International Journal of Computer Trends and Technology
ISSN: 2231–2803 / https://doi.org/10.14445/22312803/IJCTT-V73I3P116

Volume 73 Issue 3, 125-132, March 2025
© 2025 Seventh Sense Research Group®

Original Article

Reducing Incident Mean Time to Resolution Using Elasticsearch and Large Language Models

Govind Singh Rawat

Site Reliability Engineer, TikTok U.S. Data Security Inc., California, USA.

[1]*Corresponding Author : govindrawat54@gmail.com*

Received: 23 January 2025 Revised: 28 February 2025 Accepted: 20 March 2025 Published: 30 March 2025

Abstract *- Enhancing incident resolution is a key focus in modern Site Reliability Engineering (SRE). This paper presents a system that combines ElasticSearch (ES) with Large Language Models (LLMs) to reduce Mean Time to Resolution (MTTR). By embedding historical alarm data, extracting essential features, and leveraging k-nearest neighbors (kNN) search, the system efficiently links past incidents, retrieves relevant resolutions, and improves operational response through LLM interaction. This continuous feedback loop enhances incident response speed and facilitates faster incident resolution.*

Keywords *- ElasticSearch (ES), Incident mitigation, K-nearest neighbors search, Mean Time to Resolution, Site Reliability Engineering.*

1. Introduction

In large-scale distributed systems, swift and accurate incident response is critical to minimizing downtime and ensuring system reliability. Traditional monitoring solutions generate high-severity alarms to notify engineers of potential failures; however, diagnosing and resolving these alarms remains a significant challenge. Current incident response systems rely heavily on manual investigation, leading to prolonged resolution times and increased operational costs.

Existing solutions, including widely used observability tools like Datadog, Splunk, and Prometheus, offer real-time monitoring and alerting but lack key capabilities essential for efficient troubleshooting. These tools primarily depend on predefined queries, dashboard-based investigations, or manual correlation of logs, all of which can be time-consuming. Moreover, they do not effectively leverage historical resolutions or contextual insights to aid engineers in incident response.

A major research gap lies in integrating intelligent search mechanisms, contextual retrieval, and automated recommendations for incident resolution. While previous studies have explored log-based anomaly detection and knowledge graph-enhanced troubleshooting, they have not sufficiently addressed the need for a unified system that combines historical insights, efficient search capabilities, and AI-driven contextual guidance.

This paper proposes an intelligent incident resolution system that integrates Elasticsearch [1] for high-speed log retrieval, Large Language Models (LLMs) [2] for contextual troubleshooting insights, and k-nearest neighbor (kNN) search [3] for historical incident pattern recognition. By embedding these technologies within Site Reliability Engineering (SRE) [4] on-call [5] workflows, the system aims to enhance incident response efficiency, reduce MTTR [6], and provide engineers with immediate access to relevant historical resolutions. This approach bridges the gap in incident management solutions by offering a proactive, context-aware resolution framework tailored to large-scale distributed environments.

2. Problem Statement

Traditional incident response systems in large-scale distributed environments [7] often rely on basic monitoring tools and reactive maintenance, leading to prolonged downtime and degraded system reliability. Key challenges in incident management for Site Reliability Engineers include:

- Lack of efficient real-time diagnostic capabilities.
- Insufficient contextual information to quickly resolve alarms.
- High Mean Time to Resolution (MTTR) due to manual intervention and inefficient search.
- Limited automation in identifying and responding to related incidents
- Difficulty in learning from historical incidents to predict and prevent recurring issues.

A modern incident resolution system should address these challenges by leveraging efficient search capabilities, proactive failure detection, and automated remediation processes. Integrating technologies like ElasticSearch (ES) for fast search, k-nearest neighbor (kNN) search for historical context matches, and LLM for contextual insights can significantly enhance the speed of incident resolution, reducing MTTR, promoting system reliability and efficiency during the issue resolution process.

3. Literature Review

Efficient alarm and incident management in distributed systems is critical to maintaining system reliability and minimizing downtime. As modern infrastructures grow increasingly complex, organizations rely on automated monitoring, anomaly detection, and intelligent troubleshooting tools to manage system failures. However, existing approaches still present gaps in providing seamless, context-aware incident resolution. This review explores various advancements in anomaly detection, knowledge graph-enhanced troubleshooting, log retrieval, and the integration of Large Language Models (LLMs) to improve incident management workflows.

Organizations have relied on rule-based alerting systems and manual log analysis to detect and address incidents. These traditional approaches suffer from high false positive rates and lack scalability when dealing with distributed cloud environments, which generate overwhelming amounts. Previous research on log-based anomaly detection [8] has highlighted the advantages of applying machine learning models to predict failures. Traditional ML methods, such as decision trees and support vector machines (SVMs), have demonstrated moderate success. However, research on k-nearest neighbor (kNN) search in anomaly detection highlights its ability to recognize recurring failure patterns, reducing engineers' time diagnosing similar issues.

Structured data representation through knowledge graphs has proven effective in improving diagnostic efficiency [9]. Unlike traditional lookup-based systems, knowledge graphs encode relationships between system components, past incidents, and resolution steps to enable context-aware recommendations. By integrating domain-specific knowledge, these graphs can enhance incident resolution strategies beyond purely ML-driven approaches. This paper talks about how existing resolution knowledge can be combined with kNN search to provide faster correlations to solve the issues promptly.

Incident resolution workflows require efficient log indexing and retrieval mechanisms to extract relevant historical data. The use of ElasticSearch for log indexing and retrieval [10] has emerged as a scalable solution for querying large datasets. While ElasticSearch allows for keyword-based searches, its reliance on manual query construction can slow down the incident resolution process. Recent approaches propose embedding-based retrieval techniques, such as vector search and semantic similarity models, to enhance search relevance. This paper builds on these advancements by integrating kNN-based retrieval into ElasticSearch, reducing the need for manual searches and enabling engineers to retrieve past solutions based on semantic similarity rather than keyword matches.

Recent developments in Retrieval-Augmented Generation (RAG) demonstrate how combining Large Language Models [11] with document retrieval can offer context-sensitive troubleshooting assistance. However, challenges such as hallucination risks, data freshness, and model interpretability need to be addressed to fully integrate LLMs into critical incident response workflows. This paper proposes a hybrid approach that leverages RAG, ElasticSearch, and kNN retrieval of historical resolutions, ensuring highly relevant, real-time troubleshooting assistance while minimizing AI-related risks.

Based on some of the prevalent observability, alarming, and monitoring tools like Datadog [13], Splunk [14], and Prometheus [15], they still lack several essential capabilities:

Datadog offers real-time monitoring and alerting with integrations across cloud platforms, allowing engineers to detect anomalies proactively. While it supports event correlation, it lacks embedding-based retrieval to provide engineers with past solutions for similar alarms, which is catered to in this paper.

Splunk is a popular tool for log management and analytics, offering real-time insights into system behavior. However, its reliance on manual search queries and dashboard-based investigations can delay incident resolution. This paper utilizes kNN-based retrieval in ElasticSearch, reducing the need for manual searches and enhancing response efficiency.

Prometheus is an open-source monitoring system designed for time-series data collection and alerting. While effective in monitoring system health, it lacks context-aware troubleshooting capabilities and does not store historical solutions for similar alarms, which is being addressed in this paper.

The proposed system addresses the above gaps by integrating Large Language Models (LLMs) to offer structured troubleshooting recommendations. Incident response teams typically rely on communication channels to track details and steps taken by engineering teams. Some teams prefer using live chat platforms like Slack [16] or Microsoft Teams [17]. However, tracing back previous chats regarding issue resolutions and the team members involved is still a manual process.

Despite the advancements in these tools, current solutions often fail to provide an integrated approach that combines search, embeddings, and automated retrieval-based recommendations for alarms or incident resolution based on historical resolutions and related chats. This paper proposes a unified system that builds upon these foundations by leveraging ES, kNN, and LLMs, specifically designed for Site Reliability Engineering (SRE) workflows to optimize MTTR reduction and give engineers a head start in issue resolution.

4. Proposed System Architecture

The proposed system architecture aims to optimize incident resolution within Site Reliability Engineering (SRE) using advanced data processing and retrieval methodologies. For the sake of understanding, only high-severity or critical [18] alarms are targeted, but implementers are free to choose multiple or all categories of alarms. The architecture has the following major components:

- Batch (Seeding [19]) System - One-time or on-demand batch system which can populate the ElasticSearch indices with existing alarms and their features, resolution chats or texts, and their embeddings corresponding to each of the past high severity alarms.
- Active System - the automation of identifying and guiding based on past resolutions for the current alarm in the system. This core part provides run-time interaction with existing ES indices, storing the historical occurrences, alarm features, and chat resolution text with associated embeddings.
- Bot/Agent [20] [21]- Not a core component for the system, but it needs to interact with the system once an alarm is queried or analyzed.
- LLM Processor - This can be on-prem [22] or a third-party implementation of a Large Language Model like GPT-4 [23], Mistral [24], Llama [25] that can respond to user queries from a set of context text.

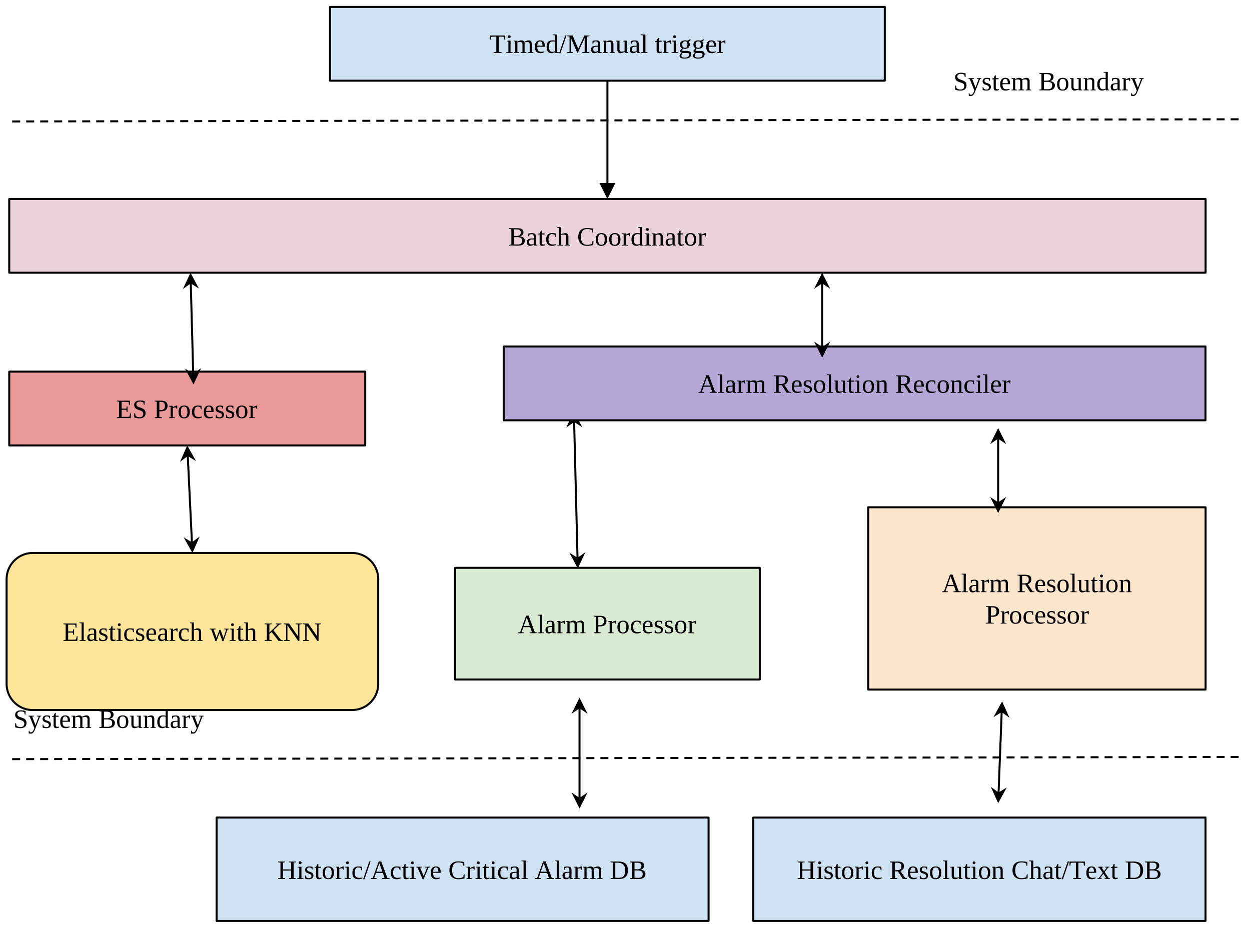
Timed/Manual trigger
System Boundary
Batch Coordinator
ES Processor
Alarm Resolution Reconciler
Elasticsearch with KNN
Alarm Processor
Alarm Resolution Processor
System Boundary
Historic/Active Critical Alarm DB
Historic Resolution Chat/Text DB

Fig. 1 Batch (Seeding) System Components Within System Boundary

4.1. Batch (Seed) System

This system includes a batch pipeline capable of accessing existing data sources for alarms and corresponding resolution chats or text. It assumes that such a database exists for past alarms and resolution communications. For instance, high-severity alarm details and their attributes may be sourced from tools like Datadog, Splunk, or Prometheus.

Resolution chats containing event timelines, active participants, and actions taken, can be retrieved from platforms like Slack Group or Microsoft Teams. This implementation can function as a standalone solution dedicated to batch data processing or share components with the Active System described later. For simplicity, the batch system is treated as a separate entity, as depicted in Figure 1.

The main components of the batch system in Figure 1 are:

4.1.1. Batch Coordinator

The entry point to the batch system, triggered either manually or by a timed schedule. Its primary responsibilities include initiating batch processing, managing data flow between subcomponents, and reporting completion to the external system that triggered the process.

4.1.2. Alarm Resolution Processor

Retrieves historical alarm resolution chat data from the database and extracts key details such as participants, start and end times, fix summary, associated alarm, and its identifier. It then returns the collected and generated information to the Alarm Resolution Reconciler.

4.1.3. Alarm Processor

Upon receiving an alarm ID from the Alarm Resolution Reconciler, it looks up the corresponding alarm and extracts relevant properties, such as alarm start time, on-call engineer, acknowledgement time, resolution time, and alarm content.

4.1.4. Alarm Resolution Reconciler

Assigns a unique identifier to the alarm and its corresponding resolution text, ensuring they can be saved uniquely in the ElasticSearch (ES) index [26]. The generated document [27] includes embeddings created from both the chat text and alarm data and the original alarm ID, chat group ID, and other properties that uniquely link the alarm to the chat interaction. Once the Batch Coordinator receives the document, it is passed to the ES processor for storage in the specified index.

4.1.5. ES Processor

Connects to ElasticSearch and stores the documents in the relevant ES index. These documents also include the associated embeddings, which can later be retrieved using k-NN search.

4.2. Active System

Certain components of the Active System overlap with those of the Batch (Seed) System. For instance, the ES Processor and the Alarm Processor perform functions similar to those in the Batch System. For clarity, figure 2 includes them as part of the Active System. Implementers have the option to design these components for reuse or keep them separate, provided the same Elasticsearch instance is used by both the Batch and Active Systems' ES Processors for k-nn search. The Active System is designed around retrieving relevant incident information, with the Active Coordinator ensuring seamless interaction between system components. The LLM Processor is crucial in providing engineers with actionable insights from historical issue data. This data is processed by the backend LLM implementation, which then generates interactions. Both the LLM and Agent/Bot are positioned at the system boundaries by design, indicating that these components can either be developed in-house or use an existing external solution to meet system objectives.

The Active System components shown in Figure 2 are outlined below:

4.2.1. Agent/Bot

This component acts as the interface between the issue communication chat group and the Active System. Depending on the requirements, it can be an in-house solution that connects the chat application with the Active System and triggers the Active Coordinator with information about the active alarm and chat group. Although it is not the main focus of this discussion, the Agent/Bot could be extended to automate tasks beyond just serving as an interface to the Active System.

4.2.2. Alarm Processor

Upon receiving an alarm ID from the Active Coordinator, this component retrieves the associated alarm and extracts key details such as alarm start time, on-call engineer, acknowledgement time, resolution time, and alarm content.

4.2.3. ES Processor

This component connects to ElasticSearch and performs a k-NN search to retrieve related historical documents.

4.2.4. Active Coordinator

As the central component of the Active System, the Active Coordinator serves as the entry point. The Agent/Bot informs the coordinator when a new chat group is opened for a specific alarm ID and provides alarm-related details. The coordinator then calls the Alarm Processor to obtain alarm details and passes them to the ES Processor. The ES Processor returns the k nearest neighbors of the current alarm to the coordinator, forwarding the current alarm and historical information to the LLM Processor.

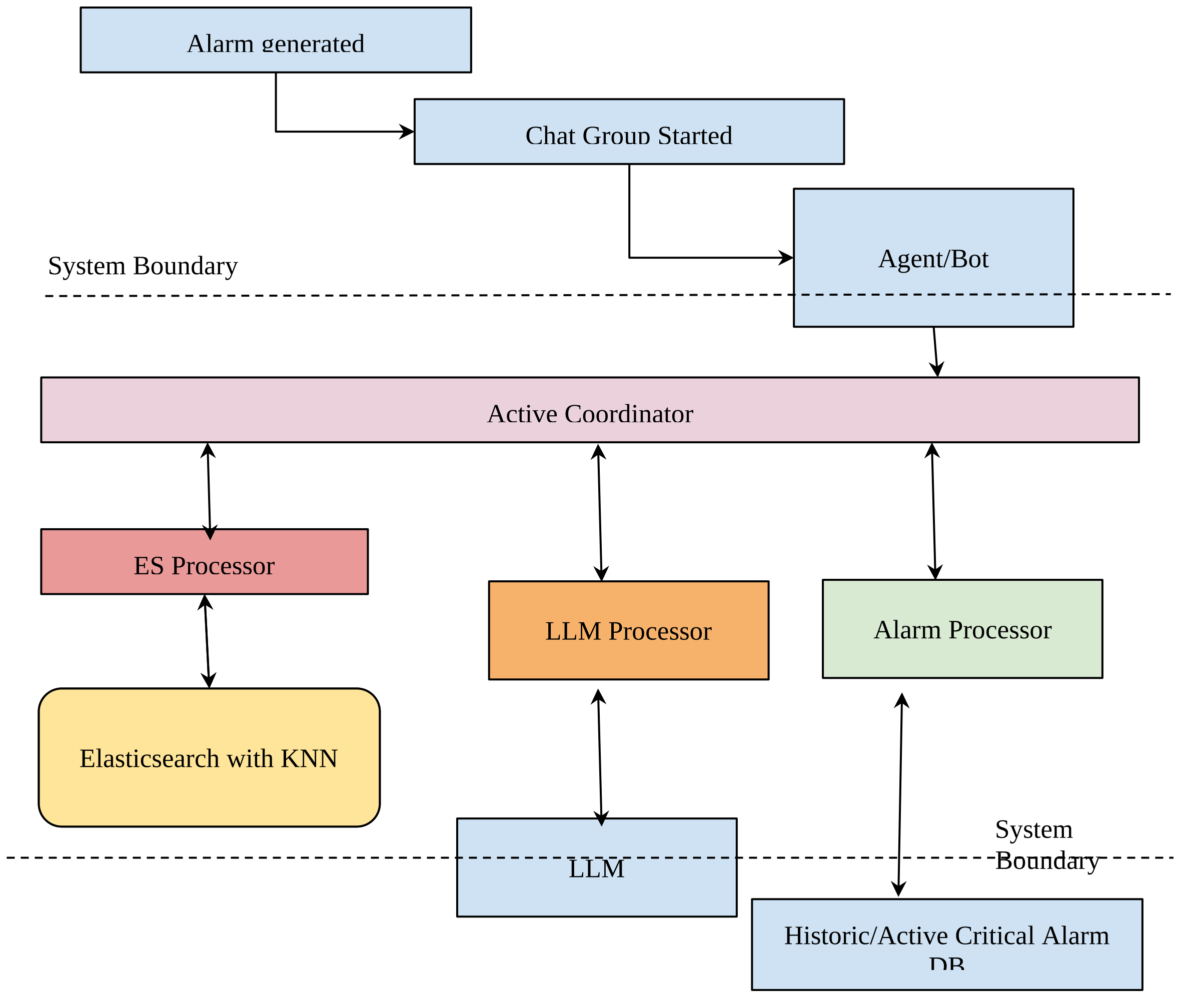

Fig. 2 Active system components with system boundary

4.2.5. LLM Processor

The LLM Processor receives the current alarm details and associated historical data from the coordinator. It triggers the LLM with a seed prompt [28] and receives an initial response, which it sends back to the coordinator. The coordinator then passes it to the agent/bot. This initial response contains predefined information the SRE team may want to address immediately. The LLM Processor waits for future interactions within the context of the ongoing conversation.

5. Experimental Setup

The setup uses Elasticsearch 8.17 with kNN search. It has an index containing 1,000 database-related issues, alarms, and their resolutions. GPT-4 serves as the backend of the LLM processor. For issue resolution and conversation processing, a Slack app acts as an agent/bot. A set of 50 critical alarms is randomly selected from database-related issues and assigned to two teams of entry-level engineers, all with the same level of work experience and familiarity with the database-related issue. An A/B test [29] is conducted, where one group of engineers resolves the alarms using the proposed system while the second group resolves them without the system. The following metrics are tracked for both teams, with and without the system's assistance:

- MTTR (Mean Time to Resolution)
- Incident escalation rate
- Number of additional teams engaged to resolve the issue
- Accuracy of retrieved relevant past incidents
- Engineer satisfaction score

6. Performance Evaluation

The following were the results of measuring the proposed A/B test metrics, where manual resolution was taken as the baseline, and Setup Assisted Resolution was measured for the team.

Table 1. Mean time to resolution
((Average time taken to resolve an alarm from detection to closure))

	MTTR (minutes)	% Improvement
Manual	85 minutes	-
Setup Assisted	47 minutes	44.7% faster

Table 2. Incident escalation rate
((Percentage of alarms requiring escalation to senior engineers or external teams))

	% of Incidents Escalated	% Reduction
Manual	31%	Values
Setup Assisted	18%	42% fewer escalations

Table 3. Number of additional teams involved
((Average number of teams consulted during incident resolution))

	Avg. Teams Involved Per Incident	% Reduction
Manual	2.8 teams	-
Setup Assisted	1.5 teams	46.4% fewer teams

Table 4. Retrieval accuracy of relevant past incidents
((Percentage of retrieved past alarm and alarm resolutions that were relevant to current alarm))

Top-K retrieved incidents (k=3)	% Relevance Score
Manual	64%
Setup Assisted	88%

Table 5. Engineer satisfaction score
((Engineer feedback on setup's usefulness for troubleshooting (Sacle 1-5)

	Avg. Satisfaction Score
Manual	3.2 / 5
Setup Assisted	4.3/5

6.1. Observations on Performance Evaluations

6.1.1. MTTR improvements

Setup-assisted troubleshooting reduced MTTR by ~44.7% due to rapid retrieval of past resolutions and guided suggestions. Engineers spent less time manually searching for solutions in logs and documentation.

6.1.2. Incident Escalation Rate

42% reduction in escalations, indicating that engineers resolved more issues independently with the setup's assistance. The setup provided better root-cause analysis, reducing the need for external expertise.

6.1.3 Number of Additional Teams Involved

Engineers using the setup needed fewer team interactions to resolve alarms, leading to less cross-team dependency. The system provided actionable insights that reduced back-and-forth communication with other teams.

6.1.4 Retrieval Accuracy of Relevant Past Incidents

The setup retrieved relevant past incidents with 88% accuracy, significantly outperforming manual searches. Higher relevance means engineers spent less time filtering through unrelated past cases.

6.1.5. Engineer Satisfaction Score

Engineers found the system highly effective in reducing time spent on resolution tasks, specifically searching relevant logs. Feedback highlighted that setup explanations improved troubleshooting confidence.

6.2. Summary of Performance Evaluation

Metric	Baseline (manual)	Setup Assisted	Improvement
MTTR	85 min	47 min	44.7% faster
Escalation Rate	31%	18%	42% fewer escalations
Avg. teams involved	2.8 teams	1.5 teams	46.4% fewer teams
Past Incident Retrieval Accuracy	64%	88%	37.5% Higher Accuracy
Engineer Satisfaction Score	64%	86%	25% Improved Satisfaction

6.3 Unique Advantages of Proposed Approach

While existing techniques rely on predefined queries, manual log analysis, and reactive troubleshooting, the proposed system integrates Elasticsearch, Large Language Models (LLMs), and k-nearest neighbor (kNN) search to provide intelligent, automated, and context-aware incident resolution. This section discusses how and why the proposed approach yields better results than conventional and state-of-the-art techniques. Traditional incident response systems depend on manual log correlation and predefined query-based investigations, often leading to prolonged resolution times. The proposed system significantly reduces MTTR by utilizing Elasticsearch to rapidly retrieve relevant past resolutions. Leveraging kNN search to find similar historical incidents aids engineers in faster diagnosis and integrates LLMs to interpret alarm contexts and suggest the most relevant resolution, which fast-tracks problem identification.

6.3.1. Experimental Validation

A/B testing showed that engineers using the proposed system resolved alarms 44.7% faster than those relying on conventional methods, with a measurable decrease in

MTTR. Existing tools such as Datadog, Splunk, and Prometheus primarily focus on real-time monitoring and alerting but lack intelligent troubleshooting capabilities, specifically in the following areas:

- Contextual Understanding**:** LLMs analyze alarm descriptions to extract key features, reducing misdiagnoses.
- Automated Historical Referencing**:** Unlike traditional systems, which do not efficiently retain past resolution knowledge, the proposed system continually learns from historical data to refine its recommendations.

5. Conclusion

The LLM-powered alarm resolution system significantly enhances incident management efficiency within Site Reliability Engineering (SRE) by integrating historical alarm data, retrieval-augmented search, and generative AI-driven recommendations. The system demonstrated notable improvements across key operational metrics through a structured evaluation, including a 44.7% reduction in Mean Time to Resolution (MTTR), a 42% drop in incident escalations, and a 46.4% decrease in cross-team dependencies. These gains highlight the system's ability to streamline troubleshooting workflows, ensuring that engineers can resolve high-severity alarms with greater speed and accuracy.

By leveraging ElasticSearch for k-NN retrieval, an Active System for real-time interaction, and an LLM Processor for intelligent guidance, the architecture minimizes manual effort while maximizing contextual awareness. The Batch (Seed) System further strengthens the solution by pre-populating historical alarm-resolution data, ensuring the retrieval process remains robust and relevant. Additionally, incorporating a Bot/Agent interface facilitates seamless integration with existing communication platforms like Slack and Microsoft Teams, improving accessibility for engineers in real time.

Overall, the system reduces operational overhead, enhances knowledge retention, and enables a data-driven approach to incident resolution. Future enhancements may include expanding the dataset beyond high-severity alarms, improving multi-turn interactions with the LLM, and integrating real-time anomaly detection mechanisms to refine response strategies further. The observed improvements in resolution efficiency, reduced escalation rates, and engineer satisfaction affirm the viability of this approach in modern SRE workflows.

References

[1] What is Elasticsearch?, Elastic. [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/current/elasticsearch-intro-what-is-es.html/

[2] Wayne Xin Zhao et al., "A Survey of Large Language Models," *arXiv*, pp. 1-144, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[3] k-Nearest Neighbor (kNN) Search, Elastic. [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/8.0/knn-search.html

[4] Betsy Beyer et al., *Site Reliability Engineering: How Google Runs Production Systems*, O'Reilly Media, pp. 1-552, 2016. [Google Scholar] [Publisher Link]

[5] Ollie Cook et al., *On-Call*, SRE Workbook Chapter 8, 2016. [Publisher Link]

[6] What is Meant Time to Resolution?, A Guide to Incident Metrics, Instatus. [Online]. Available: https://instatus.com/blog/mttr/

[7] Mohan Sitaram, Mastering the Art of Troubleshooting Large-Scale Distributed Systems, DevOps.Com, 2024. [Online]. Available: https://devops.com/mastering-the-art-of-troubleshooting-large-scale-distributed-systems

[8] Max Landauer et al., "Deep Learning for Anomaly Detection in Log Data: A Survey," *Machine Learning with Applications*, vol. 12, pp. 1-19, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[9] Syed Abdul, LogBERT: Log File Anomaly Detection Using BERT, 2022. [Online]. Available: https://medium.com/infinstor/logbert-log-file-anomaly-detection-using-bert-an-explainer-db20bfd2f91f/

[10] Salam Allawi Hussein, and Sándor R. Répás, "Anomaly Detection in Log Files Based on Machine Learning Techniques," *Journal of Electrical Systems*, vol. 20, no. 3s, pp. 1299-1311, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[11] Uday Kamath et al., *Large Language Models: A Deep* Dive, Bridging Theory and Practice, Springer, pp. 1-472, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[12] Guilherme O. Campos et al., "On the Evaluation of Unsupervised Outlier Detection: Measures, Datasets, and an Empirical Study," *Data Mining and Knowledge Discovery*, vol. 30, no. 4, pp. 891-927, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[13] Real-Time Cloud Monitoring, Datadog. [Online]. Available: https://www.datadoghq.com/

[14] Enterprise Logging and Alerting, Splunk. [Online]. Available: https://www.splunk.com/

[15] Open-Source Metrics and Monitoring, Prometheus. [Online]. Available: https://prometheus.io/

[16] Slack Software. [Online]. Available: https://slack.com/

[17] Microsoft Teams Software. [Online]. Available: https://www.microsoft.com/en-us/microsoft-teams/group-chat-software/

[18] Bill Hollifield, and Eddie Habib, *The Alarm Management Handbook: A Comprehensive Guide*, PAS, pp. 1-261, 2010. [Google Scholar] [Publisher Link]
[19] Seeding, Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Seeding_(computing)
[20] What are Bots?, Slack Api. [Online]. Available: https://api.slack.com/legacy/enabling-bot-users/
[21] Bot Overview, MicroSoft Team. [Online]. Available: https://learn.microsoft.com/en-us/microsoftteams/platform/bots/overview?utm_source=chatgpt.com
[22] Thomas Erl, Ricardo Puttini, and Zaigham Mahmood, *Cloud Computing: Concepts, Technology & Architecture*, pp. 1-528, 2013. [Google Scholar] [Publisher Link]
[23] OpenAI, GPT-4 Technical Report, 2023. [Online]. Available: https://openai.com/research/gpt-4
[24] Mistral AI, Mistral 7B: A Dense, Efficient, Open-Source Language Model, 2023. [Online]. Available: https://mistral.ai/
[25] Hugo Touvron et al., "LLaMA: Open and Efficient Foundation Language Models," *arXiv*, pp. 1-27, 2023. [Google Scholar] [Publisher Link] [CrossRef]
[26] David Brimley, What is an Elasticsearch Index?, Elastic, 2023. [Online]. Available: https://www.elastic.co/blog/what-is-an-elasticsearch-index
[27] Index, Documents and Fields, Elastic. [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/current/documents-indices.html/
[28] XiPeng Qiu et al., "Pre-Trained Models for Natural Language Processing: A Survey," *Science China Technological Sciences*, vol. 63, no. 1872-1897, 2020. [CrossRef] [Google Scholar] [Publisher Link]
[29] Ron Kohavi, Diane Tang, and Ya Xu, *Trustworthy Online Controlled Experiments: A Practical Guide to A/B Testing*, Cambridge University Press, 2020. [Google Scholar] [Publisher Link]