LLM-Assisted Joint Ticket and Log Analysis for Incident Triage in Intelligent and Connected Vehicles

Ruowei Fu
Nankai University
Tianjin, China

Shenglin Zhang*
Nankai University
Tianjin, China

Wenwei Gu
Nankai University
Tianjin, China

Weiguo Li
Huawei Inc.
Shenzhen, China

Yongqian Sun
Nankai University
Tianjin, China

Dan Pei
Tsinghua University
Beijing, China

Abstract

With the rapid development of intelligent and connected vehicles (ICVs), in-vehicle systems have grown increasingly complex, leading to frequent anomalies in real-world usage. To maintain a high-quality user experience, efficient incident triage is essential for timely issue resolution. However, with the growing volume and complexity of incidents, traditional manual triage processes are becoming increasingly insufficient. Recent advances in large language models (LLMs) have demonstrated strong capabilities in reasoning and natural language understanding, offering new opportunities for automation. We propose *InsightTriage*, an end-to-end automated incident triage system designed and deployed for real-world industrial scenarios, which leverages LLMs to jointly analyze user queries and vehicle logs to generate accurate and explainable triage results. To address the lack of domain-specific knowledge, *InsightTriage* first constructs a component-level structured knowledge base in an offline phase by leveraging LLMs to automatically extract knowledge from historical vehicle logs. In the online phase, to mitigate the context length limitations of LLMs in log analysis, *InsightTriage* introduces a log retriever pretrained with contrastive learning, which efficiently identifies query-relevant log entries. Comprehensive experiments conducted on a dataset collected from the production environment of a top-tier global supplier for ICVs, demonstrate that *InsightTriage* achieves superior performance, reaching a weighted F1-Score of 0.801 (an improvement of 0.137 over the best baseline), while also providing explainable triage reports.

CCS Concepts

• **Software and its engineering → Software maintenance tools**.

Keywords

Incident Triage, Large Language Model, Log Analysis

*Corresponding author.

ASE '26, Munich, Germany

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

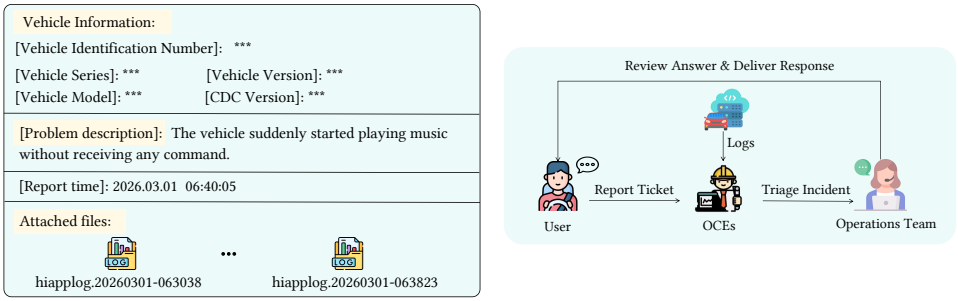
ACM Reference Format:
Ruowei Fu, Shenglin Zhang, Wenwei Gu, Weiguo Li, Yongqian Sun, and Dan Pei. 2026. LLM-Assisted Joint Ticket and Log Analysis for Incident Triage in Intelligent and Connected Vehicles. In *Proceedings of In 41th IEEE/ACM International Conference on Automated Software Engineering (ASE '26).* ACM, New York, NY, USA, 12 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

With the rapid advancement of intelligent and connected vehicles (ICVs), user experience has been significantly enhanced across various aspects, including voice interaction, intelligent connectivity, and automated control [30, 31]. However, the complexity and continuous evolution of the system lead to users frequently encountering various issues during real-world usage, such as voice wake-up failures, Bluetooth disconnections, and seat adjustment malfunctions. Such issues not only impair user experience but also introduce significant challenges to enterprise-level operation and maintenance [18, 41].

In real-world production environment, user-reported issues are typically accompanied by large volumes of telemetry logs uploaded from vehicles to cloud servers, where on-call engineers (OCEs) perform incident triage. The objective is to identify the root cause of each incident and provide appropriate responses or solutions to users [5]. However, this process is often time-consuming and labor-intensive due to the large scale and high complexity of in-vehicle systems, which require OCEs to carefully analyze user queries together with extensive vehicle system logs to reach reliable conclusions. Moreover, effective triage heavily depends on the expertise and experience of OCEs. Given the diversity of system components and failure scenarios, accumulating sufficient domain knowledge demands substantial effort. Consequently, triage automation has become a critical enterprise need to improve operational efficiency and alleviate manual workload.

In our context, the available textual information consists of user-reported issues and system logs. However, user descriptions are often brief and ambiguous, providing insufficient detail for accurate triage based solely on user queries. While system logs provide a detailed record of runtime system behavior [13, 21, 36], existing incident triage methods typically rely on simple keyword selection, which is insufficient to capture issue-relevant context [33]. As these approaches fail to incorporate user reports during log analysis, critical triage evidence may be overlooked. Therefore, effectively integrating user reports with system logs for comprehensive analysis remains a fundamental challenge.



(a) Example of incident ticket **(b) Ticket management system**

Figure 1: Illustration of the incident ticket and the overview of the ticket management system.

Recent advancements in large language models (LLMs) provide a promising direction for addressing these limitations. With strong capabilities in natural language understanding and reasoning [14, 24, 37], LLMs can jointly interpret user queries and system logs, and further perform multi-step reasoning to identify underlying causes of incidents. Moreover, they can generate explainable triage results with supporting evidence, thereby improving OCEs' understanding and decision confidence in real-world deployments. Inspired by these advantages, we propose an LLM-enhanced incident triage framework that integrates user queries and system logs for comprehensive analysis. However, it still faces two domain-specific challenges (detailed in Section 3).

Challenge 1: Insufficient Domain Knowledge. ICVs consist of numerous interdependent components developed by different teams, which leads to fragmented and incomplete domain knowledge. Such knowledge is often difficult to systematically organize, thereby limiting its effective utilization by LLMs for reasoning over large-scale system logs.

Challenge 2: Context Length Limitations of LLMs. Logs generated by large-scale ICVs are massive and noisy, often exceeding the context length of LLMs. Directly processing raw logs is inefficient and may obscure critical signals, making accurate retrieval of the most relevant log subsets essential for effective triage.

To address these challenges, we propose ***InsightTriage***, an automated incident triage system enhanced by LLM-based reasoning. For Challenge 1, we leverage LLMs to automatically extract and construct component-level structured knowledge from historical logs, forming a more comprehensive and systematic domain knowledge representation that provides explicit guidance for accurate and efficient triage. For Challenge 2, we introduce log-query contrastive pretraining to learn semantic representations of logs, enabling effective retrieval of query-relevant log entries and mitigating the context length limitation of LLMs in large-scale log analysis.

The main contributions of this work are as follows:

- We propose a novel end-to-end framework for automated incident triage, enhanced by LLM-based reasoning. To the best of our knowledge, *InsightTriage* is the first approach in the ICV ticket management domain that explicitly incorporates log data into the incident triage process. By jointly leveraging textual reports and system logs, our method enables more accurate and explainable triage.
- Comprehensive solution addressing key automation challenges. We: (1) propose an LLM-based operational knowledge mining approach that extracts component-level structured knowledge from historical logs, alleviating the lack of domain knowledge (Section 4.2), and (2) pretrain a log retriever via contrastive learning on large-scale log–query pairs, improving the efficiency of log analysis (Section 4.3).
- To substantiate the practical effectiveness of *InsightTriage*, we conduct a comprehensive evaluation on a dataset collected from the production environment of Huawei, a top-tier global supplier for ICVs. The results demonstrate the practical effectiveness of *InsightTriage*, achieving a weighted F1-Score of 0.801 for incident triage and outperforming the best baseline method by 0.137.

2 BACKGROUND

2.1 Incident Tickets

The inherent complexity of ICVs, combined with the rapid evolution of software and functionalities, inevitably leads to a wide range of issues encountered by users during real-world usage, such as voice assistant execution inaccuracy, network latency degradation, and driver distraction detection anomalies. These issues arise from a wide range of root-cause domains, such as the cockpit domain controller (CDC), telematics box (T-Box), and advanced driver assistance system (ADAS), among others. When such issues occur, users typically report them to the vehicle service provider in the form of incident tickets through various channels, such as in-vehicle voice assistants or companion mobile applications. As illustrated in Figure 1a, each ticket serves as a structured and comprehensive record of an incident, providing essential information to support subsequent triage and resolution processes. In particular, a ticket generally includes the following fields: the report time, a problem description (user's detailed description of the observed issue), vehicle-related information (e.g., vehicle identification number, vehicle series, vehicle model, vehicle version, and CDC version), as well as supplementary attachments (e.g., log data capturing the vehicle's runtime status), among others.

2.2 Incident Triage

ICV cloud service providers typically employ a sophisticated ticket management system to efficiently handle user-reported incident

tickets. Figure 1b illustrates a representative workflow of a manual ticket handling process. Specifically, when encountering issues during vehicle operation, users create and submit detailed incident tickets describing the observed issues. Subsequently, OCEs analyze each incident by jointly leveraging the user-provided problem descriptions and vehicle logs uploaded to the cloud. Based on this information, they assign the incident to the appropriate category [2, 3, 22, 29, 33, 35] and provide corresponding resolution suggestions. These responses then undergo manual quality assurance before the validated responses are delivered to users through operations teams. Through this systematic workflow, service providers aim to promptly identify and resolve user-reported issues, thereby improving the overall user experience. Within this process, incident triage plays a critical role, as it ensures accurate triage and effective assignment of tickets to appropriate handling pathways, ultimately enhancing both the efficiency and reliability of the ticket management system.

3 MOTIVATION

The ICV ticket management system of Huawei receives a large volume of incident tickets on a daily basis, posing significant challenges to the efficiency of OCEs in resolving issues. Existing incident triage methods primarily rely on textual information within tickets, including a title and discussions [3, 22, 29]. However, in our scenario, the system does not support iterative engineer discussions, leaving no such discussion content available. Consequently, accurate and reliable incident triage becomes difficult when relying solely on short and often ambiguous user-reported problem descriptions. To mitigate the limitations of insufficient textual information, prior studies have incorporated system logs to enhance triage performance. Nevertheless, these approaches typically rely on simple log selection mechanism [33] that are not guided by user queries. As a result, they may retrieve irrelevant logs, introducing noise into LLM-based reasoning and analysis, or miss critical evidence, making them unsuitable for our scenario.

To address these challenges, we propose an end-to-end automated incident triage system that leverages the reasoning capabilities of LLMs to jointly analyze user problem descriptions and vehicle logs. Our method provides transparent and explainable triage decisions, thereby improving both the accuracy and trustworthiness of the results for OCEs. However, it still faces the following two challenges: insufficient domain knowledge, and the context length limitations of LLMs.

3.1 Challenge 1: Insufficient Domain Knowledge

Modern ICV systems are characterized by complex architectures composed of numerous interdependent components, which are typically developed by different teams. Effective incident triage requires a comprehensive understanding of the entire system. However, in real-world scenarios, such domain knowledge is often incomplete and lacks systematic organization. Given the large scale and high heterogeneity of these systems, it is challenging for individual engineers to fully master the details of all components, making domain experts a potential bottleneck in the incident triage process.

This challenge is further amplified in LLM-assisted automated incident triage. On the one hand, LLMs are typically trained on generic plain text corpora and therefore lack sufficient understanding of domain-specific knowledge [11, 27], which limits their effectiveness in real-world ICV operations scenarios. On the other hand, they cannot directly access or leverage engineers' implicit experiential knowledge. In the absence of explicit domain priors, LLMs struggle to reliably identify relevant system components and their associated logs for a given user issue. For example, when a user reports "unable to adjust the seat position", an LLM may only extract superficial keywords such as "seat adjustment" or "voice command" for log retrieval, without understanding the underlying system structure (e.g., the seat control components such as CarMDS). Consequently, it tends to blindly explore the entire log space without adequate guidance, retrieving either irrelevant or insufficient evidence. This not only introduces noise but also leads to inefficient analysis, ultimately degrading both the accuracy and efficiency of incident triage.

To address this, we propose an LLM-based operational knowledge mining approach that automatically extracts and constructs component-level structured knowledge base from historical vehicle logs. This approach yields a more comprehensive and systematic representation of domain knowledge, thereby improving the efficiency and accuracy of incident triage.

3.2 Challenge 2: Context Length Limitations of LLMs

Log data serves as a fundamental source of evidence for OCEs in incident triage. However, ICV systems operate at a large scale and generate massive volumes of logs. When applying LLMs to log analysis, the limited context length becomes a critical constraint that must be carefully addressed [23]. In practice, most log entries are irrelevant to a given incident. Directly feeding such large-scale and noisy log data into LLMs is not only computationally expensive and time-consuming, but may also weaken or hide key signals associated with the target incident, thereby degrading triage performance. Therefore, effectively identifying and extracting the subset of logs that are truly relevant to a specific incident is a necessary prerequisite for reliable LLM-based analysis.

To address this, we propose a log retrieval method based on large-scale log-query contrastive pretraining, which learns semantically representations of log data to enable efficient retrieval. By retrieving and retaining only the most relevant log entries for each incident before LLM processing, our approach mitigates the context length limitation and improves the effectiveness of incident triage.

4 METHODOLOGY

4.1 Overview

In this section, we introduce the details of *InsightTriage*. Figure 2 illustrates the overall framework of *InsightTriage*, which consists of two phases: an offline phase and an online phase. In the offline phase, *InsightTriage* leverages an LLM to analyze historical vehicle log data and construct a structured knowledge base of system components, which facilitates subsequent LLM-based reasoning for incident triage tasks. In the online phase, upon receiving an incident ticket, the FailureAgent autonomously selects and invokes two types of

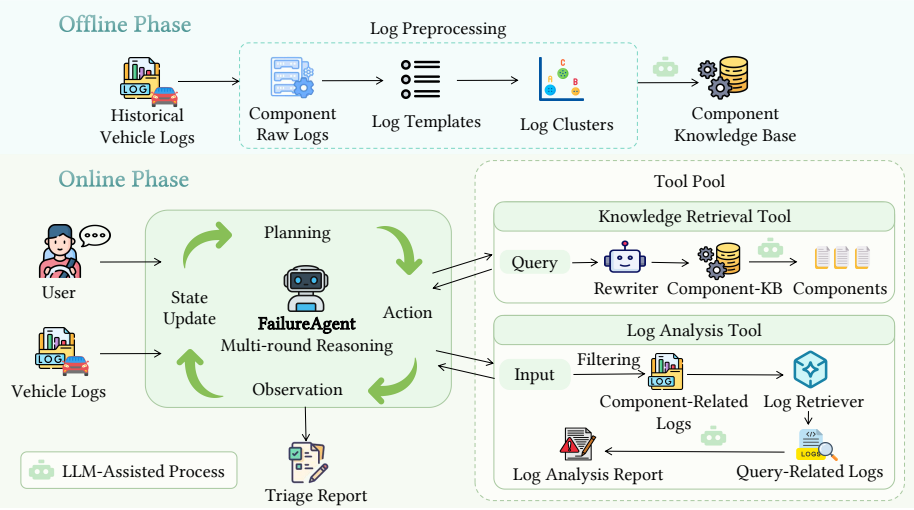


Figure 2: The framework of *InsightTriage*.

tools, namely knowledge retrieval and log analysis. It performs multi-round iterative exploration to progressively examine relevant log information. Based on the accumulated evidence, it ultimately produces an explainable incident triage report that includes both supporting log evidence and the corresponding triage analysis.

4.2 Offline Phase

To enable effective incident triage in large-scale ICV scenarios, a profound understanding of complex system behaviors is essential. Such knowledge bases are typically constructed from domain-specific documentation [9]. However, in our scenario, these resources are often unavailable due to a fragmented development paradigm in which different system components are built by independent teams. Moreover, manually constructing a unified knowledge base through cross-team collaboration is impractical. Therefore, *InsightTriage* introduces an LLM-assisted pipeline to autonomously construct a structured component-level knowledge base from large-scale historical vehicle logs. The construction process consists of two main stages: log preprocessing and knowledge base construction.

4.2.1 Log Preprocessing. To improve the quality of component-level knowledge items, we first preprocess the raw logs to extract clean, structured, and information-rich key log content.

Log Parsing. We collect a large volume of raw log data from a historical database. First, we parse the logs using regular expressions and partition them according to the component field in each log entry, forming an initial sequence of log messages. However, due to the limited context window of LLMs and the fact that component knowledge base construction mainly relies on key log content, raw logs contain a large amount of runtime parameters and other information unrelated to component functionality. Such information may interfere with subsequent knowledge extraction. Therefore, it is necessary to further structure the raw logs to extract stable semantic patterns. Specifically, we adopt the classical log parsing method Drain [12] to extract log templates and parameters from the log data. The extracted log templates are then used as the basis for constructing the component knowledge base.

Log Clustering. To further reduce log template volume while retaining representative log patterns, we adopt an unsupervised clustering approach to group similar log templates. This allows subsequent knowledge extraction to focus on a compact set of representative log templates rather than the entire corpus. Specifically, given N log templates, we first construct an $N \times N$ distance matrix using the Jaccard distance, which measures pairwise syntactic-level similarity between log templates based on token-set overlap [20]. Based on the distance matrix, we apply the DBSCAN [7] algorithm to cluster log templates into coherent clusters. This algorithm does not require a predefined number of clusters and is defined by two key parameters: the neighborhood radius ε and the minimum number of points required to form a valid cluster, *MinPts*. It is capable of discovering clusters of arbitrary shapes and is robust in the presence of noise. Finally, for each cluster, we select a representative log template defined as the one with the minimum average distance to all other log templates within the same cluster. This representative log template preserves the shared syntactic-level characteristics of the cluster, while redundant templates are removed to provide a clean and non-redundant input for component knowledge extraction.

4.2.2 Knowledge Base Construction. To support efficient and accurate incident triage, *InsightTriage* constructs a structured component-level knowledge base to facilitate online triage reasoning. Building upon the preprocessing stage, this process focuses on extracting diagnostic-relevant knowledge items from the processed log templates and organizing them into a unified representation. The overall construction workflow consists of two key steps.

Step 1: Predefinition of the Component Knowledge Schema. To ensure that the extracted knowledge is both well-structured and effective for triage, *InsightTriage* constrains knowledge representation within a predefined diagnosis ontology. Specifically, we abstract the key elements of incident triage into the following concept set: $Concepts = \{Component, Function\ Name, Function\ Description,$

Keywords}. Among these concepts, *Component* serves as the central entity, while the other elements characterize it from different semantic perspectives. *Function Name* provides a concise label describing the component's capabilities. *Function Description* offers a more detailed explanation of its behavior and responsibilities. And *Keywords* capture representative phrases or high-frequency terms from logs that reflect the component's operational characteristics. This unified schema establishes a standardized representation for subsequent knowledge extraction. Figure 3 illustrates an example of the constructed knowledge item for the CCNService component.

Component: CCNService

Function Name: Vehicle Connectivity and Communication Management Service
Function Description: CCNService (Car Connectivity Network Service) is a core communication control and service component in intelligent vehicles. It is primarily responsible for in-vehicle network connection management, TBox service control, and network diagnostics. This component integrates multiple network management capabilities, including the control and configuration of priority networks, Wi-Fi, and cellular networks, ensuring the stability and reliability of vehicle connectivity. Its core functionalities include:

1. Network Connection Management: Monitors and manages the status of in-vehicle network interfaces, including Ethernet, Wi-Fi, and cellular interfaces, and handles network changes as well as disconnection and reconnection logic.
2. TBox Service Control: Manages the communication module of the in-vehicle infotainment system, including service binding, attribute subscription, and callback handling.

...

Keywords: CCNStateMachine, CCNCheckUtil, Vehicle.TBox.WifiAP.WifiAPLogin, ...

Figure 3: Example of component knowledge case.

Step 2: Extraction of Component Knowledge from Log Data. After log preprocessing, we feed component-wise aggregated key log content into an LLM to extract structured knowledge in accordance with the predefined schema. Specifically, for each component, its clustered representative log templates are provided as contextual input, and a carefully designed prompt template is employed to guide the LLM in summarizing its functionality and operational characteristics. As illustrated in Figure 4, the prompt instructs the model to generate the corresponding function name, function description, and keywords based on the input log data.

To further ensure adaptability in real-world production environments with frequent system updates, the knowledge base can be periodically incrementally maintained, with newly introduced components constructed through the same extraction pipeline.

4.3 Online Phase

In this subsection, we present the online phase of *InsightTriage*, which focuses on the design and orchestration of tools for the incident triage task. *InsightTriage* adopts a modular architecture, facilitating its extensibility across diverse systems. During this phase, multiple tools are coordinated under the ReAct paradigm [34], which progressively narrows the problem scope and facilitates effective incident triage.

4.3.1 Knowledge Retrieval Tool. In real-world scenarios, user-reported problem descriptions are typically based on observable failure symptoms, which often exhibit a semantic gap from underlying component-level knowledge. Moreover, user queries are frequently incomplete, ambiguous, or noisy, which further hinders effective component knowledge retrieval.

Role
You are an expert in log analysis for intelligent vehicle systems, equipped with a strong understanding of common issues in this domain, advanced log analysis capabilities, excellent technical writing skills, and comprehensive knowledge of intelligent vehicle technologies.

Task
Given a component name and its corresponding log templates, analyze and summarize the core functionality of the component. Please complete the task by following these steps:

1. Step 1: Semantic Understanding and Key Information Extraction
- Based on the component name and log templates, identify key log content that reflect the component's functionality
- Extract the most representative behavioral patterns and semantic signals

2. Step 2: Noise Filtering and Information Refinement
- Remove low-value or irrelevant information
- Ensure the remaining information is concise, clear, and informative

3. Step 3: Function Summarization and Structured Representation
- Summarize the core functionality and responsibilities of the component based on the extracted information.
- Use concise, precise, and unambiguous language, and avoid introducing any assumptions or information not grounded in the log templates.
- Please strictly follow the JSON format below: {"Function Name": "", "Function Description": "", "Keywords": ["", ""]}

Example
...

Component Name
[Component name]

Log Templates
[The log template sequence of this component]

Figure 4: The prompt for component knowledge extraction.

To address this challenge, we adopt a hypothesis-generation-based retrieval strategy [8]. Specifically, an LLM is first employed to rewrite and expand the user query into multiple hypothetical documents that describe potential causes of the incident from different perspectives. The generated hypotheses are then used as augmented queries to retrieve semantically relevant component knowledge entries from the component knowledge base via similarity-based search, yielding a top N candidate set (with N set to 50).

Subsequently, the LLM is further employed to refine the retrieved top N candidates by jointly analyzing the user query and the retrieved knowledge entries. The model evaluates whether the functionality of each component could potentially lead to or influence the observed failure symptoms. Based on this symptom–cause reasoning, the LLM identifies the most relevant subset of components, producing the final top K knowledge entries (with K set to 5).

4.3.2 Log Analysis Tool. After obtaining component knowledge relevant to the user query via the knowledge retrieval tool, the FailureAgent further invokes the log analysis tool to perform component-level log analysis. This tool leverages the natural language understanding and reasoning capabilities of LLMs to support log-based diagnosis. However, when applying LLMs to log analysis, context-length limitations remain a critical challenge. To mitigate this, it is essential to select query-relevant log entries so that the model can focus on the most informative evidence within a limited context window. Inspired by prior work on code search [19], *InsightTriage* adopts a contrastive pretraining strategy to train a log retriever using large-scale log-query pairs, enabling efficient identification of log entries that are most relevant to the user query. Overall, we describe the design of the log analysis tool from three key perspectives: data construction, model training, and online usage.

Data Construction. During the construction of the component knowledge base, we obtain semantic annotations for a large number of log fragments. Based on these annotations, we construct training pairs (l, d), where l denotes a raw log snippet and d represents its corresponding semantic description. This process yields a large-scale dataset for training the log retriever.

Model Training. We train the log retriever using a contrastive learning objective to align user queries with relevant logs in a shared embedding space. To enhance semantic understanding in Chinese scenarios, we adopt Qwen3-Embedding-0.6B [40] as the embedding model. For a given query embedding q_i, its corresponding positive log sample is denoted as l_i^+, while other log samples within the same batch are treated as negative examples. The optimization objective is defined as following:

$$\mathcal{L}_i = -\log \frac{\exp\left(s(q_i, l_i^+)/\tau\right)}{\sum_{l \in C} \exp\left(s(q_i, l)/\tau\right)} \tag{1}$$

where $s(\cdot,\cdot)$ denotes the similarity function, τ is the temperature coefficient, and C represents the set of candidate log samples within the batch. This objective encourages the model to minimize the distance between positive log-query pairs while maximizing the separation from negative samples, thereby learning a discriminative embedding space for log retrieval.

Online Usage. During online usage, the trained log retriever computes similarity scores between the user query and all candidate log entries, and retrieves the top N most relevant ones. These retrieved logs are then fed into an LLM for further semantic analysis and anomaly summarization, enabling the generation of component-level log analysis report. The resulting outputs are subsequently returned to the FailureAgent for further analysis.

4.3.3 Overall Workflow. The FailureAgent serves as the central controller in the online phase and adopts the ReAct paradigm to enable iterative reasoning with tool interaction. Given an incident ticket and system logs as input, the agent operates in an iterative multi-step reasoning loop. Each iteration consists of four stages: state update, planning, action execution, and observation. In each iteration, the agent first updates its state by integrating newly acquired information with historical context. It then performs planning to decide whether to invoke external tools (e.g., a knowledge retrieval tool or log analysis tool) for additional evidence. Based on this decision, the agent executes the corresponding action, and finally incorporates the observed results into the next iteration of reasoning. To support reliable triage reasoning, the agent is equipped with reflection-oriented prompts that enable it to assess whether sufficient evidence has been collected for high-confidence triage or whether further exploration is required. Based on this self-reflection, the agent dynamically decides to continue tool use or terminate the process and return a triage result. A maximum iteration budget (set to 15) is enforced to prevent excessive exploration, after which the agent generates a final triage report based on the accumulated evidence.

Through this closed-loop interaction process, the agent progressively aggregates evidence from both structured knowledge and unstructured logs, reduces uncertainty, and refines its understanding of the incident. Finally, grounded in the observed log evidence, the FailureAgent generates an accurate and explainable incident triage report. This aids OCEs in comprehending the triage results.

5 EVALUATION

In this section, we conduct a comprehensive evaluation of *InsightTriage* to answer the subsequent research questions (RQs):

- **RQ1:** How does *InsightTriage* perform in incident triage?
- **RQ2:** Does each component of *InsightTriage* contribute significantly to *InsightTriage*'s performance?
- **RQ3:** How effective are the triage reports of *InsightTriage* in terms of readability and usefulness?

5.1 Experiment Setup

5.1.1 Dataset. To comprehensively evaluate the performance of *InsightTriage*, we collect 4,526 incident tickets from the real-world production environment of Huawei, which serves millions of ICVs, over the past several months, covering 73 major incident types. Among them, 3922 tickets from Q3 2025 to Q4 2025 are used to construct the knowledge base and for model training, while the remaining 604 tickets from the most recent three months are used to evaluate the triage performance.

5.1.2 Implementation Details. We conduct all the experiments with eight NVIDIA V100 GPU. Considering the trade-off between cost and Chinese language understanding capability, we adopt GLM-4.5-Air [10] as the backbone model for both knowledge base construction and reasoning. In addition, we use Qwen3-Embedding-0.6B [40] to train the log retrieval model.

5.1.3 Evaluation Metrics. Following prior work [22, 29], we adopt widely used evaluation metrics, including Precision, Recall, Macro F1-Score, Micro F1-Score, and Weighted F1-Score, to evaluate incident triage performance. Precision measures the proportion of correctly assigned tickets among all predicted assignments: Precision = $\frac{TP}{TP+FP}$. Recall measures the ability of the model to identify all relevant tickets for each category: Recall = $\frac{TP}{TP+FN}$. Macro F1-Score evaluates triage performance by averaging per-category F1-Scores equally across all incident types: Macro F1 = $\frac{1}{N}\sum_{i=1}^{N} \mathrm{F1}_i$. Micro F1-Score assesses overall triage effectiveness by aggregating all correctly and incorrectly triaged tickets across categories: Micro F1 = $\frac{2 \cdot TP_{\text{all}}}{2 \cdot TP_{\text{all}} + FP_{\text{all}} + FN_{\text{all}}}$. Weighted F1-Score accounts for class imbalance by weighting each category's F1-Score by its support in the triage dataset: Weighted F1 = $\sum_{i=1}^{N} w_i \cdot \mathrm{F1}_i$.

To evaluate the quality of the triage reports, we adopt a human evaluation. Specifically, we conduct human participants to rate each report using a 5-point Likert scale from two objective criteria: Readability and Usefulness, as shown in Table 1.

- **Readability**: Measures the clarity and quality of the triage report, including linguistic fluency, structural organization, and ease of understanding.
- **Usefulness**: Evaluates the practical value of the report for incident triage in terms of user intent alignment, log-grounded evidence, and root cause correctness.

5.1.4 Baseline Approaches. To evaluate incident triage performance of *InsightTriage*, we adopt the following methods.

Table 1: Criteria for readability and usefulness evaluation.

Score	Readability	Usefulness
1	The report is difficult to understand, poorly structured, and contains unclear and verbose expressions.	The report does not address the user query, lacks log evidence, and is irrelevant to the incident.
2	The report is partially understandable but contains redundancy and ambiguity that reduce clarity.	The report partially addresses the user query but provides very limited log evidence and weak diagnostic reasoning.
3	The report is generally understandable but somewhat disorganized and lacking in coherence.	The report addresses the user query at a basic level with some log evidence, but analysis is superficial.
4	The report is clear and well-structured with only minor linguistic or organizational issues.	The report effectively addresses the user query, supported by relevant log evidence, with coherent reasoning.
5	The report is highly clear, concise, well-structured, and easy to understand with coherent logical flow.	The report fully addresses the user query, is strongly grounded in log evidence, and provides comprehensive reasoning.

Table 2: Performance comparison of triage methods.

Method	Precision	Recall	Macro F1	Micro F1	Weighted F1	Avg.Tokens
DeepTriage [29]	0.582	0.529	0.417	0.529	0.500	-
COMET [33]	0.244	0.243	0.169	0.243	0.211	12.8k
TickIt [22]	0.727	0.668	0.669	0.668	0.662	9.3k
Triangle [35]	0.709	0.681	0.654	0.681	0.664	76.8k
InsightTriage	**0.842**	**0.805**	**0.801**	**0.805**	**0.801**	140.7k

- **DeepTriage [29]:** DeepTriage ensembles several submodels, including a multiple additive regression tree, a light gradient boosting machine, an inverted index, a locality-sensitive hashing model, and a deep neural network.
- **COMET [33]:** COMET selects crucial log information, utilizes LLMs with domain knowledge for keywords extraction, and performs similarity-based triage using embeddings generated by a fine-tuned FastText model.
- **TickIt [22]:** TickIt adopts a Chain-of-Thought reasoning paradigm combined with retrieval-augmented In-Context Learning, leveraging LLMs to achieve precise multi-category ticket triage.
- **Triangle [35]:** Triangle is an end-to-end multi-agent incident triage system that leverages semantic distillation and multi-role agent negotiation with team information enhancement.

5.2 RQ1: The Overall Performance

The comparative results presented in Table 2 clearly indicate that *InsightTriage* consistently outperforms other baseline methods, which achieves a Weighted F1-Score of 0.801. Specifically, *InsightTriage* surpasses the SOTA approach by 13.7% in Weighted F1-Score. The poor performance of DeepTriage can be attributed to its reliance on traditional machine learning sub-models with shallow feature representations, which limits their ability to capture complex contextual semantics. Although COMET leverages LLMs and domain knowledge to extract keywords from TrimmedLogs for incident triage, it remains ineffective in our scenario. Its rule-based log filtering relies on predefined fault keywords (e.g., "crash"), which is unsuitable for our symptom-driven setting, where issues are highly diverse and cannot be captured by a finite and stable keyword set. For instance, an issue such as "QQ Music loses focus unexpectedly" may correspond to "qq.live", while "Video loss caused by encoding errors" relate to "deleteOldestVideo". In addition, its frequency-based log selection mechanism is ineffective in this setting, as the relevance of important logs is independent of their occurrence frequency. TickIt's effectiveness is constrained by its lack of log analysis, which consequently limits its incident triage capability. While Triangle relies on multi-role negotiation for triage, overconfidence from individual expert agents during the multi-agent discussion may bias the final decision and degrade triage accuracy. In contrast, *InsightTriage* achieves the best performance by extracting structured system knowledge from historical logs and training a log retriever to identify query-relevant log entries. By jointly modeling user queries and retrieved logs through LLM-based reasoning, it significantly improves both incident triage accuracy and robustness.

Moreover, we present a comparison of token consumption between our method and baseline methods. Although *InsightTriage* incurs higher token usage of 140.7k than other baselines, this overhead represents a necessary trade-off to improve triage accuracy. By performing deep semantic analysis on logs, *InsightTriage* can generate more explainable and reliable triage results, thereby substantially improving triage accuracy. In real-world production environment, the reliability and explainability of triage results are often prioritized over short-term computational cost.

5.3 RQ2: Ablation Study

To comprehensively assess the contribution of each core component in *InsightTriage*, we conducted a systematic ablation study. The results, presented in Table 3, focus on the effectiveness of two key modules: the component knowledge base, and the log retriever.

5.3.1 Effectiveness of Component Knowledge Base. Removing the component knowledge base from the full framework leads to a

Table 3: Effectiveness of each component in *InsightTriage*.

Method	Precision	Recall	Macro F1	Micro F1	Weighted F1	Avg.Tokens
InsightTriage	0.842	0.805	0.801	0.805	0.801	140.7k
-Component Knowledge Base	0.720	0.677	0.679	0.677	0.669	138.2k
-Log Retriever	0.690	0.613	0.630	0.613	0.609	146.8k

significant performance degradation. As shown in Table 3, the Weighted F1-Score drops from 0.801 to 0.669. This indicates that, without domain knowledge guidance, the LLM struggles to accurately capture complex vehicle system behaviors and becomes more prone to hallucinations during the reasoning process. Moreover, the absence of the knowledge base limits the depth of model reasoning, resulting in incomplete analyses. In contrast, *InsightTriage* equipped with the component knowledge base enables more comprehensive reasoning. Although this design increases token consumption, it significantly improves triage accuracy.

5.3.2 Effectiveness of Log Retriever. The log retriever improves incident triage by selecting query-relevant log entries from raw logs. As shown in Table 3, removing this module leads to a 19.2% drop in Weighted F1-Score, while Macro F1-Score and Micro F1-Score decrease by 17.1% and 19.2%, respectively. Meanwhile, Avg. Tokens increase from 140.7k to 146.8k. Without the retriever's filtering mechanism, a large amount of redundant and low-relevance log entries are directly fed into the LLM, resulting in an unnecessarily expanded context. This not only increases inference costs but also amplifies the impact of noisy information on model predictions. These results demonstrate that the log retriever improves accuracy through effective noise reduction and key information extraction, while also enhancing efficiency by compressing the input context.

5.4 RQ3: Triage Report Quality

Table 4: Human evaluation of *InsightTriage*'s quality.

Method	Readability	Usefulness
InsightTriage	4.30	3.87

InsightTriage leverages the reasoning and understanding capabilities of LLM to generate structured triage reports grounded in log evidence, by jointly incorporating user queries and system logs. Guided by prompt engineering and a domain-specific knowledge base, *InsightTriage* can effectively capture user intent and extract relevant log evidence, thereby improving the reliability of incident triage. To evaluate report quality, we randomly sample 100 cases covering all incident categories. Following the criteria defined in Table 1, we conduct a human evaluation in which each report is rated along two dimensions: Readability and Usefulness.

As shown in Table 4, the reports achieve an average Readability score of 4.30, indicating that they are well-structured and easy to understand. The average Usefulness score reaches 3.87, suggesting that the reports provide accurate root cause analysis and are well supported by faithful and relevant log evidence.

6 Deployment

6.1 Workflow of Deployment

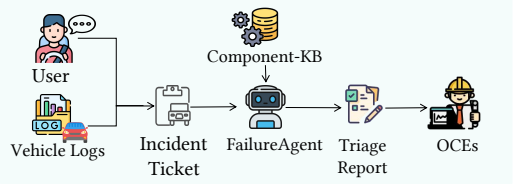


Figure 5: The deployment workflow of *InsightTriage*.

InsightTriage has been deployed in the production, where it serves as a triage assistant for incident management. The deployment workflow is illustrated in Figure 5. For each user-reported ticket, the FailureAgent first analyzes the reported issue and initiates the diagnostic process by querying the component knowledge base via a knowledge retrieval tool to establish an initial investigation direction. It then autonomously selects and invokes both log analysis and knowledge retrieval tools to perform iterative and in-depth triage analysis. Finally, the system produces an explainable triage report supported by concrete log evidence, which is provided to OCEs to facilitate effective incident triage. Additionally, based on token usage statistics, *InsightTriage* consumes an average of 119.6k prompt tokens and 21.1k response tokens per incident. Based on the official pricing [26], we estimate that the cost per incident for triage is approximately $0.033, demonstrating its practical applicability in real-world production environments.

6.2 Case Study

To comprehensively demonstrate the effectiveness of *InsightTriage*, we present a detailed case study. To maintain strict corporate data privacy standards, all sensitive information has been excluded from the description.

As illustrated in Figure 6, a user reports an incident in which "the vehicle suddenly started playing music without receiving any command." Upon receiving the user report and raw vehicle logs, the FailureAgent begins reasoning and progressively invokes external tools to identify the root cause. In the initial stage, the FailureAgent identifies potentially relevant system components by invoking the Knowledge Retrieval Tool. A query related to "automatic music playback" returns several relevant components, including WakeupService and WakeupEngineWrapper. Based on the retrieved domain knowledge, the agent proceeds to the next stage, where it invokes the Log Analysis Tool to conduct a targeted analysis of logs associated with WakeupService. Subsequently, the FailureAgent conducts a sequence of exploratory analyses, iteratively refining its hypotheses through multi-step reasoning and tool interaction.

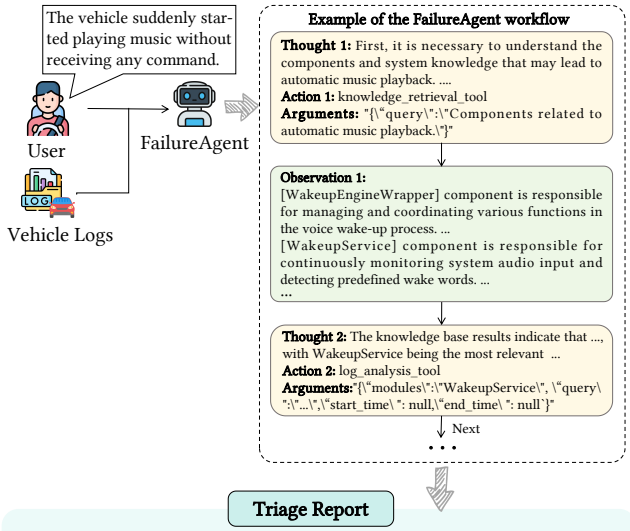


Figure 6: The workflow of an example.

Based on the accumulated evidence, the FailureAgent ultimately generates a structured triage report. As shown in the Triage Report, the analysis identifies a logic flaw occurring at 06:37:58, where the system incorrectly interprets the media control command "previous song" as a wake-word trigger. The FailureAgent further correlates this anomaly with the subsequent invocation of the NetEase Cloud Music application via DTE_MsWrapper. By integrating these observations, the FailureAgent identifies the root cause as the wake-word shortcut "previous song" incorrectly triggering the music app, which activates media actions in the absence of a valid wake state.

This case study demonstrates that *InsightTriage* can effectively integrate user-reported query with relevant log evidence to perform automated incident triage, while generating an explainable triage report that enhances engineers' confidence in the results.

6.3 Lessons Learned

6.3.1 Importance of Semantic-Driven Log Analysis. In our scenario, issues often lack explicit error signatures. For instance, a user may report an inability to adjust the seat, while system logs contain no explicit error messages. Instead, triage evidence is implicitly reflected in contextual states, such as whether the vehicle is in a non-P gear that restricts seat control. As a result, traditional pattern-matching or anomaly detection approaches become ineffective [6, 17, 25]. To address this, it is necessary to leverage LLMs to semantically interpret user-reported problem descriptions and capture triage intents. Guided by these intents, the system can proactively explore logs and extract relevant evidence based on semantic cues. This semantics-driven log analysis paradigm is better suited for handling complex and uncertain behavioral issues in real-world production environments.

6.3.2 Structured Domain Knowledge. A substantial semantic gap exists between user-reported symptoms and low-level system logs. Experimental results indicate that relying solely on the implicit reasoning capabilities of LLMs is insufficient to achieve stable cross-layer semantic alignment and effective incident triage. In contrast, component-level structured domain knowledge automatically constructed from historical logs provides explicit priors of system behavior, thereby significantly improving both the accuracy and stability of incident triage.

7 DISCUSSION

7.1 Limitations

Our study explores the potential of jointly analyzing user-reported incident and system logs for incident triage and identifies two major limitations.

7.1.1 Limitation in Handling Image and Video-based Data. The current approach primarily focuses on text-driven incident triage and does not incorporate multimodal information such as images or videos. However, in real-world ticketing scenarios, users may report issues through images or videos, which contain rich and critical contextual cues. The lack of multimodal modeling capabilities limits the system's ability to fully utilize non-textual evidence when handling complex tickets, which may result in incomplete semantic understanding and reduced triage accuracy. To address this limitation, future work could explore the integration of multimodal LLMs to build a unified cross-modal reasoning framework. By incorporating image understanding and video analysis tools, the system can enable collaborative modeling of multimodal information, thereby improving performance in complex real-world scenarios.

7.1.2 Performance Limitations and Resource Overhead in Multi-round Reasoning Loops. The core of our framework relies on iterative Reasoning–Action–Observation loops to progressively refine incident triage. While this design improves both accuracy and robustness, it also introduces non-negligible computational overhead due to multiple LLM inferences and repeated tool invocations. In time-critical industrial environments, such cumulative latency may hinder real-time responsiveness. Future work could explore distilling the reasoning capabilities into smaller, specialized models to reduce inference latency while preserving triage accuracy.

7.2 Threats to Validity

7.2.1 Internal threats. *InsightTriage* relies on LLM-based reasoning for incident triage, which is susceptible to hallucinations in complex ICV scenarios, thereby potentially resulting in incorrect triage decisions. To mitigate this issue, we design prompt templates for the FailureAgent and introduce a structured component knowledge base (detailed in Section 4.2). This design requires the FailureAgent to explicitly ground its reasoning in retrieved component knowledge and relevant log information at each reasoning loop (detailed in Section 4.3). Such a fact-constrained reasoning mechanism helps reduce hallucinated content. Experimental results show that this approach effectively mitigates this threat.

7.2.2 External threats. The threat to external validity lies in the extent to which our experimental results can be generalized. We evaluated *InsightTriage* only on a real-world production dataset from the Huawei ICV ticket management system. Nevertheless, we believe that the proposed method exhibits strong transferability. Specifically, *InsightTriage* adopts a modular design, and its core reasoning-and-execution paradigm is highly generalizable. This allows the framework to be easily adapted to triage scenarios in other organizations by reconstructing the component-level knowledge base and applying lightweight adaptations to relevant modules. In future work, we plan to further evaluate and enhance the generalizability and robustness of the system in more diverse real-world operational environments.

8 RELATED WORK

8.1 Incident Triage

Recent research on incident triage has been extensive, focusing on improving the efficiency and accuracy of automated ticket assignment through the integration of both textual and non-textual features. For example, [16] enhances triage accuracy by employing an ensemble approach that integrates multiple diverse classifiers, including Naive Bayes, SVM, KNN, and Decision Trees. DeepCT [3] formulates triage as an incremental learning process, using a gated recurrent unit to capture temporal dependencies in discussions, applying attention mechanisms to reduce noise, and optimizing stage-wise predictions via a continuous loss function. Additionally, [2] leverages deep learning techniques to further improve incident triage performance. DeepTriage [29] integrates multiple machine learning methods to construct an ensemble model, enabling automated and accurate incident triage and significantly reducing incident mitigation time. COMET [33] employs AutoExtractor to preprocess large-scale logs, leverages domain knowledge with LLMs to extract key information, and applies embedding-based similarity retrieval for efficient and accurate incident triage in cloud service environments. TickIt [22] proposes an LLM-based automated online ticket escalation framework that enables accurate and topic-aware ticket escalation and triage by continuously updating ticket states, exploring ticket correlations, and leveraging category-guided supervised fine-tuning to improve model performance. Triangle [35] introduces a multi-agent collaborative incident triage system that achieves efficient and accurate incident triage through semantic distillation, multi-role agent collaboration with negotiation mechanisms, and team knowledge augmentation.

Despite these advancements, existing incident triage methods still struggle to effectively incorporate logs into comprehensive analysis. To address this limitation, we propose *InsightTriage*, which introduces knowledge-guided exploratory reasoning. By integrating automatically generated domain knowledge with semantic-driven log analysis, *InsightTriage* enables more explainable and robust incident triage in complex and non-deterministic scenarios.

8.2 LLMs in Cloud System Maintenance

With the advancement of LLMs, many studies have explored their applications in various aspects of cloud incident management by leveraging their strong capabilities in natural language understanding and reasoning [1, 4, 15, 28, 32, 38, 39, 42]. For example, D-Bot [42] leverages knowledge extraction, prompt generation, tree-structured reasoning, and multi-agent collaboration to effectively identify the root causes of database anomalies. Nissist [1] constructs a structured knowledge base from unstructured troubleshooting guides and incident mitigation history, and employs a multi-agent system for semi-automated incident handling, thereby significantly reducing mitigation time. mABC [38] is a multi-agent collaborative framework that integrates a standardized agent workflow with a blockchain-inspired voting mechanism, enabling efficient root cause analysis and resolution. Flow-of-Action [28] is an SOP-enhanced LLM multi-agent system that employs standardized process constraints, action sets, and multi-agent collaboration to mitigate LLM hallucinations in root cause analysis. Unlike previous approaches, *InsightTriage* is the first to integrate automatically constructed structured knowledge from historical log data with LLM-enhanced log reasoning, enabling efficient incident triage.

9 CONCLUSION

In this paper, we propose *InsightTriage*, an LLM-enhanced framework for automated and explainable incident triage in ICV scenarios. By automatically mining historical vehicle logs and constructing structured component-level domain knowledge, *InsightTriage* builds a comprehensive understanding of the overall ICV system, thereby providing more systematic knowledge support for incident triage. Meanwhile, we employ log–query contrastive pretraining to learn semantic representations of logs, enabling efficient retrieval of key log evidence relevant to user queries and alleviating the context length limitations of LLMs in log analysis. Furthermore, *InsightTriage* introduces an LLM-driven reasoning module that jointly models user queries, retrieved log evidence, and domain knowledge to generate structured and explainable triage reports, thereby improving both the readability and usefulness of triage results. Extensive experiments conducted on a ticket datasets from the production environment of Huawei demonstrate that *InsightTriage* significantly outperforms all state-of-the-art methods, achieving a Weighted F1-Score of 0.801. These results demonstrate its effectiveness as an automated and practical solution for incident triage.

10 Data Availability Statement

Due to privacy concerns related to user-reported ticket data, as well as company confidentiality and compliance policies, the dataset and source code used in this study cannot be made publicly available.

References

[1] Kaikai An, Fangkai Yang, Junting Lu, Liqun Li, Zhixing Ren, Hao Huang, Lu Wang, Pu Zhao, Yu Kang, Hua Ding, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. 2024. Nissist: An Incident Mitigation Copilot based on Troubleshooting Guides. In *ECAI 2024 - 27th European Conference on Artificial Intelligence, 19-24 October 2024, Santiago de Compostela, Spain - Including 13th Conference on Prestigious Applications of Intelligent Systems (PAIS 2024) (Frontiers in Artificial Intelligence and Applications)*, Ulle Endriss, Francisco S. Melo, Kerstin Bach, Alberto José Bugarín Diz, Jose Maria Alonso-Moral, Senén Barro, and Fredrik Heintz (Eds.). IOS Press, 4471–4474. doi:10.3233/FAIA241032

[2] Junjie Chen, Xiaoting He, Qingwei Lin, Yong Xu, Hongyu Zhang, Dan Hao, Feng Gao, Zhangwei Xu, Yingnong Dang, and Dongmei Zhang. 2019. An empirical investigation of incident triage for online service systems. In *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice, ICSE (SEIP) 2019, Montreal, QC, Canada, May 25-31, 2019*, Helen Sharp and Mike Whalen (Eds.). IEEE / ACM, 111–120. doi:10.1109/ICSE-SEIP.2019.00020

[3] Junjie Chen, Xiaoting He, Qingwei Lin, Hongyu Zhang, Dan Hao, Feng Gao, Zhangwei Xu, Yingnong Dang, and Dongmei Zhang. 2019. Continuous Incident Triage for Large-Scale Online Service Systems. In *34th IEEE/ACM International Conference on Automated Software Engineering, ASE 2019, San Diego, CA, USA, November 11-15, 2019*. IEEE, 364–375. doi:10.1109/ASE.2019.00042

[4] Yinfang Chen, Huaibing Xie, Minghua Ma, Yu Kang, Xin Gao, Liu Shi, Yunjie Cao, Xuedong Gao, Hao Fan, Ming Wen, Jun Zeng, Supriyo Ghosh, Xuchao Zhang, Chaoyun Zhang, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Tianyin Xu. 2024. Automatic Root Cause Analysis via Large Language Models for Cloud Incidents. In *Proceedings of the Nineteenth European Conference on Computer Systems, EuroSys 2024, Athens, Greece, April 22-25, 2024*. ACM, 674–688. doi:10.1145/3627703.3629553

[5] Zhuangbin Chen, Yu Kang, Liqun Li, Xu Zhang, Hongyu Zhang, Hui Xu, Yangfan Zhou, Li Yang, Jeffrey Sun, Zhangwei Xu, Yingnong Dang, Feng Gao, Pu Zhao, Bo Qiao, Qingwei Lin, Dongmei Zhang, and Michael R. Lyu. 2020. Towards intelligent incident management: why we need it and how we make it. In *ESEC/FSE '20: 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event, USA, November 8-13, 2020*, Prem Devanbu, Myra B. Cohen, and Thomas Zimmermann (Eds.). ACM, 1487–1497. doi:10.1145/3368089.3417055

[6] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. 2017. DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, Bhavani Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu (Eds.). ACM, 1285–1298. doi:10.1145/3133956.3134015

[7] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA*, Evangelos Simoudis, Jiawei Han, and Usama M. Fayyad (Eds.). AAAI Press, 226–231.

[8] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023. Precise Zero-Shot Dense Retrieval without Relevance Labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, 1762–1777. doi:10.18653/V1/2023.ACL-LONG.99

[9] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. 2023. Retrieval-Augmented Generation for Large Language Models: A Survey. *CoRR* abs/2312.10997 (2023). arXiv:2312.10997 doi:10.48550/ARXIV.2312.10997

[10] GLM. 2025. GLM-4.5: Agentic, Reasoning, and Coding (ARC) Foundation Models. *CoRR* abs/2508.06471 (2025). arXiv:2508.06471 doi:10.48550/ARXIV.2508.06471

[11] Yue Guo, Zian Xu, and Yi Yang. 2023. Is ChatGPT a Financial Expert? Evaluating Language Models on Financial Natural Language Processing. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, 815–821. doi:10.18653/V1/2023.FINDINGS-EMNLP.58

[12] Pinjia He, Jieming Zhu, Zibin Zheng, and Michael R. Lyu. 2017. Drain: An Online Log Parsing Approach with Fixed Depth Tree. In *2017 IEEE International Conference on Web Services, ICWS 2017, Honolulu, HI, USA, June 25-30, 2017*, Ilkay Altintas and Shiping Chen (Eds.). IEEE, 33–40. doi:10.1109/ICWS.2017.13

[13] Shilin He, Xu Zhang, Pinjia He, Yong Xu, Liqun Li, Yu Kang, Minghua Ma, Yining Wei, Yingnong Dang, Saravanakumar Rajmohan, and Qingwei Lin. 2022. An empirical study of log analysis at Microsoft. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2022, Singapore, Singapore, November 14-18, 2022*, Abhik Roychoudhury, Cristian Cadar, and Miryung Kim (Eds.). ACM, 1465–1476. doi:10.1145/3540250.3558963

[14] Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards Reasoning in Large Language Models: A Survey. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023 (Findings of ACL)*, Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, 1049–1065. doi:10.18653/V1/2023.FINDINGS-ACL.67

[15] Pengxiang Jin, Shenglin Zhang, Minghua Ma, Haozhe Li, Yu Kang, Liqun Li, Yudong Liu, Bo Qiao, Chaoyun Zhang, Pu Zhao, Shilin He, Federica Sarro, Yingnong Dang, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. 2023. Assess and Summarize: Improve Outage Understanding with Large Language Models. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2023, San Francisco, CA, USA, December 3-9, 2023*, Satish Chandra, Kelly Blincoe, and Paolo Tonella (Eds.). ACM, 1657–1668. doi:10.1145/3611643.3613891

[16] Leif Jonsson, Markus Borg, David Broman, Kristian Sandahl, Sigrid Eldh, and Per Runeson. 2016. Automated bug assignment: Ensemble-based machine learning in large scale industrial contexts. *Empir. Softw. Eng.* 21, 4 (2016), 1533–1578. doi:10.1007/S10664-015-9401-9

[17] Van-Hoang Le and Hongyu Zhang. 2021. Log-based Anomaly Detection Without Log Parsing. In *36th IEEE/ACM International Conference on Automated Software Engineering, ASE 2021, Melbourne, Australia, November 15-19, 2021*. IEEE, 492–504. doi:10.1109/ASE51524.2021.9678773

[18] Liqun Li, Xu Zhang, Xin Zhao, Hongyu Zhang, Yu Kang, Pu Zhao, Bo Qiao, Shilin He, Pochian Lee, Jeffrey Sun, Feng Gao, Li Yang, Qingwei Lin, Saravanakumar Rajmohan, Zhangwei Xu, and Dongmei Zhang. 2021. Fighting the Fog of War: Automated Incident Detection for Cloud Systems. In *Proceedings of the 2021 USENIX Annual Technical Conference, USENIX ATC 2021, July 14-16, 2021*, Irina Calciu and Geoff Kuenning (Eds.). USENIX Association, 131–146.

[19] Xiaonan Li, Yeyun Gong, Yelong Shen, Xipeng Qiu, Hang Zhang, Bolun Yao, Weizhen Qi, Daxin Jiang, Weizhu Chen, and Nan Duan. 2022. CodeRetriever: A Large Scale Contrastive Pre-Training Method for Code Search. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (Eds.). Association for Computational Linguistics, 2898–2910. doi:10.18653/V1/2022.EMNLP-MAIN.187

[20] Derek Lin, Rashmi Raghu, Vivek Ramamurthy, Jin Yu, Regunathan Radhakrishnan, and Joseph Fernandez. 2014. Unveiling clusters of events for alert and incident management in large-scale enterprise it. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, Sofus A. Macskassy, Claudia Perlich, Jure Leskovec, Wei Wang, and Rayid Ghani (Eds.). ACM, 1630–1639. doi:10.1145/2623330.2623360

[21] Qingwei Lin, Hongyu Zhang, Jian-Guang Lou, Yu Zhang, and Xuewei Chen. 2016. Log clustering based problem identification for online service systems. In *Proceedings of the 38th International Conference on Software Engineering, ICSE 2016, Austin, TX, USA, May 14-22, 2016 - Companion Volume*, Laura K. Dillon, Willem Visser, and Laurie A. Williams (Eds.). ACM, 102–111. doi:10.1145/2889160.2889232

[22] Fengrui Liu, Xiao He, Tieying Zhang, Jianjun Chen, Yi Li, Lihua Yi, Haipeng Zhang, Gang Wu, and Rui Shi. 2025. TickIt: Leveraging Large Language Models for Automated Ticket Escalation. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering, FSE Companion 2025, Clarion Hotel Trondheim, Trondheim, Norway, June 23-28, 2025*, Leonardo Montecchi, Jingyue Li, Denys Poshyvanyk, and Dongmei Zhang (Eds.). ACM, 343–354. doi:10.1145/3696630.3728558

[23] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the Middle: How Language Models Use Long Contexts. *Trans. Assoc. Comput. Linguistics* 12 (2024), 157–173. doi:10.1162/TACL_A_00638

[24] Yilun Liu, Shimin Tao, Weibin Meng, Feiyu Yao, Xiaofeng Zhao, and Hao Yang. 2024. LogPrompt: Prompt Engineering Towards Zero-Shot and Interpretable Log Analysis. In *Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings, ICSE Companion 2024, Lisbon, Portugal, April 14-20, 2024*. ACM, 364–365. doi:10.1145/3639478.3643108

[25] Weibin Meng, Ying Liu, Yichen Zhu, Shenglin Zhang, Dan Pei, Yuqing Liu, Yihao Chen, Ruizhi Zhang, Shimin Tao, Pei Sun, and Rong Zhou. 2019. LogAnomaly: Unsupervised Detection of Sequential and Quantitative Anomalies in Unstructured Logs. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, Sarit Kraus (Ed.). ijcai.org, 4739–4745. doi:10.24963/IJCAI.2019/658

[26] GLM-4.5-Air model pricing. 2025. https://bigmodel.cn/pricing

[27] Cayque Monteiro Castro Nascimento and André Silva Pimentel. 2023. Do Large Language Models Understand Chemistry? A Conversation with ChatGPT. *J. Chem. Inf. Model.* 63, 6 (2023), 1649–1655. doi:10.1021/ACS.JCIM.3C00285

[28] Changhua Pei, Zexin Wang, Fengrui Liu, Zeyan Li, Yang Liu, Xiao He, Rong Kang, Tieying Zhang, Jianjun Chen, Jianhui Li, Gaogang Xie, and Dan Pei. 2025. Flow-of-Action: SOP Enhanced LLM-Based Multi-Agent System for Root Cause Analysis. In *Companion Proceedings of the ACM on Web Conference 2025, WWW 2025, Sydney, NSW, Australia, 28 April 2025 - 2 May 2025*, Guodong Long, Michale Blumestein, Yi Chang, Liane Lewin-Eytan, Zi Helen Huang, and Elad Yom-Tov

(Eds.). ACM, 422–431. doi:10.1145/3701716.3715225

[29] Phuong Pham, Vivek Jain, Lukas Dauterman, Justin Ormont, and Navendu Jain. 2020. DeepTriage: Automated Transfer Assistance for Incidents in Cloud Services. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash (Eds.). ACM, 3281–3289. doi:10.1145/3394486.3403380

[30] Ankur Sarker, Haiying Shen, Mizanur Rahman, Mashrur Chowdhury, Kakan C. Dey, Fangjian Li, Yue Wang, and Husnu S. Narman. 2020. A Review of Sensing and Communication, Human Factors, and Controller Aspects for Information-Aware Connected and Automated Vehicles. *IEEE Trans. Intell. Transp. Syst.* 21, 1 (2020), 7–29. doi:10.1109/TITS.2019.2892399

[31] Zhengyu Tan, Ningyi Dai, Yating Su, Ruifo Zhang, Yijun Li, Di Wu, and Shutao Li. 2022. Human-Machine Interaction in Intelligent and Connected Vehicles: A Review of Status Quo, Issues, and Opportunities. *IEEE Trans. Intell. Transp. Syst.* 23, 9 (2022), 13954–13975. doi:10.1109/TITS.2021.3127217

[32] Haopei Wang, Anubhavnidhi Abhashkumar, Changyu Lin, Tianrong Zhang, Xiaoming Gu, Ning Ma, Chang Wu, Songlin Liu, Wei Zhou, Yongbin Dong, Weirong Jiang, and Yi Wang. 2024. NetAssistant: Dialogue Based Network Diagnosis in Data Center Networks. In *21st USENIX Symposium on Networked Systems Design and Implementation, NSDI 2024, Santa Clara, CA, April 15-17, 2024*, Laurent Vanbever and Irene Zhang (Eds.). USENIX Association.

[33] Zexin Wang, Jianhui Li, Minghua Ma, Ze Li, Yu Kang, Chaoyun Zhang, Chetan Bansal, Murali Chintalapati, Saravan Rajmohan, Qingwei Lin, Dongmei Zhang, Changhua Pei, and Gaogang Xie. 2024. Large Language Models Can Provide Accurate and Interpretable Incident Triage. In *35th IEEE International Symposium on Software Reliability Engineering, ISSRE 2024, Tsukuba, Japan, October 28-31, 2024*. IEEE, 523–534. doi:10.1109/ISSRE62328.2024.00056

[34] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

[35] Zhaoyang Yu, Aoyang Fang, Minghua Ma, Jaskaran Singh Walia, Chaoyun Zhang, Shu Chi, Ze Li, Murali Chintalapati, Xuchao Zhang, Rujia Wang, Chetan Bansal, Saravan Rajmohan, Qingwei Lin, Shenglin Zhang, Dan Pei, and Pinjia He. 2025. Triangle: Empowering Incident Triage with Multi-Agent. In *2025 40th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 674–686. doi:10.1109/ASE63991.2025.00062

[36] Wei Yuan, Shan Lu, Hailong Sun, and Xudong Liu. 2020. How are distributed bugs diagnosed and fixed through system logs? *Inf. Softw. Technol.* 119 (2020). doi:10.1016/J.INFSOF.2019.106234

[37] Biao Zhang, Barry Haddow, and Alexandra Birch. 2023. Prompting Large Language Model for Machine Translation: A Case Study. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA (Proceedings of Machine Learning Research)*, Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (Eds.). PMLR, 41092–41110.

[38] Wei Zhang, Hongcheng Guo, Jian Yang, Zhoujin Tian, Yi Zhang, Chaoran Yan, Zhoujun Li, Tongliang Li, Xu Shi, Liangfan Zheng, and Bo Zhang. 2024. mABC: Multi-Agent Blockchain-inspired Collaboration for Root Cause Analysis in Micro-Services Architecture. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, 4017–4033.

[39] Xuchao Zhang, Supriyo Ghosh, Chetan Bansal, Rujia Wang, Minghua Ma, Yu Kang, and Saravan Rajmohan. 2024. Automated Root Causing of Cloud Incidents using In-Context Learning with GPT-4. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering, FSE 2024, Porto de Galinhas, Brazil, July 15-19, 2024*, Marcelo d'Amorim (Ed.). ACM, 266–277. doi:10.1145/3663529.3663846

[40] Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 Embedding: Advancing Text Embedding and Reranking Through Foundation Models. *CoRR* abs/2506.05176 (2025). arXiv:2506.05176 doi:10.48550/ARXIV.2506.05176

[41] Nengwen Zhao, Junjie Chen, Xiao Peng, Honglin Wang, Xinya Wu, Yuanzong Zhang, Zikai Chen, Xiangzhong Zheng, Xiaohui Nie, Gang Wang, Yong Wu, Fang Zhou, Wenchi Zhang, Kaixin Sui, and Dan Pei. 2020. Understanding and handling alert storm for online service systems. In *ICSE-SEIP 2020: 42nd International Conference on Software Engineering, Software Engineering in Practice, Seoul, South Korea, 27 June - 19 July, 2020*, Gregg Rothermel and Doo-Hwan Bae (Eds.). ACM, 162–171. doi:10.1145/3377813.3381363

[42] Xuanhe Zhou, Guoliang Li, Zhaoyan Sun, Zhiyuan Liu, Weize Chen, Jianming Wu, Jiesi Liu, Ruohang Feng, and Guoyang Zeng. 2024. D-Bot: Database Diagnosis System using Large Language Models. *Proc. VLDB Endow.* 17, 10 (2024), 2514–2527. doi:10.14778/3675034.3675043

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009