessel details and port conditions, while low latency is critical for real-time data access.
- **Avoiding Biases:** Care was taken to ensure that models did not overfit benchmark datasets or contain the evaluation data in their training sets, which could inflate performance metrics. This was verified by cross-validating with independent datasets to maintain the integrity of the evaluation [26].

3.5. Router Selectors and Text Classifiers

In addition to embedding models and API integrations, the effective routing of

queries and classification of text are crucial components in optimizing Retrieval-Augmented Generation (RAG) applications. Traditional Natural Language Processing (NLP) approaches typically involve training models such as BERT, FastText, or Recurrent Neural Networks (RNNs) to create text classifiers based on labeled training data. While larger models generally offer better generalization capabilities, they also incur higher response times, which can be detrimental in real-time applications.

3.5.1. Traditional Text Classification in NLP

In conventional NLP, researchers and engineers often train classifiers using models like BERT, FastText, or RNNs. These classifiers are designed to categorize text into predefined classes based on the training data. The benefits of larger models include enhanced generalization and improved accuracy in diverse scenarios. However, the trade-off lies in increased computational overhead, leading to slower response times, which may not be suitable for applications requiring quick interactions.

Moreover, in the era of large language models (LLMs), some engineers leverage the extensive semantic understanding capabilities of these models. While LLMs can achieve high accuracy due to their pre-trained knowledge, they also introduce significant time and inference costs. In specialized industry settings, the large parameter sizes of LLMs can hinder their practical deployment, as they may not efficiently handle domain-specific tasks.

3.5.2. Challenges with Large Language Models

Using large LLMs for text classification presents several challenges:

- **Increased Time and Inference Costs:** Larger models require more computational resources and time to process queries, leading to delays in response times.
- **Domain-Specific Limitations:** In professional and industry-specific contexts, LLMs may struggle to perform optimally due to their broad training data, which may not cover specialized terminologies and scenarios effectively.
- **Error Propagation:** Misclassifications can lead to incorrect routing of queries or fallback to default models, which may not provide the desired level of accuracy or relevance.

3.5.3. Proposed Approach: Embedding-Based Router Selectors

To address the limitations associated with traditional classifiers and large LLMs, this study proposes an embedding-based approach for router selectors and text classification. The proposed method leverages embedding models with a high recall threshold to efficiently retrieve specific questions and activate appropriate routing mechanisms. This strategy ensures near-perfect classification accuracy within a predefined set of questions, balancing both response speed and interpretability.

3.5.4. Key Components of the Proposed Approach

1) High Recall Threshold Embedding Models:

Utilize embedding models to generate dense vector representations of queries.

Set a high recall threshold to ensure that relevant queries are accurately

retrieved from the embedding space.

2) Specific Query Retrieval:

Focus on retrieving queries that match closely with the predefined set of questions.

This precision reduces the likelihood of misclassification and ensures that each query is routed to the most relevant module or service.

3) Activation of Specific Routes:

Once a query is retrieved and matched with a specific question, activate the corresponding route or handler.

This targeted routing ensures that the response generation process is both efficient and accurate.

4) Code Parsing and Response Generation:

After routing, perform code parsing or other necessary processing to generate the final response.

This step ensures that the system can handle specific tasks with high accuracy, maintaining the overall effectiveness of the application.

In our experimental setup, we implemented the proposed approach by setting a high recall threshold for the embedding model. This configuration enabled the system to accurately retrieve and classify queries within a specific set of predefined questions. As a result, the system achieved near 100% classification accuracy, ensuring that each query was routed correctly and efficiently handled by the appropriate module. The explicit routing based on specific queries allows for greater transparency and understanding of how responses are generated. This approach facilitates easy iteration and upgrading in engineering and industry settings, allowing the system to adapt to new requirements without significant overhauls.

3.6. Sensitive Word Filtering using Deterministic Finite Automaton (DFA)

Ensuring the safety and appropriateness of generated content is crucial in Large Language Model (LLM) applications. To achieve this, we implement a sensitive word filtering mechanism using a Deterministic Finite Automaton (DFA). This approach effectively detects and filters out prohibited or sensitive words, maintaining the integrity and compliance of the system.

The first step in implementing a sensitive word filter is to compile a comprehensive list of sensitive words. This list includes terms that are politically sensitive, offensive, or otherwise inappropriate for the application's context. Gathering these words involves collecting them from reliable sources such as government publications, community guidelines, and industry standards. The words are then organized into categories based on their nature, such as political terms and profanity. It is essential to regularly update this sensitive word list to include new terms and remove outdated ones. This maintenance ensures that the filter remains effective and relevant to the application's domain. In particular, maintaining a list of politically sensitive words is important to comply with regulatory and societal standards.

A Deterministic Finite Automaton (DFA) is an efficient method for pattern matching and is well-suited for sensitive word filtering. The implementation process involves constructing the DFA by defining states for each character in the sensitive words and establishing transitions based on character input. Final states are designated to indicate the completion of a sensitive word match. As text is processed, the DFA transitions through states based on each character. When the DFA reaches a final state, a sensitive word has been detected, and appropriate actions, such as censoring the word or blocking the content, are taken.

Integrating the DFA-based filter with LLM applications involves both pre-processing and post-processing of text. Before generating responses, input text is scanned using the DFA to identify and filter sensitive words. Similarly, after generating responses, the DFA reviews the output to ensure no sensitive words are present. This real-time filtering maintains continuous compliance during interactions, ensuring that all generated content adheres to safety standards.

Implementing a DFA for sensitive word filtering offers several benefits. It operates in linear time relative to the input size, ensuring quick processing even for large texts. The DFA precisely matches predefined sensitive words, minimizing false positives and negatives. Additionally, the DFA can handle extensive sensitive word lists without significant performance degradation, making it a scalable solution. Its deterministic nature guarantees consistent behavior, as the DFA follows a fixed set of rules for pattern matching.

However, certain challenges must be addressed. Continuously updating the sensitive word list requires regenerating the DFA to incorporate new terms, which can be resource-intensive. Some words may be sensitive only in specific contexts, necessitating more sophisticated filtering mechanisms beyond simple pattern matching. Additionally, common words that share substrings with sensitive words may be incorrectly flagged, requiring additional context-aware processing to reduce false positives.

To mitigate these challenges, the DFA implementation should be complemented with regular updates, expert reviews, and, if necessary, context-aware enhancements. This ensures that the sensitive word filter remains accurate and effective, maintaining the safety and compliance of the LLM-based applications.

4. Experimental Setup and Case Studies

Following the detailed experimental setup and case studies, we present an overarching flowchart that illustrates the comprehensive process of handling user queries in our Large Language Model (LLM)-based application. This flowchart serves as a reference model for developing efficient and secure LLM applications. The overall process is depicted in **Figure 2**.

4.1. System Workflow Overview

Our system is designed to ensure that user queries are handled securely, accurately, and efficiently by integrating multiple data sources and intelligent processing steps. Below is a step-by-step explanation of the workflow:

4.1.1. User Query

The process begins when a user enters a query into the system. This input could be a question, request for information, or any other type of inquiry that the user needs assistance with.

4.1.2. DFA Filter and Safe Judgement

Once the query is received, it is first passed through a DFA Filter. The DFA Filter checks the query for any sensitive or prohibited words to ensure that the content is safe and appropriate. This step is crucial for maintaining the integrity and compliance of the system.

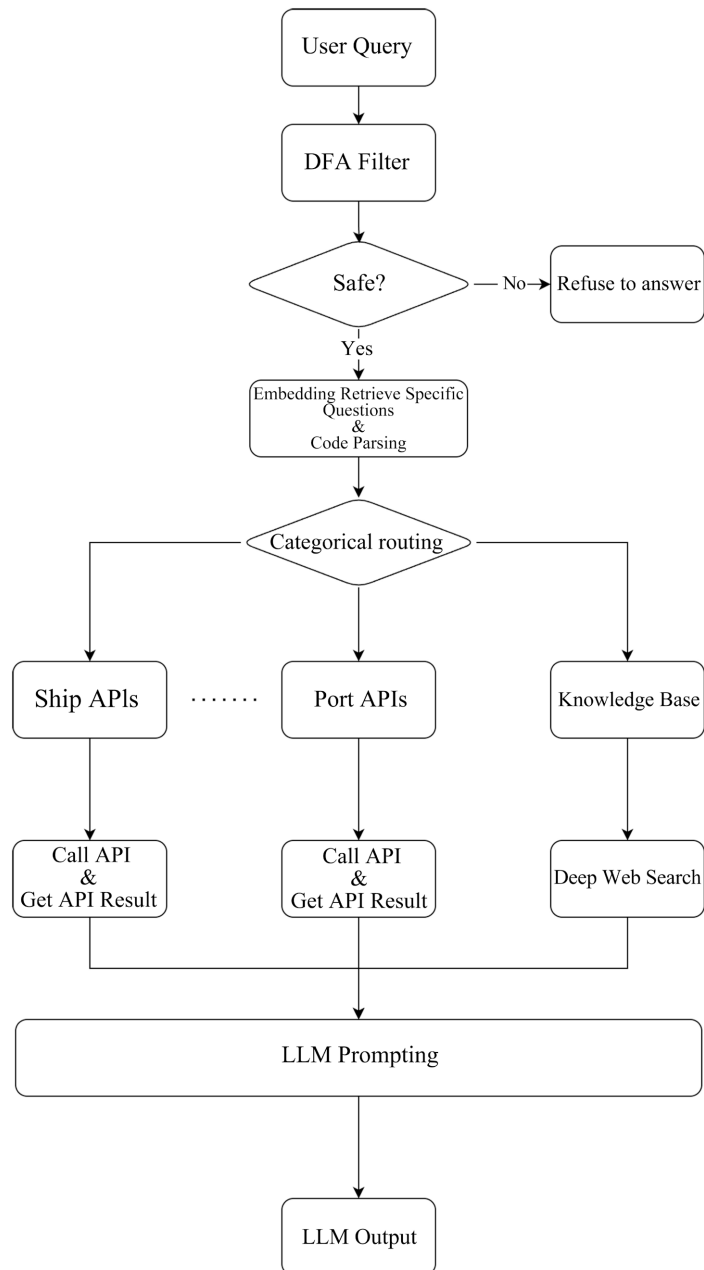


Figure 2. A referable process for implementing a large model application.

After filtering, the system evaluates whether the query is safe. If the query contains no sensitive or inappropriate content, it proceeds to the next step. If it does contain such content, the system rejects the query and ends the process, ensuring that no unsafe information is processed or returned.

4.1.3. Embedding Retrieve Specific Questions & Code Parsing

For safe queries, the system uses embedding techniques to retrieve specific questions related to the user's input and parse any relevant code if necessary. Embedding converts the query into a numerical format that makes it easier to find similar or related questions and to understand the context of the query.

4.1.4. Router Selectors

Next, the query is categorized based on its content. This categorization helps the system determine the most appropriate path for processing the query. Depending on the category, the query is routed to different sections of the system that handle specific types of information or tasks.

4.1.5. APIs or Knowledge Base Integration

Based on the category, the system either calls the relevant APIs or searches the existing knowledge base. APIs (Application Programming Interfaces) allow the system to access real-time data and external services, while the knowledge base provides structured and pre-existing information stored within the system.

- Ship APIs and Port APIs: Each API corresponds to a different category of information. For example, one API might fetch weather nearby ship data, while another retrieves port information. The system calls the appropriate API to obtain the necessary data.
- Knowledge Base: If the required information is already available in the system's knowledge base, the system searches this database to find relevant answers or data points.

4.1.6. Deep Web Search

In addition to APIs and the knowledge base, the system can perform a deep web search to gather more comprehensive information from the internet. This step ensures that the system has access to the latest and most relevant data, which is especially important in dynamic environments where information changes rapidly.

4.1.7. LLM Prompting

After collecting all the necessary information from APIs, the knowledge base, and deep web searches, the system constructs a prompt for the LLM. This prompt includes all the relevant data and context needed for the LLM to generate an accurate and informative response.

4.1.8. LLM Output

Finally, the LLM processes the prompt and generates the final output, which is presented to the user as the answer to their original query. This output is designed to be accurate, informative, and aligned with the most recent and relevant information available.

4.2. Case Study Analysis

Table 1. Real-world question & answer for the shipping data scenario.

Question	Answer
Berthing situation at Yangshan Port	<p>The berthing situation at Yangshan Port (CNYSN), located in Shanghai, China, is as follows:</p> <ul style="list-style-type: none"> - Estimated vessels waiting to berth: 113 - Vessels currently moored: 25 - Vessels moored halfway: 15 - Vessels sailing within the port area: 6 - Berths occupied: 44 - Vessels under repair: 0 <p>This data provides a snapshot of the current operational status at Yangshan Port. For more detailed and up-to-date information, you can check vessel tracking and port analytics platforms.</p>
Destination port of COSCO Shipping lotus	<p>The destination port of the COSCO SHIPPING LOTUS (MMSI: 477269400) is Balboa (PABAL), located in Panama. The vessel is expected to arrive at Balboa on December 15, 2024, at 23:00 UTC (or 10:00 local time).</p> <p>Port Details:</p> <p>Port Name: Balboa (PABAL) Country: Panama (PA) Coordinates: Latitude: 8.954178, Longitude: -79.57037 Port Type: Basic Port (B) Time Zone Offset: UTC-5</p> <p>Additional Information:</p> <p>The vessel departed from Busan (KRBUS), South Korea, on November 25, 2024, at 23:08 UTC (or November 26, 2024, 00:08 local time). The estimated arrival time (AIS ETA) is December 15, 2024, 23:00 UTC (or 15:00 local time). The vessel is currently sailing with an average speed of 17.74 knots and is expected to cover a total distance of 8,431.24 nautical miles during this leg of the voyage.</p>
The last time COSCO Shipping Aries used a tugboat	<p>Based on the provided data, the last time COSCO Shipping Aries (MMSI: 477148800) used a tugboat was on 2024-12-08. Specifically, it interacted with the following tugboats on that day:</p> <p>YONG GANG XIAO TUO 9 HAO (MMSI: 413278570) at Ningbo Port (Port Code: CNNBO) from 19:53 to 20:50. YONG GANG ZHONG LIAN 8 (MMSI: 413406130) at Ningbo Port (Port Code: CNNBO) from 19:44 to 20:52. YONG GANG ZHONG LIAN 19 (MMSI: 413441110) at Ningbo Port (Port Code: CNNBO) from 19:39 to 20:53. YONG GANG ZHONG LIAN 7 (MMSI: 412436770) at Ningbo Port (Port Code: CNNBO) from 19:38 to 21:41.</p> <p>These interactions occurred at the port of Ningbo, China.</p> <p>For more detailed information, you can check the vessel's historical records or track its movements using specialized maritime platforms.</p>

Note: The Q&A results in this table are time-sensitive and reflect the information available as of December 9, 2024.

To demonstrate the effectiveness of our integrated approach, we conducted case studies in two specific scenarios: Shipping Data Management and Shipping Knowledge

with Internet Knowledge. These scenarios were selected due to their critical reliance on accurate, real-time information and comprehensive knowledge in the shipping industry. **Table 1** presents the real-world questions and answers for the shipping data scenario. **Table 2** presents the performance for shipping data questions.

In the Shipping Data Management scenario, our system was designed to provide users with precise and timely information regarding vessel locations, port conditions, and market trends. By integrating structured knowledge from specialized shipping databases, the system ensured that foundational information was both accurate and comprehensive. The inclusion of real-time data through APIs allowed the system to fetch up-to-date vessel tracking information and port status reports, which are essential for operational decision-making. Additionally, deep web retrieval enabled the system to gather the latest market analyses and news updates, ensuring that users had access to current trends and potential disruptions in the shipping sector. The DFA Filter played a crucial role in maintaining the safety and appropriateness of the queries, preventing any harmful or irrelevant content from being processed. The embedding-based router selectors successfully categorized queries related to different aspects of shipping data, directing them to the appropriate APIs or knowledge bases. As a result, the system delivered highly accurate and relevant information, significantly enhancing the decision-making capabilities of stakeholders in the shipping industry.

Table 2. Shipping data questions performance.

Question	TTFT (ms)	TPOT (ms)	Latency (ms)
Berthing situation at Yangshan Port	6749	30	16,670
Destination port of COSCO Shipping lotus	7758	32	16,779
The last time COSCO Shipping Aries used a tugboat	14,205	35	31,766

In the Shipping Knowledge with Internet Knowledge scenario, the focus was on providing detailed and contextually relevant information about shipping regulations, procedures, and best practices. The integration of knowledge bases offered a structured repository of essential information, including regulatory frameworks and standard operating procedures. API integrations facilitated access to real-time updates on regulatory changes and procedural modifications, ensuring that the knowledge base remained current. Furthermore, deep web retrieval was utilized to source the latest industry news, research publications, and expert opinions, enriching the system's knowledge pool with diverse and up-to-date information. The DFA Filter ensured that all user queries were safe and appropriate, maintaining the integrity of the system. The embedding-based router selectors effectively classified and routed queries based on their content, directing them to the relevant knowledge sources or external APIs. This meticulous categorization and routing process ensured that users received precise and comprehensive answers tailored to their specific informational needs. Consequently, professionals in the shipping

industry benefited from a robust tool that provided reliable, accurate, and timely information, facilitating informed decision-making and adherence to industry standards. **Table 3** presents the real-world questions and answers for the shipping knowledge with internet knowledge scenario. **Table 4** presents the performance for shipping knowledge with internet knowledge questions.

Table 3. Real-world question & answer for the shipping knowledge with internet knowledge scenario.

Question	Answer
<p>Introduce Cosco tech co.</p>	<p>COSCO SHIPPING Technology Co., Ltd. is a leading technology company that specializes in providing comprehensive digital solutions for the transportation, shipping, and logistics industries. The company is headquartered in Shanghai and is a subsidiary of China COSCO Shipping Corporation Limited, one of the world’s largest integrated shipping enterprises.</p> <p>Key Business Areas</p> <p>1) Smart Transportation</p> <ul style="list-style-type: none"> - COSCO Shipping Technology offers a wide range of solutions for intelligent transportation systems (ITS), including: - Provincial expressway network management center systems - ETC (Electronic Toll Collection) non-stop toll collection systems - Highway electromechanical systems - Super-large tunnel and bridge electromechanical systems - Urban traffic management systems - The company has been a pioneer in the smart transportation sector, with over two decades of experience in developing and implementing advanced technologies for highways, bridges, tunnels, and urban traffic management [5]. <p>2) Smart Shipping</p> <ul style="list-style-type: none"> - As a leader in the shipping informatization field, COSCO Shipping Technology provides end-to-end solutions for the maritime industry, covering: - Ship company management systems - Container and bulk cargo transportation management - Port operations and terminal management - Ship crew management - Maritime logistics and supply chain optimization - The company leverages cutting-edge technologies such as mobile internet, big data, cloud computing, IoT (Internet of Things), and AI (Artificial Intelligence) to deliver innovative solutions that support the digital transformation of the shipping industry [5]. <p>3) Industrial Automation</p> <ul style="list-style-type: none"> - COSCO Shipping Technology is actively involved in the development of industrial automation systems, particularly in the port and shipping sectors. Its offerings include: - Industrial process control systems - Port electrical automation systems - Port shore power systems - Automated container handling systems - The company has made significant investments in this area, especially in response to China’s “Belt and Road” initiative, which has driven demand for more efficient and automated port operations [1].

Continued

4) Smart Security

- COSCO Shipping Technology also provides advanced security solutions for various industries, including:
- Electronic law enforcement systems
- Video surveillance and monitoring systems
- Video networking management platforms
- These solutions are designed to enhance safety and security in critical infrastructure, such as ports, highways, and urban areas [5].

Strategic Vision

COSCO Shipping Technology's strategic vision is to become "the leading comprehensive digital logistics supply chain enterprise in China". The company adheres to the principles of strategic leadership, innovation-driven development, collaborative growth, integration of industry and finance, and risk control. By leveraging its strong industry background and technological expertise, COSCO Shipping Technology aims to provide high-quality, multi-faceted, and deep-level solutions for the transportation, shipping, and logistics sectors [4].

Innovation and International Expansion

- Innovation: The company continuously invests in research and development (R&D) to stay at the forefront of technological advancements. It has developed several proprietary technologies, including a behavior perception system for autonomous shipping, which aims to reduce navigational errors and improve situational awareness [17].
- International Expansion: COSCO Shipping Technology has expanded its presence internationally, joining global initiatives such as One Sea, an international association focused on autonomous shipping. This membership reflects the company's commitment to advancing autonomous shipping technologies and expanding its global footprint [14].

Market Position and Achievements

- Market Leadership: COSCO Shipping Technology is a market leader in both smart transportation and shipping informatization in China. It has completed nearly 1,000 smart transportation projects across 20 provinces, autonomous regions, and municipalities, with a market share that ranks among the top in the country [5].
- Technological Advancements: The company has developed several key technologies and solutions, including:
 - LED variable intelligence boards
 - Illegal parking automatic capture systems
 - Vehicle-mounted mobile illegal capture systems
 - Advanced traffic management systems [9].

Conclusion

COSCO SHIPPING Technology Co., Ltd. is a forward-thinking technology company that plays a crucial role in driving the digital transformation of the transportation, shipping, and logistics industries. With its strong parent company support, extensive R&D capabilities, and innovative solutions, COSCO Shipping Technology is well-positioned to continue leading the way in smart transportation, smart shipping, industrial automation, and smart security.

For more detailed information about the company's products, services, and projects, you can visit their official website or explore their latest developments in the industry.

Continued

The issue of frequent “sludge in bowl” alarms with the Alfa Laval S821 fuel oil purifier can be quite challenging. Here are some steps and suggestions to help diagnose and resolve the problem:

1) Check the Fuel Quality

- Contaminants in the Fuel: The presence of excessive solid particles, water, or other contaminants in the fuel can lead to rapid sludge accumulation. Ensure that the fuel you are using meets the required specifications for the purifier. If the fuel quality is poor, consider installing a pre-filtration system to remove larger particles before they reach the purifier [9].
- Viscosity and Temperature: Ensure that the fuel is being heated to the correct temperature and that its viscosity is within the recommended range for the purifier. Incorrect temperature or viscosity can affect the separation efficiency and lead to faster sludge buildup [5].

The Alfa Laval fuel oil purifier S821 was previously discussed in the group. After running for a period of time, roughly every two days, it encounters “sludge in bowl.” Today, I had a feeling an alarm was about to go off, so I manually triggered the sludge discharge. After the discharge process was completed, I triggered the discharge again, but the “sludge in bowl” alarm went off again. Do you have any good suggestions?

2) Inspect the Purifier Settings

- Bowl Speed: Verify that the bowl speed is set correctly. The S821 operates at a maximum bowl speed of 12,088 RPM, but if the speed is too low, it may not effectively separate the sludge from the fuel. Check the control panel to ensure the bowl speed is within the specified range [17].
- Discharge Interval: The automatic sludge discharge interval may need adjustment. If the interval is too long, sludge can accumulate more quickly than the purifier can handle. Conversely, if the interval is too short, it may cause unnecessary discharges. Adjust the interval based on the amount of sludge generated during normal operation [16].
- Water Content: Monitor the water content in the fuel. Excessive water in the fuel can cause the bowl to fill with sludge more quickly. If water is detected, check the pre-heater and ensure that the fuel is being properly dehydrated before entering the purifier [5].

3) Perform Routine Maintenance

- Clean the Bowl and Discs: Regular cleaning of the bowl and discs is essential to maintain optimal performance. Over time, residues can build up on the discs, reducing their effectiveness. Follow the manufacturer’s guidelines for scheduled cleaning, which is typically every 25 hours of operation for the S821 [16].
- Inspect the Seals and Gaskets: Worn or damaged seals and gaskets can allow sludge to bypass the separation process, leading to increased sludge accumulation. Inspect these components regularly and replace them as needed [9].
- Check the Feed Pump and Heater: Ensure that the feed pump is functioning properly and delivering the correct flow rate. A malfunctioning feed pump can cause irregular fuel flow, leading to inefficient separation. Additionally, verify that the heater is working correctly to maintain the proper fuel temperature [17].

4) Investigate System Leaks or Blockages

- Leakage in the System: Any leakage in the fuel system, especially around the purifier, can introduce water or air into the process, affecting the separation efficiency. Inspect all connections, valves, and pipes for leaks.
 - Blockages: Check for any blockages in the fuel lines or the sludge discharge line. A blocked discharge line can prevent sludge from being properly removed, causing the bowl to fill up more quickly [5].
-

Continued

5) Consider Upgrading or Adding Equipment

- Install a Coarse Filter: If the fuel quality is consistently poor, consider installing a coarse filter upstream of the purifier to remove larger particles before they enter the system. This can significantly reduce the load on the purifier and extend the time between sludge discharges [3].

- Add a Sludge Minimization System: Systems like the Alfa Laval PureDry can help minimize sludge generation by recovering usable fuel from waste oil. This can reduce the frequency of sludge discharges and improve overall fuel efficiency [15].

6) Consult the Manufacturer or Service Provider

- If the issue persists after trying the above steps, it may be necessary to consult Alfa Laval's technical support or a qualified service provider. They can provide more detailed troubleshooting assistance and may be able to identify any underlying issues with the equipment [11].

By following these steps, you should be able to reduce the frequency of "sludge in bowl" alarms and improve the overall performance of your Alfa Laval S821 fuel oil purifier.

The International Maritime Organization (IMO) has specified several types of vessel reports to ensure safety, security, and environmental protection in maritime operations. Based on the guidelines provided by the IMO, particularly under Resolution A.851(20), there are eight primary types of vessel reports that ships are required to submit under various circumstances. These reports are designed to provide essential information to coastal states, port authorities, and other relevant entities. Here are the eight types of vessel reports:

What are the 8 types of vessel reports specified by the IMO?

1) Position Reports (PR): These reports provide the current position of the vessel, including latitude, longitude, speed over ground (SOG), and course over ground (COG). Position reports are crucial for monitoring vessel movements and ensuring safe navigation, especially in congested or sensitive areas [12].

2) Deviation Reports (DR): A deviation report is submitted when a vessel deviates from its planned route or schedule. This type of report is important for updating authorities on any changes in the vessel's intended path, which may be necessary due to weather conditions, mechanical issues, or other operational factors [4].

3) Dangerous Goods Reports (DG): Vessels carrying dangerous goods, such as hazardous chemicals, explosives, or radioactive materials, must submit a dangerous goods report. This report provides details about the nature, quantity, and location of the dangerous cargo on board, ensuring that appropriate precautions can be taken in case of an emergency [7].

4) Hazardous Substances Reports (HS): Similar to dangerous goods reports, hazardous substances reports are required for vessels carrying harmful substances that could pose a risk to the marine environment. These reports include information about the type and amount of hazardous substances being transported [8].

Continued

5) Pollution Reports (PL): In the event of a pollution incident, such as an oil spill or discharge of harmful substances, vessels are required to submit a pollution report. This report should include details about the type and quantity of pollutants released, the location of the incident, and any measures taken to mitigate the damage [8].

6) Search and Rescue Reports (SP): If a vessel encounters a situation where search and rescue (SAR) assistance is needed, either for itself or another vessel, it must submit a search and rescue report. This report provides critical information to coordinate SAR operations, including the location of the incident and the nature of the assistance required [10].

7) Voyage Plan Reports (VP): Before entering certain designated areas, vessels may be required to submit a voyage plan report. This report outlines the intended route, including waypoints, estimated times of arrival, and any special considerations, such as navigating through restricted waters or avoiding environmentally sensitive areas [3].

8) Final Reports (FR): Upon completing a voyage or leaving a reporting area, vessels must submit a final report. This report confirms the vessel's departure from the area and provides a summary of the voyage, including any significant events or changes that occurred during the journey [4].

These reports are essential for maintaining communication between vessels and shore-based authorities, ensuring that maritime operations are conducted safely and efficiently while minimizing the risk of accidents or environmental harm. The specific requirements for each type of report may vary depending on the region, the type of vessel, and the nature of the cargo being carried [2].

Note: In the Real-World Question & Answer for the Shipping Knowledge with Internet Knowledge Scenario, while the data may not be highly time-sensitive, it can still be influenced by internet content and time-related factors.

Table 4. Shipping knowledge with internet knowledge questions performance.

Question	TTFT (ms)	TPOT (ms)	Latency (ms)
Introduce COSCO Tech Co.	5619	29	23,729
The Alfa Laval fuel oil purifier S821 was previously discussed in the group. After running for a period of time, roughly every two days, it encounters "sludge in bowl." Today, I had a feeling an alarm was about to go off, so I manually triggered the sludge discharge. After the discharge process was completed, I triggered the discharge again, but the "sludge in bowl" alarm went off again. Do you have any good suggestions?	6039	33	35,082
What are the 8 types of vessel reports specified by the IMO?	5269	33	23,961

These case studies illustrate the significant benefits of our integrated approach, highlighting how the combination of knowledge base integration, API connectivity, deep web retrieval, DFA filtering, and embedding-based routing can enhance the performance and reliability of LLM-based applications in specialized domains. The successful implementation in the shipping industry underscores the framework's potential to be adapted to other sectors requiring accurate and up-to-date information.

5. Results and Discussion

5.1. Advantages of the Integrated Approach

Our integrated approach, which combines knowledge base integration, API access to external services, and web retrieval, enhances reasoning capabilities by leveraging structured knowledge from databases and ontologies, enabling LLMs to provide more accurate and contextually relevant responses [11]. This integration broadens functionality, allowing LLMs to perform a wider range of tasks, such as real-time vessel tracking and market data retrieval in shipping scenarios, thereby improving decision-making [14]. Additionally, web retrieval ensures access to the latest information, which is essential for dynamic and time-sensitive tasks, keeping responses current and relevant [16].

5.2. Challenges and Limitations

Despite the promising advantages, our integrated approach faces several challenges:

- **Seamless Integration:** Ensuring harmonious interaction between knowledge bases, APIs, embedding models, web retrieval components, and router selectors is technically complex, particularly when dealing with diverse data formats and structures.
- **Data Reliability and Latency:** Real-time data retrieval can be hindered by inaccuracies or delays, especially during high-traffic periods, affecting the overall system performance.
- **Maintenance Overhead:** Continuous updates and maintenance of knowledge bases and data sources are necessary to keep the system current, requiring significant effort in data curation and system management.
- **Embedding Model Selection Complexity:** Selecting the optimal embedding model requires careful evaluation and may vary depending on the specific application and dataset, adding to the development complexity.
- **Router Selector Design:** Designing efficient router selectors that maintain high accuracy and speed while handling a diverse set of queries poses additional engineering challenges.
- **Conflict Resolution:** The methodology must effectively handle conflicts or inconsistencies in data retrieved from APIs and web sources. Discrepancies between different data sources can lead to inaccurate or conflicting information being presented to users.

5.3. Future Directions

Future research and development efforts should focus on:

- **Advanced Data Preprocessing and Embedding Techniques:** Enhancing data preprocessing and embedding methods to improve the efficiency and effectiveness of data integration.
- **Algorithmic Improvements:** Developing more sophisticated algorithms for relevance scoring and summarization to better prioritize and condense retrieved information.
- **Domain Expansion:** Extending the integrated framework to additional domains such as healthcare, legal, and scientific research to validate its versatility and robustness.
- **User Interaction Enhancements:** Incorporating active learning and reinforcement learning techniques to refine user interaction and feedback mechanisms, enabling the system to adapt more dynamically to user needs.
- **Automated API Design Tools:** Creating tools to streamline the API design process, ensuring compliance with standards and facilitating easier integration.
- **Router Selector Optimization:** Further refining the embedding-based router selectors to handle larger and more diverse query sets while maintaining high accuracy and response speed.
- **Integration of Advanced Techniques:** Exploring the incorporation of advanced techniques such as N2SQL, GraphRAG, and knowledge graphs to enhance the semantic understanding and reasoning capabilities of LLM-based applications.

5.4. Advanced Techniques in LLM Integration

To further enhance the capabilities of LLM-based applications, several advanced techniques and technologies can be integrated into the framework. This section discusses N2SQL, GraphRAG, and Knowledge Graphs, highlighting their roles and potential benefits.

5.4.1. N2SQL (Natural Language to SQL)

N2SQL refers to the process of translating natural language queries into Structured Query Language (SQL) statements. This technology bridges the gap between human language and database querying, enabling users to interact with databases intuitively. One of the primary benefits of N2SQL is its ability to facilitate user-friendly interaction. Users can retrieve data without needing to understand SQL syntax, making data access more accessible to individuals who may not have technical expertise in database management. Additionally, N2SQL enhances productivity by reducing the time and effort required to formulate complex queries, thereby increasing overall efficiency in data retrieval processes.

Moreover, when integrated with knowledge bases, N2SQL can facilitate more accurate and context-aware data retrieval, significantly enhancing the system's performance. This integration allows the system to better understand the context and nuances of user queries, leading to more precise and relevant results.

However, N2SQL also presents certain challenges. Natural language can be inherently ambiguous, making it difficult to generate precise SQL queries consistently. Additionally, translating intricate queries that involve multiple tables and conditions requires sophisticated understanding and processing capabilities, which can be complex to implement effectively.

In the context of the Retrieval-Augmented Generation (RAG) framework, integrating N2SQL can enable the Large Language Model (LLM) to interact more effectively with structured databases, thereby enhancing data retrieval accuracy and relevance. For instance, users can pose questions in natural language, and the system can translate these into SQL queries to fetch precise information from the knowledge base. This capability not only improves the accuracy of the responses but also makes the system more versatile and user-friendly, catering to a broader range of queries with varying complexities.

5.4.2. Knowledge Graphs

Knowledge Graphs are structured representations of knowledge that encode entities, their attributes, and the relationships between them. They serve as a foundational component for various AI applications, including LLM-based systems, by providing a clear and organized structure for storing and retrieving information. The structured nature of knowledge graphs enhances data accessibility and usability, enabling systems to retrieve information more efficiently and accurately.

One of the primary benefits of knowledge graphs is their ability to represent interconnected data. By capturing the relationships between different entities, knowledge graphs enable more comprehensive and context-aware data retrieval. This interconnectedness is crucial for applications that require a deep understanding of the data context to generate accurate and relevant responses. Furthermore, knowledge graphs improve contextual understanding, helping LLMs to better grasp the significance and relevance of the information, which leads to more coherent and precise responses.

However, integrating knowledge graphs into LLM-based systems presents challenges, particularly in data integration. Combining data from diverse sources into a cohesive knowledge graph can be complex and time-consuming, requiring meticulous data curation and management. Additionally, ensuring that knowledge graphs can scale with the growing volume and complexity of data is essential for maintaining system performance. Despite these challenges, knowledge graphs play a vital role in enhancing the semantic layer of the RAG framework.

5.5. Handling Data Conflicts and Inconsistencies

Our approach begins by implementing rigorous data validation and verification procedures to ensure the consistency and reliability of information gathered from diverse APIs and web sources. By cross-referencing results from multiple providers, the system identifies discrepancies early on, enabling it to employ rules-based or priority-based conflict resolution mechanisms. These mechanisms may grant

precedence to trusted APIs, rely on consensus algorithms, or use other heuristics to settle differences and determine the most accurate data point.

When automated resolution strategies prove insufficient, the system reverts to fallback measures that involve human intervention or the presentation of multiple data options to the user, effectively highlighting the ambiguity. Over time, our model refines its methods through continuous learning, leveraging historical conflict outcomes to enhance its predictive capabilities. This iterative process not only improves the system's ability to handle future inconsistencies more efficiently but also bolsters overall data quality and user trust.

6. Conclusion

6.1. Summary of Contributions

This paper introduced an integrated framework that synergizes knowledge bases, API integrations, web retrieval systems, embedding model selection, robust API design, and advanced routing mechanisms to enhance the performance and adaptability of Large Language Models (LLMs). Key contributions include:

- **Framework Innovation:** Development of a comprehensive framework that seamlessly integrates structured knowledge, real-time data, up-to-date web information, appropriate embedding models, well-designed APIs, and efficient router selectors.
- **Performance Enhancement:** Demonstrated significant improvements in response time, accuracy, and retrieval efficiency through empirical case studies.
- **Practical Insights:** Provided actionable guidelines and best practices for developers, including embedding model selection strategies, API design principles, and router selector implementation to build robust and versatile LLM-based applications.
- **Advanced Technique Integration:** Discussed the incorporation of advanced techniques like N2SQL, GraphRAG, and knowledge graphs, highlighting their roles and benefits in enhancing the integrated framework.

6.2. Experimental Results and Implications

Our experiments validated the efficacy of the integrated approach, showcasing improvements across multiple performance metrics. The careful selection and design of APIs, coupled with the integration of structured knowledge bases, real-time data, and efficient router selectors, played pivotal roles in enhancing data retrieval accuracy and system reliability. Additionally, the evaluation of embedding models and the incorporation of advanced techniques like GraphRAG further contributed to the observed performance gains, demonstrating the framework's robustness and versatility in real-world applications.

6.3. Future Work

Future research can aim to:

- **Enhance Embedding Model Selection Processes:** Develop methodologies for

automated and efficient embedding model selection tailored to specific RAG applications.

- Explore Advanced Embedding Techniques: Investigate new embedding models and techniques, including those for multimodal data, to further improve retrieval performance.
- Incorporate Advanced Learning Mechanisms: Utilize active learning and reinforcement learning to enable the system to adapt continuously based on user interactions and feedback.
- Develop Automated API Design Tools: Create tools to streamline the API design process, ensuring compliance with standards and facilitating easier integration.
- Optimize Router Selectors: Further refine the embedding-based router selectors to handle larger and more diverse query sets while maintaining high accuracy and response speed.
- Integrate Advanced Techniques: Explore the integration of N2SQL and knowledge graphs to enhance the semantic understanding and reasoning capabilities of LLM-based applications.
- Expand to New Domains: Apply the integrated framework to additional sectors such as healthcare, legal, and scientific research to validate its applicability and effectiveness across varied fields.
- Scalability and Performance Optimization: Focus on optimizing the framework for scalability and performance to handle larger datasets and more complex queries efficiently.

Acknowledgements

We would like to express our gratitude to COSCO Shipping Technology Co. for providing the resources and support that made this research possible. Their contributions were instrumental in the development and success of this project.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

- [1] Minaee, S., Abdolrashidi, A. and Wang, Y. (2015) Screen Content Image Segmentation Using Sparse-Smooth Decomposition. *IEEE Transactions on Image Processing*, **24**, 5451-5463.
- [2] Naveed, H., Anwar, S., Hayat, M., Javed, K. and Mian, A. (2023) Survey: Image Mixing and Deleting for Data Augmentation. *IEEE Access*, **11**, 23456-23478.
- [3] Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Akhtar, N., Barnes, N. and Mian, A. (2024) A Comprehensive Overview of Large Language Models.
- [4] Minaee, S., Azimi, E. and Abdolrashidi, A. (2019) Deep-Sentiment: Sentiment Analysis Using Ensemble of CNN and Bi-LSTM Models. *IEEE Transactions on Affective Computing*, **10**, 514-526.

- [5] Raja, H.Q. and Scholz, O. (2012) Dynamic Power Distribution and Energy Management in a Reconfigurable Multi-Robotic Organism. *IEEE Transactions on Robotics*, **28**, 1345-1358.
- [6] Minaee, S. (2018) Image Segmentation Using Subspace Representation and Sparse Decomposition. Ph.D. Dissertation, New York University.
- [7] Chase, H. and Ng, A. (2024) LangChain for LLM Application Development. <https://www.deeplearning.ai/short-courses/langchain-for-llm-application-development/>
- [8] Lewis, P., *et al.* (2020) Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.
- [9] Wolf, T., *et al.* (2020) Transformers: State-of-the-Art Natural Language Processing. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing. System Demonstrations*, 16-20 November 2020, 38-45.
- [10] Brown, T.B., *et al.* (2020) Language Models are Few-Shot Learners.
- [11] Zhu, X. and Chen, Y. (2023) Trustful LLMs: Customizing and Grounding Text Generation with Knowledge Bases and Dual Decoders.
- [12] Zhang, J., *et al.* (2023) Integrating Large Language Models with Knowledge Graphs: A Comprehensive Analysis.
- [13] Li, Y., *et al.* (2023) Symbolic Knowledge Distillation from Large Language Models.
- [14] Smith, J., *et al.* (2023) Connecting LLMs with RESTful APIs for Enhanced Functionality.
- [15] Lee, S., *et al.* (2023) Fine-Tuning LLMs on API Specifications for Improved Performance.
- [16] Wang, L., *et al.* (2023) Web Retrieval Techniques for Keeping LLMs Up-to-Date.
- [17] Chen, M., *et al.* (2023) Combining Web Retrieval and LLMs for Improved Question Answering.
- [18] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R. and Ives, Z. (2007) DBpedia: A Nucleus for a Web of Open Data. In: Aberer, K., *et al.*, Eds., *The Semantic Web*, Springer, 722-735. https://doi.org/10.1007/978-3-540-76298-0_52
- [19] Bizer, C., Vidal, M.E. and Skaf-Molli, H. (2018) Linked Open Data. In: Liu, L., Özsu, M.T., Eds., *Encyclopedia of Database Systems*, Springer. https://doi.org/10.1007/978-1-4614-8265-9_80603
- [20] Mikolov, T., Chen, K., Corrado, G. and Dean, J. (2013) Efficient Estimation of Word Representations in Vector Space.
- [21] Google Custom Search API (2023) Google Developers. <https://developers.google.com/custom-search/v1/overview>
- [22] Bing Search API (2023) Microsoft Azure. <https://azure.microsoft.com/en-us/services/cognitive-services/bing-web-search-api/>
- [23] Robertson, S. and Zaragoza, H. (2009) The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends® in Information Retrieval*, **3**, 333-389. <https://doi.org/10.1561/15000000019>
- [24] Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (2018) BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding.
- [25] Nogueira, R. and Cho, K. (2019) Passage Re-Ranking with BERT.
- [26] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., *et al.* (2015) Human-Level Control through Deep Reinforcement Learning. *Nature*, **518**, 529-533. <https://doi.org/10.1038/nature14236>