

Article

Role-Aware Agentic Retrieval-Augmented Generation for Urban Rail Transit Fault Management

Wei Zhang ¹, Hui Fang ^{1,2}, Jiajun Bu ¹, Qiming Zhong ^{1,2,*} and Yueyao Yu ^{1,2,*}

¹ College of Computer Science and Technology, Zhejiang University, Hangzhou 310058, China; cstzhangwei@zju.edu.cn (W.Z.); fangh@nbmetro.com (H.F.); bj@zju.edu.cn (J.B.)

² Ningbo Rail Transit Group Co., Ltd., Ningbo 315101, China

* Correspondence: qimingzhong@zju.edu.cn (Q.Z.); yueyaoyu@zju.edu.cn (Y.Y.)

Abstract

Urban rail transit systems require reliable decision-support methods to assist operators in handling complex fault events. Although large language models provide strong reasoning and knowledge integration capabilities, their open-ended generation may produce responses that are difficult to align with operational procedures. To address this issue, we propose a role-aware agentic retrieval-augmented generation framework for transit fault management. The framework integrates keyword-based retrieval, role-aware multi-agent reasoning, and constrained action generation within a unified architecture. Operational procedures and historical cases retrieved from transit knowledge repositories provide evidence for agent reasoning, while constrained outputs ensure that generated instructions remain consistent with predefined operational actions. Experimental results on a real-world urban rail transit emergency exercise dataset demonstrate that the proposed framework improves the semantic score from 0.893 to 0.965 and increases Role F1 score from 0.824 to 0.915 compared with an Agentic Retrieval-Augmented Generation (RAG) baseline while maintaining comparable inference efficiency. These results demonstrate the potential of the proposed framework for decision support in urban rail transit fault scenarios.

Keywords: agentic retrieval-augmented generation; multi-agent systems; large language models; urban rail transit; fault management

1. Introduction

Urban rail transit (URT) systems operate under complex operational conditions, where equipment failures, signaling abnormalities, and other unexpected events require timely and reliable fault handling decisions [1]. Conventional automated decision-support approaches mainly rely on predefined rules or task-specific models [2], which are effective for well-defined fault conditions but may have limited adaptability when handling complex events that require contextual understanding.

Recently, large language models (LLMs) have shown strong potential in intelligent transportation applications due to their capabilities in language understanding, knowledge integration, and reasoning [3,4]. These capabilities provide new opportunities for assisting fault analysis and operational decision making. However, directly applying LLMs to URT operations remains challenging because their open-ended generation may produce unsupported recommendations that deviate from established operational procedures [5,6]. Therefore, reliable deployment of LLMs in URT fault management requires not



Academic Editor: Olufemi A. Omitaomu

Received: 22 July 2026

Revised: 17 August 2026

Accepted: 28 August 2026

Published: 31 August 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](https://creativecommons.org/licenses/by/4.0/)

[Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

only improved knowledge grounding but also mechanisms to control and organize the decision process.

Retrieval-augmented generation (RAG) improves the reliability of LLM-based systems by incorporating external knowledge sources, such as operational manuals and historical maintenance records, into the generation process [7,8]. Recent studies have applied RAG to industrial diagnosis and maintenance decision support by integrating domain-specific knowledge repositories [9,10]. However, these approaches primarily focus on retrieving relevant information and generating knowledge-grounded responses, without explicitly considering the collaborative decision-making process required in URT fault management. In practical operations, fault handling involves multiple roles, including dispatchers, station staff, and maintenance personnel, each with distinct responsibilities and information requirements. Existing RAG systems typically assume a shared reasoning space, which may blur role responsibilities and introduce redundant or conflicting instructions in collaborative operational environments. Therefore, an effective decision-support system should not only retrieve relevant knowledge but also determine the appropriate participants and generate role-consistent actions rather than a single unrestricted response.

To address these challenges, we propose a role-aware agentic RAG framework for URT fault management. The framework integrates keyword-based knowledge retrieval, role-aware multi-agent reasoning, and constrained action generation within a unified architecture. The knowledge base is constructed from the official *Guide to Handling Train Operation Failures*, compiled by Ningbo Rail Transit, which provides standardized procedures for vehicles, signaling and communication systems, and other operational subsystems. For each fault event, the framework retrieves relevant operational knowledge, activates the role-specific agents, and generates constrained operational actions according to predefined responsibilities. By explicitly incorporating operational roles and restricting agent outputs to predefined actions, the framework provides structured decision support for URT fault handling.

The main contributions of this work are summarized as follows:

- We propose a role-aware agentic RAG framework for urban rail transit fault management, which activates agents according to operational responsibilities and enables role-specific reasoning for fault handling decisions.
- We introduce a field-weighted keyword retrieval strategy for structured urban rail transit documents, which exploits domain-specific terminology and document structures to achieve accurate and efficient knowledge retrieval.
- Experiments on real-world urban rail transit fault cases show that the proposed method achieves competitive retrieval effectiveness while substantially improving decision quality and role assignment accuracy compared with other baselines, with only a small increase in inference latency over the strongest agentic RAG baseline.

Outline

The remainder of this paper is organized as follows. Section 2 reviews related work on retrieval-augmented generation, agentic large language models, and LLM-based intelligent transportation applications. Section 3 introduces the proposed role-aware agentic RAG framework for URT fault management. Section 4 presents the experimental results and analysis. Section 5 concludes the paper and discusses future research directions.

2. Related Work

2.1. Retrieval-Augmented Generation

The RAG framework [7] demonstrated that retrieving external documents can substantially improve performance on knowledge-intensive tasks. Subsequent work developed

dense retrieval methods, such as Dense Passage Retrieval (DPR), which learn dense representations for queries and documents and have become a core component of modern RAG systems [11,12]. Recent studies have extended RAG to industrial fault diagnosis and maintenance decision support by incorporating domain-specific knowledge, including technical manuals, maintenance records, and historical fault cases, providing more reliable and better-supported diagnostic responses [9,13–15].

Besides dense retrieval, keyword-based lexical retrieval remains an important paradigm for document search, particularly in domains with standardized terminology and structured documentation. Classical probabilistic retrieval methods, such as BM25 [16], have demonstrated strong effectiveness on lexical matching tasks and remain widely adopted in industrial information retrieval systems. Recent studies [17,18] further suggest that lexical retrieval can outperform dense retrieval when queries and documents contain consistent domain-specific vocabularies or predefined technical expressions. These observations motivate the use of lightweight keyword-based retrieval for operational manuals, where exact procedural terms are often more informative than semantic similarity.

2.2. *Agentic Large Language Models*

Recent advances have extended LLMs from passive text generators to autonomous agents capable of reasoning, planning, and interacting with external tools [19]. ReAct combines reasoning and action through iterative interaction with external environments, improving both task performance and interpretability [20]. Reflexion further introduces self-reflection through language feedback, enabling agents to improve future decisions based on previous failures [21]. More recently, multi-agent frameworks, such as CAMEL and MetaGPT, have demonstrated that decomposing complex tasks into coordinated role-specific agents can improve scalability and collaborative problem solving [22,23]. Recent surveys further identify modular agent architectures as a promising direction for long-horizon reasoning, tool use, and collaborative decision making [19,24]. Although these studies demonstrate the advantages of agent-based reasoning, most existing frameworks are designed for general-purpose tasks. Comparatively little attention has been paid to role-aware collaboration under URT operational constraints, where responsibilities are predefined and generated actions must strictly comply with standardized procedures.

2.3. *LLMs for Intelligent Transportation*

Large language models have recently attracted increasing attention in intelligent transportation systems [3,25]. Existing studies have explored their use in traffic information analysis, incident summarization, infrastructure management, and multimodal transportation intelligence by integrating heterogeneous data sources, including sensor measurements, operational reports, and visual observations [26]. Recent work has also highlighted the potential of agentic LLM frameworks for Transportation Systems Management and Operations, where intelligent systems continuously retrieve information, reason over evolving situations, and interact with external tools [27].

3. Methodology

This section presents the proposed role-aware agentic RAG framework for URT fault management. The framework is designed to support collaborative decision making among multiple operational roles under dynamic fault conditions. As illustrated in Figure 1, the proposed architecture integrates role-aware multi-agent coordination, knowledge-grounded reasoning, and constrained response generation within a unified operational loop.

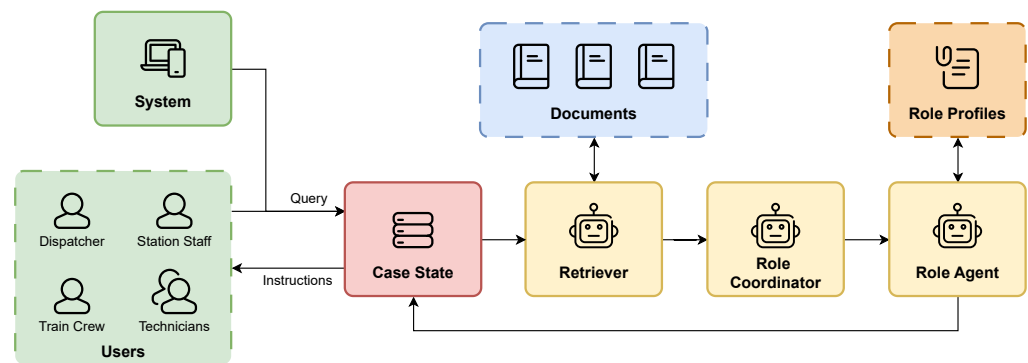


Figure 1. Overview of the proposed role-aware agentic RAG framework. A fault case state is first processed by the Retriever to obtain relevant knowledge. The retrieved knowledge is then used by the Role Coordinator to activate the required role-specific agents according to Equation (2).

The fault-handling process follows an iterative event-driven workflow. Given the current fault state, relevant operational knowledge is first retrieved from the knowledge repository. A role coordinator subsequently determines which operational roles should participate in the current reasoning cycle. Activated agents then generate operational instructions within a predefined action space, and the generated responses are incorporated into the shared case state for subsequent reasoning. This iterative mechanism allows the framework to adapt to evolving fault scenarios while preserving temporal consistency throughout the decision-making process.

3.1. Role-Aware Agentic RAG Framework

Urban rail transit fault management requires coordination among multiple operational roles, including train crew, maintenance personnel, and dispatchers. Each role has distinct responsibilities and requires different information for decision-making. Using the same reasoning context for all roles may introduce irrelevant information and lead to inconsistent decisions. To support role-specific reasoning, the proposed framework assigns an independent LLM agent to each role while maintaining coordination through a shared case state.

3.1.1. Role Definitions

We define eight roles according to the functional division commonly involved in urban rail fault handling: dispatcher, train crew, station staff, communication and signaling engineer, power supply engineer, rolling stock engineer, mechanical and electrical (M&E) engineer, and infrastructure engineer. The first three represent frontline operational personnel responsible for traffic control, train operation, and station operation, respectively, whereas the remaining five represent engineering disciplines responsible for communication/signaling, traction power, rolling stock, station M&E equipment, and track/civil infrastructure. The complete role definitions supplied to the LLM are provided in Appendix A.

3.1.2. Multi-Role Agent Architecture

For each role r , the corresponding agent receives three inputs: the retrieved operational documents $\mathcal{D}_t$, its role-specific interaction history $\mathcal{H}_r$, and the current event e_t , which contains the event source and content generated by all operational roles. The agent input is defined as

$$x_r = (\mathcal{D}_t, \mathcal{H}_r, e_t). \quad (1)$$

Unlike a shared-context agent, each role-specific agent only accesses its own historical interactions, preventing information leakage across different operational roles.

Before response generation, a role coordinator evaluates the relevance of each role to the current fault case. The activation process is defined as

$$g_r = \text{LLM}_{\text{resp}}(x_r), \quad g_r \in \{0, 1\}, \quad (2)$$

where LLM_{resp} is introduced in Appendix B.2, $g_r = 1$ indicates that role r participates in the current reasoning cycle, whereas $g_r = 0$ indicates that the role remains inactive. Only activated agents generate responses, thereby reducing redundant interactions and improving coordination efficiency while preserving role-specific reasoning.

3.1.3. Constrained Response Generation

In URT operations, unrestricted language generation may produce responses that are difficult to verify. To improve operational reliability, the proposed framework constrains each agent response to one of three predefined categories: *idle*, *wait*, and *action*. Specifically, *idle* indicates that the role is not relevant to the current fault handling process and therefore requires no action; *wait* indicates that the role should defer its decision until additional information becomes available from other roles or the operational system; and *action* indicates that the role should perform an operational task. The prompt template used in this generation step is provided in Appendix B.2.

Each generated response contains the selected action category, a single operational instruction, and a brief justification. By restricting each response to one executable action, the generated instructions remain concise and easier to verify. After generation, the response from role r is recorded as a new event,

$$\mathcal{E}_{t+1} = \mathcal{E}_t \cup \{e_{t+1,r}\}, \quad (3)$$

where $e_{t+1,r}$ denotes the event generated by role r . The updated event history is used in the next reasoning cycle. It is noted that the framework maintains a shared case state that is updated after each interaction. It integrates the latest events, role-specific interaction histories, and retrieved operational knowledge, and serves as the context for the next reasoning cycle.

3.1.4. Knowledge-Grounded Conservative Decision Making

To ensure that every decision is supported by domain knowledge, the framework retrieves relevant documents from the knowledge repository before generating a response. Given the current event history, the retrieval module produces a set of relevant references,

$$\mathcal{D}_t = \text{LLM}_{\text{Retrieve}}(\mathcal{E}_t), \quad (4)$$

where $\text{LLM}_{\text{Retrieve}}$ is in Appendix B.1. The retrieved knowledge is organized into two levels. When an exact operational procedure corresponding to the current fault is identified, it is used as the primary reference for decision generation. If no exact procedure is available, similar historical cases are retrieved as supplementary evidence. In this case, agents are not allowed to infer new operational procedures but instead provide conservative next-step guidance based on the available references. This strategy enables the framework to handle incomplete or previously unseen fault scenarios while reducing unsupported operational recommendations.

3.2. Keyword-Based Retrieval Algorithm

The proposed retrieval algorithm performs lexical matching between normalized query keywords and structured document fields without relying on dense embeddings. This design is motivated by the characteristics of urban rail fault management, where fault

reports and operational manuals use standardized equipment identifiers, fault codes, and technical terminology. Consequently, exact keyword matching is often sufficient to identify the relevant operating procedures. Candidate documents are subsequently ranked using field-weighted matching scores, resulting in a deterministic and computationally efficient retrieval process.

3.2.1. Document Indexing and Query Parsing

During offline preprocessing, each document is organized into a collection of searchable fields, including the *title*, *chapter*, and content sections such as *symptom*, *solution*, *impact*, and *introduction*. For a document d , these fields are denoted by $\mathcal{F}(d)$.

Before indexing, all textual content is normalized using a unified preprocessing pipeline. Alphabetic characters are converted to lowercase, redundant whitespace and new-line characters are removed, and language-specific tokenization is applied when necessary. For mixed Chinese–English documents, Chinese character sequences are preserved while English text is normalized to ensure consistent lexical matching. The same normalization pipeline is applied to incoming queries to maintain consistency between query terms and indexed documents.

At query time, the input is transformed into a keyword set $\mathcal{T} = \{t_1, t_2, \dots, t_m\}$. This transformation is performed either by an LLM-based query analyzer or, as a fallback, by a rule-based tokenizer. The tokenizer extracts alphanumeric tokens and contiguous Chinese character sequences of length at least two, then discards a fixed list of stopwords.

Different document fields contribute unequally to fault retrieval. Therefore, each field $f \in \mathcal{F}(d)$ is assigned a predefined weight w_f according to its diagnostic importance. These values are fixed heuristic weights rather than learned parameters: the title and symptom fields receive the highest priority, the solution field the next-highest priority, impact an intermediate priority, and contextual fields such as introduction and chapter lower priority. The weights used in this work are summarized in Table 1.

Table 1. Weight assignment for searchable fields.

Field	Weight w_f
title	5
symptom	5
solution	4
impact	3
introduction	2
chapter	2
other fields	1

3.2.2. Field-Weighted Scoring

The retrieval score is computed by accumulating weighted keyword matches across all searchable fields of a document. To determine whether a query keyword matches a document field, for a query keyword t , a document d , and one of its fields f , we define a binary match indicator

$$\mathbb{I}(t, f, d) = \begin{cases} 1, & \text{if } t \text{ appears in the normalized content of } f \text{ in } d, \\ 0, & \text{otherwise.} \end{cases} \quad (5)$$

Multiple occurrences of the same keyword within a field are counted only once. The retrieval score of document d is then computed as

$$s(d) = \sum_{t \in \mathcal{T}} \sum_{f \in \mathcal{F}(d)} w_f \cdot \mathbb{I}(t, f, d), \quad (6)$$

where w_f denotes the predefined weight assigned to field f . Documents with $s(d) > 0$ are ranked, and the top- K candidates are returned. The field weights prioritize diagnostically informative sections, while binary matching keeps the scoring process deterministic and avoids repeated contributions from the same keyword within a field.

3.2.3. Algorithm and Complexity

Algorithm 1 summarizes the complete retrieval procedure. During offline indexing, each document is parsed into searchable fields and normalized before being added to the document repository. This preprocessing stage has linear time complexity with respect to the total amount of indexed text. For each query, the retrieval algorithm scans all indexed documents and compares every query keyword with each searchable field. Assuming constant-time keyword matching on normalized text, the overall retrieval complexity is

$$O(|\mathcal{D}| \cdot |\mathcal{T}| \cdot |\mathcal{F}|),$$

where $|\mathcal{D}|$ denotes the number of indexed documents, $|\mathcal{T}|$ is the number of query keywords, and $|\mathcal{F}|$ is the average number of searchable fields per document. The proposed method is deterministic and avoids the computational overhead associated with neural encoding and approximate nearest-neighbor search.

Algorithm 1 Keyword-Based Document Retrieval

Require: Query Q , document set $\mathcal{D}$, number of returned documents K

Ensure: Top- K ranked documents

```

1:  $\mathcal{T} \leftarrow \text{EXTRACTKEYWORDS}(Q)$ 
2: if  $\mathcal{T} = \emptyset$  then
3:    $\mathcal{T} \leftarrow \text{FALLBACKTOKENIZER}(Q)$ 
4: end if
5:  $\mathcal{C} \leftarrow \emptyset$ 
6: for each document  $d \in \mathcal{D}$  do
7:    $s(d) \leftarrow 0$ 
8:   for each keyword  $t \in \mathcal{T}$  do
9:     for each field  $f \in \mathcal{F}(d)$  do
10:      if  $\mathbb{I}(t, f, d) = 1$  then
11:         $s(d) \leftarrow s(d) + w_f$ 
12:      end if
13:    end for
14:  end for
15:  if  $s(d) > 0$  then
16:    Append  $(d, s(d))$  to  $\mathcal{C}$ 
17:  end if
18: end for
19: Sort  $\mathcal{C}$  in descending order of  $s(d)$ 
20: return Top- $K$  elements of  $\mathcal{C}$ 

```

4. Experiments

This section describes the experiments used to evaluate the proposed framework. The experiments are designed to assess retrieval effectiveness, semantic quality, role assignment accuracy, and inference efficiency while comparing the proposed approach with

representative retrieval-augmented and non-retrieval baselines. Details of the dataset, baseline methods, evaluation metrics, implementation settings, and experimental results are presented in the following subsections.

4.1. Dataset

The retrieval corpus is constructed based on the *Guide to Handling Train Operation Failures*, an internal training manual used for handling operational incidents in Ningbo Rail Transit. The manual consists of five chapters and covers dozens of operational incidents. Each incident and its response procedure are treated as an individual retrieval unit. This corpus serves as the external knowledge source for the retrieval-based methods.

This evaluation dataset contains 159 emergency drill scenarios sourced from various internal training materials. These scenarios are designed by human experts for frontline emergency response training in Ningbo Rail Transit. The dataset covers the major failure categories encountered in metro operations, including rolling stock (vehicle), signaling and communications, power supply, mechanical and electrical (M&E) systems, and track and civil engineering failures. Table 2 summarizes the distribution of fault categories. Each case describes a complete operational scenario, including a fault description, background information, standard response procedures, and the relevant role responsibilities required for emergency handling.

Table 2. Distribution of fault categories in the evaluation dataset.

Fault Category	Number of Cases
Rolling Stock Faults	30
Signaling and Communication Faults	69
Power Supply Faults	18
Mechanical and Electrical System Faults	27
Track and Civil Engineering Faults	15
Total	159

It should be noted that although each case has a corresponding handling procedure in the manual, these drill cases are not included in the retrieval corpus. The drill cases serve as evaluation queries, while their response procedures and role assignments are used as ground truth. Therefore, these cases provide a domain-specific benchmark for evaluating retrieval performance, semantic consistency, and role assignment accuracy.

4.2. Baselines

To comprehensively evaluate the effectiveness of the proposed framework, four representative configurations are selected as baselines, ranging from pure large language model generation to progressively enhanced retrieval and agent-based architectures.

LLM Only: The language model directly generates handling instructions without accessing any external knowledge.

BM25 RAG: Relevant manual documents are retrieved using the classical BM25 lexical retrieval algorithm, and the retrieved contents are provided as additional context for instruction generation.

Dense RAG: Documents are retrieved using dense embeddings, reranked by an LLM reranker, and then supplied to the LLM.

Agentic RAG: The proposed keyword-based retrieval module is combined with a standard RAG pipeline employing a single agent.

Role-Aware Agentic RAG: The complete framework proposed in this paper, integrating keyword-based retrieval, role-aware multi-agent reasoning and constrained action generation.

It should be noted that the selected baselines provide a staged comparison of the proposed framework. Specifically, LLM Only evaluates the foundation model without external knowledge retrieval; BM25 RAG and Dense RAG assess the effect of retrieval-based knowledge augmentation; Agentic RAG further incorporates agentic retrieval and reasoning; and the proposed Role-Aware Agentic RAG adds role-aware activation and coordination on top of this architecture. These comparisons provide a progressive evaluation of the framework, with retrieval, agentic reasoning, and role-aware coordination introduced in stages. Although the baselines do not isolate each component individually, the performance differences across these configurations show the overall benefit of adding these components.

4.3. Evaluation Metrics

The proposed framework is evaluated from three complementary perspectives.

Retrieval effectiveness is evaluated using Hit@1 and Exact Retrieval Ratio (ERR). Hit@1 measures whether the ground-truth procedure is ranked first, while ERR measures the proportion of queries for which the retriever returns exactly one document. Since BM25 and dense retrieval do not inherently support exact-match filtering, we apply a z-score based thresholding strategy to the retrieval scores and retain only candidates with a z-score above 2. If no candidate exceeds this threshold, we fall back to returning the top-K retrieved documents to ensure robust subsequent inferences. A query is counted as an exact retrieval only if exactly one document survives this filtering.

Decision quality is evaluated using three metrics. The F1 Score measures the character-level overlap between the generated instruction and the reference procedure (both in Chinese), capturing lexical fidelity at the character level. Semantic Score measures the semantic similarity between the generated instruction and the reference procedure, computed by the Qwen3-Reranker-4B model. Role F1 measures the agreement between the predicted and ground-truth sets of operational roles. It is computed from the precision and recall of the predicted role set with respect to the reference roles involved in each fault case.

Inference efficiency is measured by the average response latency (in seconds), reflecting end-to-end response time in practical deployment.

4.4. Experimental Settings

All experiments are conducted using the same implementation and evaluation pipeline to ensure a fair comparison. Unless otherwise specified, all methods share the same backbone LLM, document corpus, prompt template, decoding configuration, and test set. Therefore, the observed performance differences can be attributed to the retrieval strategy and reasoning architecture rather than implementation variations.

The instruction generation model is Qwen3.6-35B-A3B [28]. Dense retrieval is performed using Qwen3-Embedding-4B, while document reranking and semantic evaluation are performed using Qwen3-Reranker-4B. For all retrieval-based methods, the number of retrieved documents is fixed at $\text{Top-}K = 5$. The decoding temperature is set to 0.1 in all experiments. The proposed framework executes role-specific agents in parallel whenever independent responses can be generated, reducing unnecessary sequential reasoning while preserving execution consistency.

All LLM inference experiments are conducted on a server equipped with two Huawei Ascend 910B4 NPUs (Huawei, Shenzhen, China), each with 64 GB of device memory. Qwen3.6-35B-A3B is served using vLLM 0.18.0 with tensor parallelism across the two de-

vices and expert parallelism enabled. The serving configuration uses a GPU memory utilization ratio of 0.9, a maximum model length of 262,144 tokens, a maximum of 32 concurrent sequences, and a maximum of 32,768 batched tokens. Prefix caching and asynchronous scheduling are also enabled.

For latency evaluation, the experimental client processes one fault case at a time and does not explicitly batch multiple fault cases into a single inference request. Within each fault case, however, independent role-agent requests are issued concurrently when multiple roles are activated. Consequently, these concurrent requests may be dynamically batched by the vLLM serving engine. The reported latency therefore represents the end-to-end wall-clock latency of processing one complete fault case, including retrieval, role coordination, parallel role-agent inference, and response aggregation. This measurement protocol is applied consistently across the evaluated methods.

All experiments are conducted with 5 independent runs using different random seeds for LLM generation to account for the inherent stochasticity of autoregressive decoding. We report the mean and standard deviation of each metric across the 5 runs to provide a comprehensive assessment of both performance and stability. This experimental protocol supports the reproducibility and reliability of our findings.

4.5. Results

Tables 3–5 summarize the quantitative comparison among different methods in terms of retrieval effectiveness, generation quality, and inference latency, respectively. For retrieval effectiveness, all retrieval-based methods achieve high Hit@1 values, with scores above 0.96, indicating that the target procedure is usually ranked at the top position. Dense RAG achieves a Hit@1 of 0.994, while both Agentic RAG methods reach 1.0. A more significant difference is observed in ERR, which measures the proportion of queries for which the retrieval filtering process retains exactly one candidate document. BM25 and Dense Retrieval achieve ERR values of 0.245 and 0.239, respectively, whereas Agentic RAG and the proposed framework achieve substantially higher values of 0.846 and 0.851. Together with their Hit@1 scores of 1.0, these results suggest that both Agentic RAG methods more frequently isolate a single candidate while retaining the ground-truth procedure at the top rank.

Table 3. Retrieval effectiveness. Higher is better for both metrics. Values are reported as mean (standard deviation) over multiple runs.

Method	Hit@1	ERR
LLM Only	N/A	N/A
BM25 RAG	0.969 (0.000)	0.245 (0.000)
Dense RAG	0.994 (0.000)	0.239 (0.000)
Agentic RAG	1.0 (0.000)	0.846 (0.024)
Role-Aware Agentic RAG	1.0 (0.000)	0.851 (0.011)

Table 4. Response quality. Higher is better for all metrics. Values are reported as mean (standard deviation) over multiple runs.

Method	F1 Score	Semantic Score	Role F1
LLM Only	0.267 (0.003)	0.229 (0.012)	0.458 (0.011)
BM25 RAG	0.633 (0.019)	0.851 (0.024)	0.809 (0.032)
Dense RAG	0.650 (0.012)	0.862 (0.019)	0.800 (0.023)
Agentic RAG	0.659 (0.015)	0.893 (0.016)	0.824 (0.042)
Role-Aware Agentic RAG	0.679 (0.007)	0.965 (0.005)	0.915 (0.004)

Table 5. End-to-end inference latency (in seconds). Lower is better. Values are reported as mean (standard deviation) over multiple runs.

Method	Latency
LLM Only	4.188 (0.277)
BM25 RAG	4.605 (0.210)
Dense RAG	5.592 (0.162)
Agentic RAG	15.942 (1.959)
Role-Aware Agentic RAG	16.481 (1.414)

For decision quality, retrieval augmentation provides substantial improvements over LLM-only generation. The character-level F1 Score increases from 0.267 for LLM Only to 0.633 and 0.650 for BM25 RAG and Dense RAG, respectively, and further to 0.659 for Agentic RAG. The proposed framework achieves the highest F1 Score of 0.679, indicating improved lexical consistency with the reference procedures. A more significant improvement is observed in semantic similarity. The Semantic Score increases from 0.893 for Agentic RAG to 0.965 for the proposed framework, corresponding to an 8.1% relative improvement. Role F1 also increases from 0.824 to 0.915, indicating more accurate assignment of handling instructions to the appropriate operational roles. This suggests that the benefit of the role-aware framework lies not only in reproducing the surface form of reference procedures, but in generating instructions that are more semantically aligned with the expected operational decisions while maintaining lexical consistency.

Regarding inference efficiency, LLM Only, BM25 RAG, and Dense RAG require less than 6 s on average because they do not involve multi-agent reasoning. Agentic RAG introduces additional retrieval and reasoning steps, increasing the latency to 15.942 s. The proposed framework incurs a slightly higher latency of 16.481 s due to role activation and agent coordination. In practical deployment, a 16.5-s response is acceptable for most decision-support tasks, including fault assessment. Nevertheless, in rare and highly time-critical situations, this latency may be restrictive. Overall, the proposed framework achieves improved decision quality with a modest latency increase over the Agentic RAG baseline.

4.6. Discussion

The experimental results indicate that retrieval accuracy alone is insufficient to guarantee reliable operational decisions. Although retrieval-based methods achieve high Hit@1 scores above 0.96, their decision quality differs substantially. For example, Dense RAG achieves a Hit@1 of 0.994 but obtains a semantic score of only 0.862, whereas the proposed framework reaches 0.965. This result suggests that accurate retrieval is a necessary foundation, but effective role-aware coordination and constrained action generation are also important for converting retrieved knowledge into appropriate operational decisions.

Another notable observation is that keyword-based retrieval is well suited to standardized reference documents. Operational manuals use consistent equipment identifiers, fault codes, and procedural terminology, allowing lexical matching to retrieve relevant documents with high accuracy. As a result, the proposed retrieval module achieves consistently high retrieval performance without relying on dense embedding models. In practice, frontline staff may use informal or non-standard terms that do not directly match the terminology in the reference documents. Our LLM-based keyword extraction provides some ability to interpret such expressions and map them to relevant operational terms, and we did not observe a substantial impact in our experiments. Nevertheless, handling highly domain-specific or evolving terminology remains a limitation to be explored in the future.

This study is based on a single URT operator, Ningbo Rail Transit, and one internal operational manual. The results may not fully reflect the differences across operators,

particularly in operational procedures and terminology. For real-world deployment, we will evaluate the framework with broader datasets from multiple operators and different operational environments. Future work will also test the framework with different LLM backbones to examine whether the results remain consistent.

In the current experiments, we did not observe conflicting instructions among simultaneously activated roles. However, such conflicts may occur in rare cases when different roles provide inconsistent recommendations. The current framework does not include a separate mechanism for resolving such conflicts, which will be investigated in future work.

Finally, the framework is intended to support rather than replace human operators. In practical deployment, generated instructions can be reviewed by human operators before execution. Such a human-in-the-loop mechanism is not implemented or evaluated in this study. Future work will examine how operator feedback can be incorporated into the decision-making process and its effects on operational reliability.

5. Conclusions

This paper presented a role-aware agentic RAG framework for urban rail transit fault management. Building upon a standard RAG pipeline, the framework integrates keyword-based retrieval, role-aware multi-agent reasoning, and constrained action generation within a unified event-driven architecture. By activating only the operational roles relevant to the current fault and constraining each response to predefined action categories, the framework improves role assignment. Experimental results on an evaluation dataset constructed from real-world emergency exercise materials demonstrate that the proposed framework consistently achieves the best decision-quality results among the evaluated methods while maintaining competitive retrieval effectiveness and practical inference latency.

Future work will extend the framework to multi-modal operations by incorporating sensor data, equipment logs, and visual observations. Continual knowledge updating and online human feedback will also be investigated to improve the system's ability to adapt to evolving operational conditions in real-world urban rail transit environments.

Author Contributions: Methodology, Q.Z. and Y.Y.; software, Q.Z.; validation, Y.Y.; formal analysis, W.Z., H.F., J.B., Q.Z. and Y.Y.; data curation, H.F.; writing—original draft, W.Z., H.F., Q.Z. and Y.Y.; writing—review and editing, J.B., Q.Z. and Y.Y.; visualization, Q.Z. and Y.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Natural Science Foundation of Shaanxi Province, grant number 2024JC-YBMS-569, and Key Research and Development Program of Ningbo City, grant number 2023Z130.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author due to commercial confidentiality and operational security concerns.

Acknowledgments: During the preparation of this manuscript, the authors used deepseek-v4-flash and OpenAI GPT-5.6 Terra for the purposes of language correction and polishing. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

Conflicts of Interest: Authors Qiming Zhong, Yueyao Yu and Hui Fang were employed by Ningbo Rail Transit Group Co., Ltd., China. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

URT	Urban Rail Transit
M&E	Mechanical and Electrical
DPR	Dense Passage Retrieval
LLM	Large Language Model
RAG	Retrieval-Augmented Generation
BM25	Best Matching 25
ERR	Exact Retrieval Ratio

Appendix A. Role Definitions

The proposed framework contains eight predefined roles according to the functional division commonly involved in urban rail fault handling. The first three roles correspond to frontline operational personnel responsible for traffic control, train operation, and station operation, respectively, whereas the remaining five correspond to engineering disciplines responsible for communication and signaling, traction power, rolling stock, station mechanical and electrical equipment, and track/civil infrastructure. Table A1 provides the complete definitions used in the implementation.

Table A1. Definitions of the eight predefined roles used in the proposed framework.

Role	Definition
Dispatcher	Responsible for train traffic dispatching and control, monitoring train operating status, issuing dispatching commands, and maintaining operational safety and traffic order.
Train Crew	Responsible for train driving and onboard operational duties, confirming the technical condition of trains before departure from the depot, executing dispatching commands, monitoring train operating parameters, and reporting abnormalities during operation.
Station Staff	Responsible for station passenger organization and platform safety, maintaining orderly boarding and alighting, guiding passenger movements, and handling emergencies and passenger requests within the station.
Communication and Signaling Engineer	Responsible for routine maintenance and fault handling of communication and signaling systems, including interlocking, automatic train control, and wireless communication equipment, to maintain safe and reliable signaling operation.
Power Supply Engineer	Responsible for monitoring and maintaining traction-power and power-distribution systems, handling faults involving overhead contact systems or power rails and substations, and maintaining continuous and stable power supply.
Rolling Stock Engineer	Responsible for train inspection, maintenance, and emergency fault handling; monitoring train technical conditions; and handling faults involving vehicle equipment such as passenger doors, traction systems, and braking systems.
Mechanical and Electrical Engineer	Responsible for the operation and maintenance of station mechanical and electrical equipment, including escalators, platform screen doors, environmental control systems, water supply and drainage systems, and fire-protection systems.
Infrastructure Engineer	Responsible for inspection and maintenance of track and civil infrastructure, including track geometry, track bed, and tunnel structures, and for handling track defects and clearance abnormalities.

Appendix B. Prompt Templates

This appendix reports the prompt templates used in the proposed method. The prompts are translated into English for presentation. In the implementation, Chinese prompts are used to match the language of the documents and users. Curly braces denote variables filled by the system at runtime.

Appendix B.1. Retrieval Prompts

The prompt template used for keyword extraction during the retrieval process is as follows:

```
You are a fault manual retrieval agent. Extract retrieval clues  
from the user query.
```

```
User query:
```

```
{query}
```

```
The book chapter structure includes:
```

- chapter
- title
- introduction
- symptom
- impact
- solution

```
Requirements:
```

- "keywords" should include fault symptoms, component names, error codes, impacts, resolution actions, etc.
- Note that users may use colloquial or abbreviated terms. Map these to standard terminology.
- "search_paths" must be ordered by priority.
- Output only JSON format. Do not output Markdown or explanatory text.

```
Output JSON format:
```

```
{  
  "keywords": ["keyword1", "keyword2"],  
  "search_paths": ["symptom", "impact", "solution", "title"],  
  "reason": "Explanation of why this search strategy is chosen"  
}
```

The prompt template for document judgment and ranking is as follows:

```
You are a fault manual chapter-matching agent. You have received the
following user query and candidate chapter contents.
User query:
{query}

Document candidates:
{document_candidates}

Determine whether a specific chapter can be uniquely matched.

Requirements:
- "candidate_ids" must be sorted in descending order of content
  relevance (highest probability first).
- If a unique candidate can be identified, set "exact_match" = true;
  otherwise, set "exact_match" = false.
- When "exact_match" = false, "info_required" must list the additional
  information needed from the user to resolve the ambiguity.
- If no further information is needed, "info_required" may be empty.
- Output only JSON format. Do not output Markdown or explanatory text.

Return JSON format as follows:
{
  "exact_match": true or false,
  "candidate_ids": ["doc_id1", "doc_id2", ...],
  "reason": "Brief rationale for the decision",
  "info_required": []
}
```

Appendix B.2. Role Routing Prompt

The prompt template for role routing is as follows:

You are the instruction routing agent in a metro fault handling system. Your task is to determine, based on the latest event and the current case context, which roles need to respond to that event or need to have their pending instructions modified.

Role Responsibilities:

{role}: {role_responsibility}

Routing rules:

- If the event originates from the system (automated report), determine the relevant roles based on the technical domain implied by the event content and the reference documents.
- If the event originates from a user report, that role should pay attention to this event; other roles should review the resolution process in the reference document and any previous instructions to determine whether the event will affect or trigger the next step in the process.
- Carefully consider the roles indicated in the reference documents. Output only roles that are directly relevant to the current event. Do not include irrelevant roles.

Output Requirements:

- Output must be valid JSON. Do not output Markdown or explanatory text.
- The "roles" array must contain role names exactly as they appear in the role responsibilities list above.
- "reason" should be kept within 50 words if possible.

Output JSON format:

```
{
  "roles": ["Role1", "Role2"],
  "reason": "Brief rationale for the routing decision"
}
```

Reference documents:

{documents}

Historical events:

{previous_events}

Previous instructions:

{previous_instructions}

Current event:

{current_event}

Appendix B.3. Instruction Generation Prompt

The prompt template for instruction generation is as follows:

You are a single-role agent in subway fault handling. Your current assigned role is: {role}.

The responsibility of {role} is: {role_responsibility}

You are only allowed to decide whether to issue one next-step instruction to {role}.

You are not responsible for document retrieval, not allowed to reply directly to the system, and not allowed to write instructions for other roles.

Core rules:

- Output must be valid JSON. Do not output Markdown or explanatory text.
- "state" should be exactly one of the following three values:
 - "idle": if the role is irrelevant to the current fault and requires no action;
 - "wait": if the role needs more information from others or the system before deciding;
 - "action": if the role should perform an operational task.
- "content" may contain only one currently executable action. Do not output a full procedure at once, do not use numbered lists, and do not write multiple consecutive actions.
- If the latest event comes from the system, treat it only as a global fact update; do not output anything to the system.
- Generate instructions based on document matches. Provide single-step operational guidance strictly following the matched sections.
- Do not invent proprietary buttons, screen names, device IDs, or procedure steps that do not appear in the provided documents.
- Both "content" and "reason" should be kept within 50 words if possible.

Output JSON format:

```
{
  "state": "idle"/"wait"/"action"
  "content": "A single action step for this role",
  "reason": "Brief explanation of why this action/decision is taken"
}
```

Reference documents:

{documents}

Historical events:

{previous_events}

Current event:

{current_event}

References

1. Zang, Y.; Shangguan, W.; Cai, B.; Wang, H.; Pecht, M.G. Methods for fault diagnosis of high-speed railways: A review. *Proc. Inst. Mech. Eng. Part O J. Risk Reliab.* **2019**, *233*, 908–922. [\[CrossRef\]](#)
2. Sapkota, R.; Roumeliotis, K.I.; Karkee, M. Ai agents vs. agentic ai: A conceptual taxonomy, applications and challenges. *Inf. Fusion* **2026**, *126*, 103599. [\[CrossRef\]](#)
3. Wandelt, S.; Zheng, C.; Wang, S.; Liu, Y.; Sun, X. Large language models for intelligent transportation: A review of the state of the art and challenges. *Appl. Sci.* **2024**, *14*, 7455. [\[CrossRef\]](#)
4. Gan, L.; Chu, W.; Li, G.; Tang, X.; Li, K. Large models for intelligent transportation systems and autonomous vehicles: A survey. *Adv. Eng. Inform.* **2024**, *62*, 102786. [\[CrossRef\]](#)
5. Ji, Z.; Lee, N.; Frieske, R.; Yu, T.; Su, D.; Xu, Y.; Ishii, E.; Bang, Y.J.; Madotto, A.; Fung, P. Survey of hallucination in natural language generation. *ACM Comput. Surv.* **2023**, *55*, 1–38. [\[CrossRef\]](#)
6. Huang, L.; Yu, W.; Ma, W.; Zhong, W.; Feng, Z.; Wang, H.; Chen, Q.; Peng, W.; Feng, X.; Qin, B.; et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Trans. Inf. Syst.* **2025**, *43*, 1–55. [\[CrossRef\]](#)
7. 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. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 9459–9474.
8. 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, Lisbon, Portugal, 14–15 April 2024; pp. 194–199.
9. Cheng, Y.J.; Chen, S.L. An Event-Driven Dual-Mode Multilingual RAG Framework for Industrial Maintenance. *IEEE Access* **2026**, *14*, 81494–81514. [\[CrossRef\]](#)
10. Wulf, J.; Meierhofer, J.; Pascher, N. Guided Operations and Maintenance with Retrieval-Augmented Generation: System Design and Evaluation at a Machine Manufacturer. *Procedia CIRP* **2026**, *139*, 349–354. [\[CrossRef\]](#)
11. Karpukhin, V.; Oguz, B.; Min, S.; Lewis, P.; Wu, L.; Edunov, S.; Chen, D.; Yih, W.T. Dense Passage Retrieval for Open-Domain Question Answering. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Online, 16–20 November 2020; pp. 6769–6781.
12. Gao, Y.; Xiong, Y.; Gao, X.; Jia, K.; Pan, J.; Bi, Y.; Dai, Y.; Sun, J.; Wang, H.; Wang, H.; et al. Retrieval-augmented generation for large language models: A survey. *arXiv* **2023**, arXiv:2312.10997.
13. Xu, J.; Xu, Z.; Jiang, Z.; Chen, Z.; Luo, H.; Wang, Y.; Gui, W. Labeling-free RAG-enhanced LLM for intelligent fault diagnosis via reinforcement learning. *Adv. Eng. Inform.* **2026**, *69*, 103864. [\[CrossRef\]](#)
14. Wang, Y.; Cai, X.; Cui, B.; Tang, X.; Wood, C.; Hou, Y. An instruction fine-tuning and retrieval-augmented generation framework for intelligent defect diagnosis and maintenance decision support in high-speed railway turnouts. *Comput.-Aided Civ. Infrastruct. Eng.* **2026**, *49*, 100098. [\[CrossRef\]](#)
15. Lukens, S.; Bishof, M.; Siddiqui, N.; West, D. An Evaluation Framework for Fault Diagnosis Using Technical Manuals in Retrieval-Augmented Large Language Models. In Proceedings of the Annual Conference of the PHM Society, Bruges, Belgium, 2–5 June 2025; Volume 17.
16. Robertson, S.; Zaragoza, H. *The Probabilistic Relevance Framework: BM25 and Beyond*; Now Publishers Inc.: Hanover, MA, USA, 2009; Volume 4.
17. Yang, P.; Fang, H.; Lin, J. Anserini: Enabling the use of lucene for information retrieval research. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Tokyo, Japan, 7–11 August 2017; pp. 1253–1256.
18. Thakur, N.; Reimers, N.; Rüklé, A.; Srivastava, A.; Gurevych, I. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv* **2021**, arXiv:2104.08663.
19. Ferrag, M.A.; Tihanyi, N.; Debbah, M. From llm reasoning to autonomous ai agents: A comprehensive review. *IEEE Access* **2026**, *14*, 84237–84285. [\[CrossRef\]](#)
20. Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; Cao, Y. React: Synergizing Reasoning and Acting in Language Models. In *Proceedings of the 11th International Conference on Learning Representations; ICLR*: London, UK, 2023.
21. Shinn, N.; Cassano, F.; Gopinath, A.; Narasimhan, K.; Yao, S. Reflexion: Language agents with verbal reinforcement learning. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 8634–8652. [\[CrossRef\]](#)
22. Li, G.; Hammoud, H.; Itani, H.; Khizbullin, D.; Ghanem, B. Camel: Communicative agents for “mind” exploration of large language model society. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 51991–52008. [\[CrossRef\]](#)
23. Hong, S.; Zhuge, M.; Chen, J.; Zheng, X.; Cheng, Y.; Wang, J.; Zhang, C.; Yau, S.; Lin, Z.; Zhou, L.; et al. MetaGPT: Meta programming for a multi-agent collaborative framework. In Proceedings of the International Conference on Learning Representations, Vienna, Austria, 7–11 May 2024; Volume 2024, pp. 23247–23275.
24. Huang, X.; Liu, W.; Chen, X.; Wang, X.; Wang, H.; Lian, D.; Wang, Y.; Tang, R.; Chen, E. Understanding the planning of llm agents: A survey. *arXiv* **2024**, arXiv:2402.02716.

25. Shoaib, M.R.; Emara, H.M.; Zhao, J. A survey on the applications of frontier ai, foundation models, and large language models to intelligent transportation systems. In *Proceedings of the 2023 International Conference on Computer and Applications (ICCA)*; IEEE: New York, NY, USA, 2023; pp. 1–7.
26. Li, S.; Wang, Z.; Li, J.; Boriboonsomsin, K.; Barth, M.J.; Wu, G. Large Language Models in Transportation Systems Management and Operations: From Text Reasoning to Multi-modal Decision Support. *arXiv* **2026**, arXiv:2606.00991.
27. Xu, H.; Yuan, J.; Zhou, A.; Xu, G.; Li, W.; Ban, X.; Ye, X. Genai-powered multi-agent paradigm for smart urban mobility: Opportunities and challenges for integrating large language models (llms) and retrieval-augmented generation (rag) with intelligent transportation systems. *arXiv* **2024**, arXiv:2409.00494.
28. Qwen Team. Qwen3.6-35B-A3B: Agentic Coding Power, Now Open to All. 2026. Available online: <https://qwen.ai/blog?id=qwen3.6-35b-a3b> (accessed on 14 May 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.