ADA: An AI-Powered Assistant for Managed Service Providers

Eleonora Breci[1], Sediola Ruko[1], Luigi V. Mancini[1] and Dorjan Hitaj[1,*]

[1]Department of Computer Science, Sapienza University of Rome

Abstract

In today's digital landscape, organizations increasingly rely on cloud-based infrastructures to manage their computational, storage, and networking needs. These environments provide scalability, flexibility, and cost-efficiency, but also introduce growing complexity in maintaining robust cybersecurity practices. As enterprises scale, Managed Service Providers are tasked with monitoring vast volumes of alerts, incidents, and system states, often under tight time constraints and with limited human resources. Traditional incident management models, while serviceable, frequently struggle to keep pace with the volume and urgency of modern security demands. As such, there is need for intelligent, automated solutions that can support human operators in prioritizing and resolving security alarms more effectively. Large Language Models and generative AI techniques have recently shown great promise in addressing such challenges by enabling context-aware reasoning, summarization, and decision support. This paper presents ADA, an AI-powered assistant for supporting security operations through intelligent alarm triage and incident management integrating generative AI capabilities with a manual Retrieval-Augmented Generation mechanism to enhance the contextual accuracy and relevance of responses. ADA can be deployed on a modular cloud-native architecture that includes serverless components to enable integration within enterprise communication environments. Evaluation on real-world alarm data showed a response accuracy of 91.35%, significantly reducing operator workload and resolution times.

Keywords

AI-assistants, Incident management, Managed Service Provider

1. Introduction

The rapid adoption of cloud computing has fundamentally transformed how organizations manage computational, storage, and networking resources. Cloud-based infrastructures offer significant advantages, including scalability, operational flexibility, and cost efficiency. However, these benefits are accompanied by increasing complexity in maintaining comprehensive cybersecurity postures. Modern enterprises generate vast volumes of security alerts, events, and operational telemetry, creating challenges for human operators tasked with monitoring, analyzing, and responding to potential threats. Managed Service Providers (MSPs) [1] play a critical role in supporting enterprises by delivering continuous security monitoring, incident detection, and response services. Despite advancements in security information and event management (SIEM) [2, 3, 4, 5, 6, 7] systems and automated rule-based detection, traditional incident management processes are often insufficient to keep pace with the scale, velocity, and sophistication of modern security operations. Analysts frequently face alert fatigue, delayed incident response, and difficulty in prioritizing alarms effectively, all of which can compromise the overall security posture of the organization. Recent advances in artificial intelligence, particularly Large Language Models (LLMs) and generative AI techniques, have opened new opportunities to augment human analysts also in security operations. In the past years, generative AI and LLMs have found broad applicability across a wide range of domains, driven by their ability to generate, transform, and reason over unstructured data [8]. In natural language processing, they are used for tasks such as text generation[9], summarization [10], translation [11, 12], and question answering [13], enabling more natural and efficient human–computer interaction. In software engineering, LLMs support code generation, debugging,

Joint National Conference on Cybersecurity (ITASEC & SERICS 2026), February 09-13, 2026, Cagliari, IT

*Corresponding author.

breci.2058813@studenti.uniroma1.it (E. Breci); ruko@di.uniroma1.it (S. Ruko); mancini@di.uniroma1.it (L. V. Mancini); hitaj.d@di.uniroma1.it (D. Hitaj)

documentation, and test case creation, improving developer productivity and code quality [14]. In cybersecurity and IT operations, generative AI systems assist with threat detection and analysis [15], incident response [16], security automation [17], forensics [18], penetration testing [19], phishing detection [20], chatbots [21] and beyond. Furthermore, beyond technical domains, generative models are increasingly applied in healthcare for clinical documentation, decision support, and medical imaging analysis [22], in finance for risk assessment, fraud detection, and automated customer support [23]. Creative industries also benefit from generative AI through applications in image, audio, and video synthesis, as well as design and storytelling [24]. Collectively, these applications highlight the versatility of generative AI and LLMs as general-purpose technologies capable of augmenting human expertise, streamlining complex workflows, and enabling new forms of intelligent automation across diverse sectors. Regarding incident response, relying on generative AI features such as context-aware reasoning, natural language understanding, and automated summarization, can be suited for AI-powered assistants that can support alarm triage, incident classification, and decision-making processes, enabling faster and more accurate responses providing not only operational efficiency but also improved situational awareness and reduced human error.

In this work, we conduct a comprehensive evaluation of alternative design choices for AI-assistants aimed at enhancing security operations, with a particular focus on intelligent alarm triage and incident management. Based on such evaluation and the insights gained we present the design and implementation of ADA, an AI-powered assistant that supports security analysts in the decision-making processes. ADA combines generative AI capabilities with a manually engineered Retrieval-Augmented Generation mechanism to enrich model responses with domain-specific knowledge, thereby improving contextual accuracy and relevance. To demonstrate the feasibility of the proposed approach, we implemented a proof-of-concept assistant within a modular, cloud-native architecture that leverages serverless components. This design enables scalable deployment and facilitates seamless integration with existing enterprise communication platforms and IT service management environments.

Summarizing, in this paper we make the following contributions:

- We present a through evaluation of various AI-based solutions for enhancing incident response workflow, discussing over the trade-offs between multiple key factors that affect the general operational workflow.
- We design and implement ADA, an AI assistant that integrates generative AI with retrieval-based mechanisms for security alarm triage. We provide information on how to implement such solution on an already existing cloud-native architecture for deploying AI-powered assistants within enterprise and MSP environments.
- We evaluate ADA on operational alarm datasets, highlighting improvements in response accuracy, workload reduction, and time-to-resolution.

This paper is organized as follows: Section 2 provides background information necessary to understand the upcoming sections. Section 3 introduces ADA our AI-powered assistant alongside the key driving factors that lead us to the final solution architecture and its evaluation. Section 4 discusses the related works in the domain and section 5 concludes the paper.

2. Background

2.1. Managed Service Provider

A managed service provider (MSP) is a company that specializes in providing and managing IT services for organizations of various sizes. These providers offer advanced solutions that include specialized servers, networks, and applications, ensuring security and business continuity. Applications and infrastructure provided by MSPs are typically hosted in cloud environments, where they are managed and monitored. MSPs often operate as providers of web hosting or enterprise application services, allowing enterprises to outsource the management of their IT infrastructure through tailored delivery contracts. This model enables companies to focus on their strategic activities, delegating responsibility

for technical administration, system upgrades, and IT security to MSPs. This approach ensures that companies reduce operational costs while maintaining flexibility and scalability of resources.

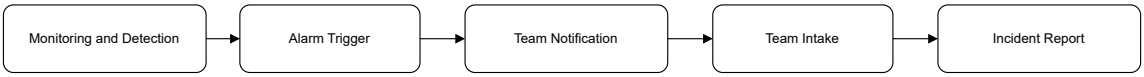

Figure 1: The incident response management workflow.

Nowadays, by considering the large amount of data processed, rapid incident detection and effective resolution are critical to maintain business continuity and minimize service disruptions. A well-structured incident management process enables organizations to respond quickly to unexpected issues, but also supports continuous improvement because it can identify root causes and preventing future events. Typically, the incident management process consists of five main steps, namely monitoring and detection, alarm trigger, incident response team notification, incident response team intake (and resolution) and lastly, the incident report as shown in Figure 1. The monitoring and detection steps relies on the use of tools that can monitor any anomalies in the monitored resources and then if a problem is detected or certain watched parameters surpass a pre-determined threshold, an alert is generated. This alert automatically triggers an incident notification to the MSP incident management team and subsequently the team follows the resolution procedure for that alert type and upon resolution a report containing the timeline of the incident, causes, and key insights for future improvements to prevent such incident form happening is produced by the team. With ADA, our aim is to position our AI-assistant in the Team Intake step, in order to enhance the operations team situational awareness and provide fast and swift access to operational guidelines and suggestions on steps to undertake to resolve the incident following the company's standard operational procedures.

2.2. Deep Learning

Deep Learning (DL) relies heavily on the use of neural networks, which are ML algorithms inspired by the human brain and are designed to resemble the interactions among neurons [25]. While standard ML algorithms require the presence of handcrafted features to operate, NNs determine relevant features on their own, learning them directly from the input data during the training process [26]. Two main requirements underline the success of NNs in general: 1) substantial quantities of rich training data, and 2) powerful computational resources. Large amounts of diverse training data enable NNs to learn features suitable for the task at hand, while simultaneously preventing them from memorizing the training data. Such features are better learned when NNs have multiple layers, thus the deep neural networks. Research has shown that the single-layer, shallow counterparts are not good at learning meaningful features and are often outperformed by other ML algorithms [26]. DNN training translates to vast numbers of computations requiring powerful resources, with graphical processing units (GPUs) a prime example. Deep Learning is the key factor for an increased interest in research and development in the area of Artificial Intelligence (AI), resulting in a surge of ML based applications that are reshaping entire fields and seedling new ones. Variations of DNNs have successfully been implemented in a plethora of domains, including here, but not limited to, image classification [27, 28, 29], natural language processing [30, 31, 32], speech recognition [33, 34], data (image, text, audio) generation [35, 36, 37, 38], cyber-security [39, 40, 41] and more.

2.2.1. Generative AI

Generative AI is a branch of DL where the focus is for the model to learn underlying data distributions and produce novel synthetic outputs. Early deep generative methods include Variational Autoencoders (VAEs) [42, 43], which formalized latent-variable modeling through variational inference and enabled tractable learning of continuous latent spaces. Generative Adversarial Networks (GANs) [44] advanced the field by introducing an adversarial training paradigm that produced highly realistic samples, particularly in imaging domains. In parallel, energy-based models [45, 46] provided an alternative probabilistic

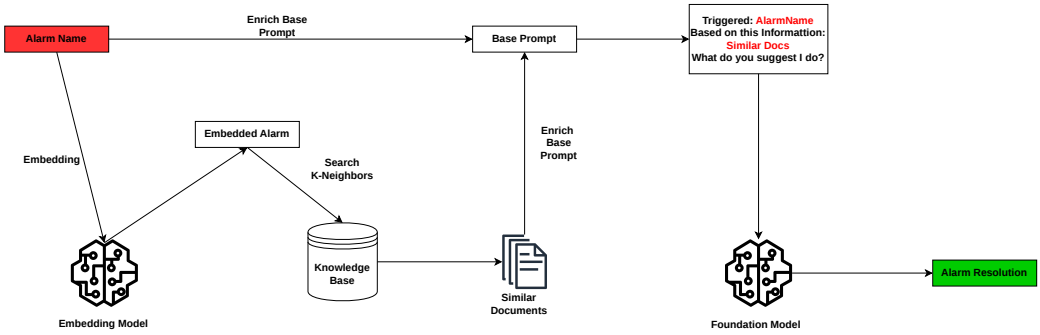

Figure 2: The Retrieval-Augmented Generation pattern, which combines retrieval of relevant external documents with generative models to produce more informed and contextually accurate responses.

framework in which data generation and inference were governed by learned energy functions defining relative likelihoods. Normalizing flows [47, 48] offered a complementary direction, enabling exact likelihood estimation and invertible transformations that map complex data distributions to simpler ones while retaining efficient sampling capabilities. In recent years, diffusion models [49, 50] achieved state-of-the-art performance in high-fidelity generative tasks by iteratively denoising samples drawn from learned diffusion processes. Building on these foundations, transformer-based architectures [51] expanded generative modeling into large-scale sequence domains, enabling context-aware generation through self-attention mechanisms and giving rise to Large Language Models (LLMs) [52, 53, 54, 55, 56] capable of advanced reasoning, summarization, and code synthesis. Through the years, these generative models have found their way into a diverse and rich range of applications such as text, image, audio recognition and synthesis, and even security applications improving significantly on their predecessors.

2.3. Retrieval Augmented Generation

Retrieval-Augmented Generation (RAG) [57] is a hybrid approach that enhances the performance of language models by integrating external knowledge retrieval into the text generation process. Instead of relying solely on the model's internal parameters, which are fixed after training, RAG first retrieves relevant information from a large corpus, such as documents, web data, or knowledge bases. This retrieved content is then provided as context to the generator (such as an LLM), which uses it to produce more accurate and contextually rich responses. This framework is particularly effective in open-domain question answering, and domains where up-to-date or domain-specific information is crucial. By decoupling knowledge storage from model inference, RAG systems mitigate issues related to model hallucination, enable rapid updates to organizational knowledge without retraining, and support fine-grained control over the scope of information accessible to the model. This approach has become foundational in enterprise applications, including security operations, incident response, and technical support [58, 59, 60], where reliable information retrieval must accompany natural language reasoning. Figure 2 presents the RAG pattern in practice in the context of incident response. Upon alarm triggering, the alarm identifiers such as name are embedded to then subsequently search a knowledge-base containing information about the specific alarm. This provides auxiliary, helpful information to enhance the base prompt, upon querying the subsequent LLM model, subsequently allowing the model to provide more accurate and contextualized alarm resolution feedback.

3. ADA

The goal of ADA is to enhance the efficiency and accuracy of Managed Service Providers in cloud-based environments. To do so, considerations need to be taken into account when it comes to the level of complexity regarding the initial setup and maintenance of the system, as are needed also for the

performance in the intended task. To gather such insights we decided to evaluate the Generative AI stack of Amazon, aiming at providing general-enough insights that are applicable to other vendors in the AI ecosystem. In what follows we provide a description the functional and non-functional requirements that guided its development, followed by a discussion of candidate components, the evaluation methodology used to select them, the final architecture, and considerations for deployment and integration.

3.1. Requirements

The development of ADA was guided by a set of functional and non-functional requirements derived from the operational needs of MSP security teams. These requirements reflect the necessity to handle large volumes of security alarms efficiently while maintaining accuracy, reliability, and responsiveness. In addition to supporting core capabilities such as alarm triage and contextual assistance, the system was required to integrate seamlessly with cloud-native infrastructures and existing operational tools. This subsection summarizes the key requirements that informed subsequent design and technology selection decisions.

- **Accuracy:** Percentage (%) of correct answers over the total attempts. A correct answer is defined as when the input alarm matches the procedure provided in the output. We set the accuracy threshold at 75% to consider a solution viable or not.
- **Quality:** Evaluation of the correctness of details in an answer relative to the total details across all answers. Quality values below the threshold of 3 out of 5 points make the solution unsuitable.
- **Quantity:** Assessment of the number of details in an answer relative to the total number of details possible in that answer. Quantity values below the threshold of 3 out of 5 points make the solution unsuitable in our setting.
- **Response Time:** The amount of time, measured in milliseconds (ms), required for the system to provide an output.
- **Model Response Time:** The time taken for interactions with AI models, measured in milliseconds (ms).
- **Initial Setup Effort:** The effort required to build the solution from scratch.
- **Maintenance Effort:** The effort needed to keep the solution updated or to recover it in case of disruption.
- **Flexibility:** The ability to implement changes to the infrastructure or to tune the model once the system is live.
- **Availability:** An indicator of the probability that a system will perform its intended functions correctly and continuously without interruption.
- **Maximum Throughput:** Measures the system's ability to withstand stress.
- **Time to Train:** The measured time required to train the model.
- **Cost:** Overall costs associated with the solution and its scalability.

3.2. Candidate Solutions

To meet the identified requirements, several candidate solutions were examined for the desired end goal, that of seamlessly building an AI assistant for enhancing the operational efficiency of MSP teams. These included alternative generative AI models, retrieval mechanisms for contextual grounding, and deployment strategies based on cloud-native services. Rather than assuming a single predefined approach, the design process explored multiple options in order to understand their respective trade-offs in terms of performance, complexity, cost, and operational suitability. The candidate solutions were chosen from the same Generative AI stack provider, for simplicity in evaluation and able to provide transferrable insights on other providers. The chosen stack was the Generative AI stack from Amazon which comprises applications, services and infrastructure at different granularity and complexity levels. The stack is divided into three main layers namely the **a)** productivity boosting applications (known as

Table 1
Performance comparison of different approaches for Incident response

	SageMaker + RAG	SageMaker + fine-tuning	Bedrock + manual RAG	Bedrock + fine-tuning	Bedrock + knowledge base	Amazon Q (Business)
Accuracy	79.9%	6.15%	91.35%	0.95%	84.7%	75.7%
Quality						
Quantity						
Response Time	14.2s	4.8s	5.4s	2.3s	7s	5.7s
Model RT	13.3s	4.5s	4.6s	1.9s	6.7s	5.4s
Initial Setup Effort	≈12h	≈9h	≈11h	≈4h	≈3h	≈3h
Maintenance Effort	High	High	Medium	Medium	Medium	Medium
Flexibility	High	High	Medium	High	Medium	Low
Availability	99.3%	99.85%	99.2%	99.8%	99.7%	99.8%
Max. Throughput	3VU	30VU	30VU	30VU	30VU	30VU
Time to train	≈2min	≈6h	≈2min	≈1h	≈5min	≈15min
Cost per Month	2.776 USD	1.458 USD	158 USD	5.121 USD	225 USD	105 USD

Amazon Q), **b)** Models and Tools to build generative AI applications (known as Amazon Bedrock) and **c)** Infrastructure to build and train AI models (known as SageMaker).

3.2.1. Amazon Q

Amazon Q is a service that simplifies the integration and utilization of AI-driven content and conversational interfaces. It's designed to provide natural-language assistance across tasks, from generating code and summarizing documents to answering queries informed by enterprise data and driving productivity in applications like analytics and contact center automation. It is envisioned to use if the goal is to enrich the user experience through a virtual assistant capable of real-time interaction, integrating multiple knowledge sources to provide contextual and personalized responses. It is built on top of AWS's foundational models (such as Titan) and leverages generative AI capabilities for interactive dialogue and task execution. Generally speaking Amazon Q can be classified as a *cloud-hosted customizable AI-assistant service*.

3.2.2. Amazon Bedrock

Amazon Bedrock is a fully managed AWS service for building and deploying generative AI applications by providing access to a variety of high-performance foundation models from AWS and partner providers through a unified API. It abstracts infrastructure management and allows developers to focus on experimenting, customizing via techniques like RAG, and integrating generative capabilities directly into applications without managing model hosting or scaling. Generally speaking Amazon Bedrock can be categorized as a *managed foundation model inference service*.

3.2.3. Amazon SageMaker

Amazon SageMaker is a fully managed, end-to-end cloud platform for machine learning, covering the entire ML life-cycle, from preparing data, training and tuning models, to deploying and monitoring them at scale. It supports custom model development and training as well as hosting pre-built models. In general terms, the SageMaker belongs to the *comprehensive machine learning platform category*.

Summarizing, in the Generative AI stack we chose as representative, each layer is designed to address specific and complementary needs within AI generative technologies: Amazon Q users can take advantage of an already complete solution without having to delve into complex implementation concepts needing only to make slight modifications to fit the specific project. Amazon Bedrock offers simplified integration via API and fully serverless management, removing the complexity associated with infrastructure management and allowing the team to focus on innovation and customization of AI applications and Amazon SageMaker provides an environment for building and abstracting AI models from scratch, thus requiring the highest level of expertise.

3.3. Candidate Evaluation

The candidate solutions were systematically evaluated against the requirements outlined in section 3.1, using a combination of quantitative measurements and qualitative assessment. Evaluation criteria included response accuracy, latency, scalability, ease of integration, and operational overhead. Where possible, empirical testing was conducted using representative workloads, while architectural and operational considerations were assessed through comparative analysis. The outcome of this evaluation process directly informed the selection of the components used in ADA's final architecture and the result of this evaluation is shown in Table 1.

Accuracy was assessed by analyzing 100 responses per solution and classifying each response into one of three categories: Exact Match, Semantic Match, or No Match. The proportion of responses in each category was computed, and the final accuracy score was obtained as the average across these categories. Quality was measured by assigning each response a score from 1 to 5 based on the completeness of the answer and the level of detail provided to support procedural execution. These scores were evaluated within the same three match categories, and a weighted average was calculated to derive the final quality metric. Quantity was evaluated by scoring each response on a 1 to 5 scale according to the amount of information provided, with the final score computed as a weighted average. Response Time was measured as the total time, in seconds, required to complete the full interaction flow for each solution, and an average value was calculated. Model Response Time specifically captured the time spent interacting with the AI models, also measured in seconds and averaged across runs. Initial setup time was estimated by listing all configuration steps required to deploy each solution and summing the time needed to complete them, providing an estimate of the initial implementation effort. Maintenance effort was assessed by identifying the number of activities and tools that must be monitored to ensure correct system operation; each solution was assigned a score based on the presence or absence of these activities, reflecting the ongoing effort required to keep the system functional and up to date. Flexibility was evaluated by applying real-time changes to the system and measuring how quickly each solution adapted, with individual scores aggregated to produce an overall flexibility rating. Availability was measured as the probability that the system performs its intended functions without interruption. Maximum throughput was assessed by subjecting each solution to increasing load level. Through Postman [61], the number of Virtual Users (VUs) was increased to the point where the solution was no longer able to answer calls and the maximum number for each was evaluated. Time to train was evaluated by considering the duration of the training phase, accounting for the differing nature of the approaches: fine-tuning-based solutions were trained on a dataset consisting of 10,000 data points, whereas for RAG-based solutions the time was primarily determined by the embedding process. Finally, cost was evaluated by estimating the monthly operational expenses associated with each solution, enabling comparison not only in terms of technical performance but also economic impact.

According to our evaluation, with the main goal being the overall improved workflow of the incident response team, consulting also with actual human experts, we concluded that an AI-assistant can be set up more easily for MSPs following a managed foundation model inference service. In our evaluation scenarios, the one that performed best across the board was relying on Amazon Bedrock and enhance the response using manual RAG. Amazon Bedrock allowed us to query models such as the Anthropic Claude 3 Sonnet [62] and the incorporation of the RAG pattern for providing context pertinent to our organizational use case allowed us to achieve an over 90% accuracy in the assistant's responses.

3.4. Deployment and Integration

Based on the results of the candidate evaluation, a final architecture was defined that integrates the selected components into a cohesive and modular system. This section presents the chosen architecture and explains how its individual components interact to support alarm triage, contextual reasoning, and operator interaction, especially in existing organizational workflows. Moreover, the practical deployment of ADA requires consideration of how the system integrates with existing MSP infrastructures and operational workflows. As such, particular care was given to the integration with enterprise

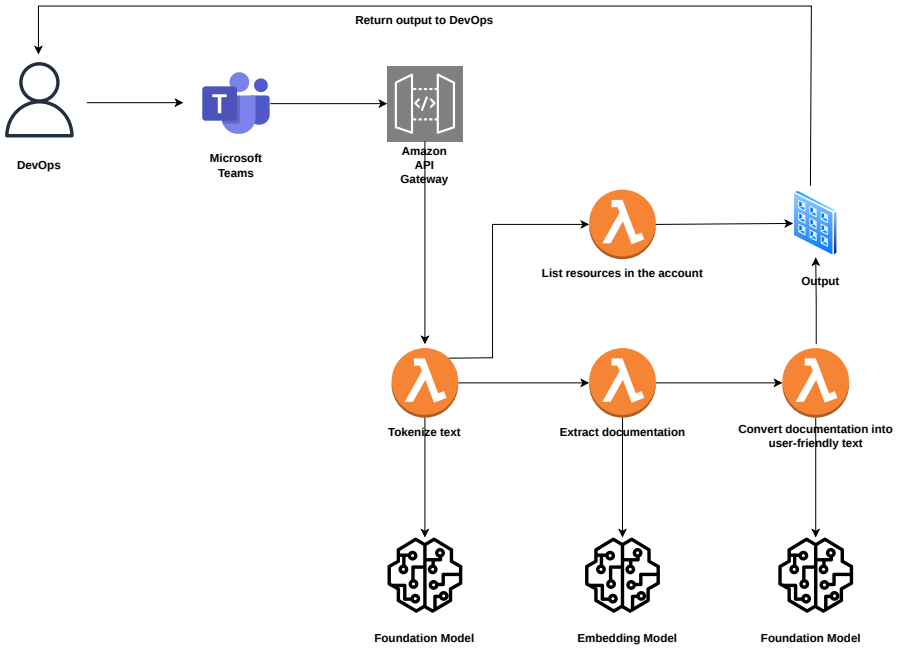

Figure 3: ADA workflow and integration in practice. The user queries on Microsoft Teams Channel, triggering an Amazon API Gateway and initiating the flow. The output/answer will be returned to the user as a comment on the original message in the chat.

communication platforms, and interaction with monitoring and alerting systems.

From a conceptual point of view, ADA can be divided into three main parts that need to be set up and implemented. The first step is the alert triggering step, which comprises of a user, i.e a member of the incident response team posing a question related to the incident at hand. This question in our implementation is posed by the user using Microsoft Teams (see section 3.4.3 below). The message posted on the Microsoft Teams chat acts then as the pipeline initialization trigger. The user message is incorporated into a request via a gateway API that is directed to an AWS Lambda function containing an LLM model that tokenizes the user input into system comprehensible tokens. After the tokenization phase, a new Lambda function is invoked relying on an embedding model in order to find the most relevant documents within the loaded company documentation (see section 3.4.1 for an overview of how company documentation is stored and processed for use in the ADA pipeline). An embedding model is a model that converts phrases or texts into numerical vectors within a given low dimensional space, capturing semantics between words. This vector representation enables the system to quantify the degree of likeness between different words or phrases, thus making it easier to find relevant documents or information from a corpus with regards to a specific query. In our architecture, after the LLM has completed the user's query, the embedding model is utilized to check this representation against the rest of the database, extracting the most relevant content so that an accurate answer can be generated. This step is of particular importance because it filters the answers and ensures that the output is consistent with the question asked. Finally, a third Lambda function containing another LLM model aims to generate a clear text based on the previously extracted documents following the RAG paradigm for response enrichment (see section 3.4.2). Following, the answer will then be returned to the user directly on Microsoft Teams. The overall architecture is depicted in Figure 3.

3.4.1. Documentation Ingestion

Documentation is uploaded in a specific shared storage accessible by the Lambda routines. Upon upload (or update) a Lambda function is triggered. This Lambda function retrieves documents from the storage such as an Amazon S3 bucket and breaks them down into chunks, for ease of management and

processing. Each chunk is then processed by the Amazon Titan model, which we use to embed the text. Next, the generated embeddings are entered into a vector database. Along with the vectors, related text chunks are also stored in the database, so that the original content can be easily traced back when needed.

3.4.2. Response Enrichment

The enrichment process is triggered by a message posted on the Microsoft Teams Channel. It begins when two parameters are passed to the system: the alarm name and the project name, which are sent via an API Gateway that triggers the corresponding Lambda function. This Lambda is tasked with processing the alarm, starting with embedding the alarm name. The alarm name, now represented as a vector, is compared in the vector database using cosine similarity. Once identified the most relevant document content, these contents are then provided as context to a the language model for augmented generation. This extra context facilitates the creation of an enriched prompt. This enriched prompt is then sent to a language model, which parses the content and provides a response based on the information extracted from the documentation and similar procedures.

3.4.3. Microsoft Teams Integration

Integration with Microsoft Teams was implemented through the use of Microsoft Power Automate, which enables enhanced workflows with automated flows. Users send requests via Microsoft Teams. Power Automate handles these requests by making an HTTP call to an API Gateway. The requests are then sent to Amazon Bedrock, where they are processed further. Finally, Power Automate provides a response in the channel, sharing the appropriate resolution procedure. The workflow, as shown begins with a trigger that detects when a user posts a message in a Teams channel. This event starts the entire process. Once the trigger is triggered, Power Automate performs a series of scheduled actions. The first action is to transform text in JSON so that it can be passed as argument for the next action. Then, it makes an HTTP call to an API Gateway, which acts as an intermediary for requests. This step is crucial since it handles communication between Teams and AWS Lambda. The requests are then forwarded to Amazon Bedrock. At this stage, Amazon Bedrock analyzes the content of the request and generates an appropriate response. Finally, Power Automate completes the workflow by sending a response to the Teams channel, providing the user with the resolution procedure or requested information. This process ensures smooth and automated interaction, improving efficiency and speed in the handling of user requests.

4. Related Work

With the increasing complexity of cloud-based and distributed IT systems, AIOps solutions have been proposed to automate incident detection, diagnosis, and resolution at scale. Early AIOps approaches primarily rely on ML and DL techniques to correlate logs, metrics, and traces in order to identify anomalous behaviors and reduce mean time to resolution. However, these methods typically require extensive manual feature engineering and domain specific preprocessing pipelines to extract meaningful signals from heterogeneous operational data. As a consequence, scalability is limited and adaptation to new environments or evolving workloads often requires substantial reconfiguration. Zhang et al. [63] systematically analyze these limitations, showing that ML/DL-based AIOps systems are generally effective only in early incident triage phases, while root-cause analysis and remediation remain largely dependent on human expertise. To overcome these constraints, recent research has investigated the integration of LLMs and RAG techniques into AIOps pipelines. By enabling semantic reasoning over unstructured data and natural language interaction, these approaches aim to reduce the reliance on handcrafted features and improve diagnostic flexibility. Zhou et al. [64, 65] propose LLMDB and DB-GPT, two frameworks that incorporate database specific knowledge, such as technical manuals, operational logs, and tool specifications into LLM based systems for diagnosis and anomaly analysis. LLMDB

introduces an end-to-end paradigm that converts domain documentation into embeddings, orchestrates tool call pipelines, and uses executor agents to compose metric, activity, and workload analyzes, with an emphasis on vector database caching to reduce inference cost. DB-GPT complements this approach by focusing on prompt generation strategies, database specific fine tuning, and model designs that embed non textual features such as data distributions and query graphs, while enforcing output validity for database operations. Although these systems demonstrate the feasibility of grounding LLMs in operational knowledge, they are primarily evaluated as research prototypes and focus on diagnostic reasoning and optimization tasks in controlled settings. Their architectures rely largely on static documentation and offline embedding pipelines, and they do not address the orchestration of complex, multi turn incident workflows in production environments. Moreover, automation is generally limited to recommendation or explanation, without direct support for coordinated remediation actions. Isaza et al. [66] propose a RAG-based generative system for IT support ticket resolution, targeting the automatic generation of solution recommendations. Their approach focuses on single-turn tickets that can be resolved without iterative interaction, leveraging a static corpus of internal documentation. Although effective in simplified support scenarios, the system does not address the challenges posed by dynamic, multi-turn tickets or continuously evolving enterprise knowledge bases. Chen et al. [67] introduced RCACopilot, a production oriented framework for cloud incident root cause analysis that integrates automated multi source diagnostic collection, including logs, traces, and metrics. RCACopilot employs summarization to handle large diagnostic dumps and uses time aware nearest neighbor selection combined with Chain of Thought prompting to enable LLMs to predict root cause categories and generate structured explanations. This approach demonstrates strong capabilities in automated data collection and diagnostic reasoning, but its primary focus remains on classification and explanation rather than remediation orchestration. Zhang et al. [68] further extend LLM-based AIOps research with RAG4ITOps, a supervised and fine tunable RAG framework specifically designed for IT operations and maintenance. Their approach relies on an extensive offline pipeline, including document preprocessing, chunking, data distillation, contrastive learning–based embedding fine-tuning, and retrieval-augmented fine-tuning of an LLM. Although RAG4ITOps achieves strong accuracy in knowledge acquisition and troubleshooting tasks, its architecture remains training-intensive and oriented toward static, curated corpora, limiting adaptability in rapidly changing production environments.

Compared to these approaches, ADA is designed for real, dynamic, and often multi turn production tickets operating across distributed cloud infrastructures. Unlike DL-based AIOps systems, ADA does not require manual feature engineering, and unlike training centric RAG frameworks such as RAG4ITOps, it avoids frequent fine-tuning by prioritizing dynamic retrieval from continuously updated enterprise knowledge sources. In contrast to diagnosis or recommendation focused systems, ADA supports higher degrees of automation, including solution orchestration and optional auto-remediation. Furthermore, ADA is implemented as a fully AWS-native pipeline leveraging managed LLM and embedding services, enabling seamless integration with operational tools and scalable deployment. These characteristics position ADA as a production oriented AIOps solution that extends beyond incident triage moving towards end-to-end operational automation.

5. Conclusions and Future Work

This paper presented a comparative study of several approaches for enhancing operational efficiency of incident response for Managed Service Providers in cloud-based security environments highlighting the most efficient approach into a fully-fledged AI-powered chatbot system named ADA. Through the incorporation of domain-specific documentation and a retrieval-augmented generation mechanism, ADA delivers contextually accurate and operationally relevant responses to a wide spectrum of user queries, spanning alarm resolution, incident triage, and infrastructure monitoring tasks. Empirical evaluation demonstrated substantial enhancements in incident management workflows. The alarm triage component achieved an accuracy of 91.35%, contributing to decreased response times and allowing analysts to redirect attention toward higher-complexity activities. Furthermore, we showed that such

approach can be seamlessly integrated with existing operational platforms showing that ADA not only increases efficiency but also improves the usability and intuitiveness of routine security operations.

Acknowledgments

This work was partially supported by project SERICS (PE00000014) under the NRRP MUR program funded by the EU-NextGenerationEU and REPLY company.

Declaration on Generative AI

During the preparation of this work, the author(s) used Grammarly in order to: Grammar and spelling check. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] SAP, What is a managed services provider (msp)?, 2025. URL: https://www.sap.com/products/hcm/workforce-management/what-is-a-msp.html.

[2] Exabeam, Exabeam siem, 2025. URL: https://www.exabeam.com/capabilities/siem/.

[3] Splunk, Splunk siem, 2025. URL: https://www.splunk.com/en_us/products/enterprise-security-essentials.html.

[4] LogRythm, Logrythm siem, 2025. URL: https://docs.logrhythm.com/lrsiem/docs/.

[5] IBM, Q-radar siem, 2025. URL: https://www.ibm.com/it-it/products/qradar-siem.

[6] Microsoft, Microsoft azure sentinel, 2025. URL: https://learn.microsoft.com/en-us/azure/sentinel/.

[7] Elastic, Elastic siem, 2025. URL: https://www.elastic.co/security/siem.

[8] D. H. Hagos, R. Battle, D. B. Rawat, Recent advances in generative ai and large language models: Current status, challenges, and perspectives, IEEE Transactions on Artificial Intelligence 5 (2024) 5873–5893. doi:10.1109/TAI.2024.3444742.

[9] T. Nagano, G. Kurata, S. Thomas, H.-K. J. Kuo, D. Bolanos, H. Jung, G. Saon, Llm based text generation for improved low-resource speech recognition models, in: ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2025, pp. 1–5. doi:10.1109/ICASSP49660.2025.10888566.

[10] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, L. Zettlemoyer, BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, in: D. Jurafsky, J. Chai, N. Schluter, J. Tetreault (Eds.), Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Online, 2020, pp. 7871–7880. URL: https://aclanthology.org/2020.acl-main.703/. doi:10.18653/v1/2020.acl-main.703.

[11] N. Kalchbrenner, P. Blunsom, Recurrent continuous translation models, in: D. Yarowsky, T. Baldwin, A. Korhonen, K. Livescu, S. Bethard (Eds.), Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Seattle, Washington, USA, 2013, pp. 1700–1709. URL: https://aclanthology.org/D13-1176/.

[12] K. Papineni, S. Roukos, T. Ward, W.-J. Zhu, Bleu: a method for automatic evaluation of machine translation, in: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, ACL '02, Association for Computational Linguistics, USA, 2002, p. 311–318. URL: https://doi.org/10.3115/1073083.1073135. doi:10.3115/1073083.1073135.

[13] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark,

C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language models are few-shot learners, in: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems, volume 33, Curran Associates, Inc., 2020, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[14] X. Du, M. Liu, K. Wang, H. Wang, J. Liu, Y. Chen, J. Feng, C. Sha, X. Peng, Y. Lou, Evaluating large language models in class-level code generation, in: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering, ICSE '24, Association for Computing Machinery, New York, NY, USA, 2024. URL: https://doi.org/10.1145/3597503.3639219. doi:10.1145/3597503.3639219.

[15] M. A. Ferrag, M. Ndhlovu, N. Tihanyi, L. C. Cordeiro, M. Debbah, T. Lestable, N. S. Thandi, Revolutionizing cyber threat detection with large language models: A privacy-preserving bert-based lightweight model for iot/iiot devices, IEEe Access 12 (2024) 23733–23750.

[16] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, et al., A survey of large language models, arXiv preprint arXiv:2303.18223 1 (2023).

[17] Y. Yao, J. Duan, K. Xu, Y. Cai, Z. Sun, Y. Zhang, A survey on large language model (llm) security and privacy: The good, the bad, and the ugly, High-Confidence Computing 4 (2024) 100211. URL: http://dx.doi.org/10.1016/j.hcc.2024.100211. doi:10.1016/j.hcc.2024.100211.

[18] F. R. Alzaabi, A. Mehmood, A review of recent advances, challenges, and opportunities in malicious insider threat detection using machine learning methods, IEEE Access 12 (2024) 30907–30927. doi:10.1109/ACCESS.2024.3369906.

[19] R. Fang, R. Bindu, A. Gupta, D. Kang, Llm agents can autonomously exploit one-day vulnerabilities, arXiv preprint arXiv:2404.08144 (2024).

[20] S. Jamal, H. Wimmer, An improved transformer-based model for detecting phishing, spam, and ham: A large language model approach, 2023. URL: https://arxiv.org/abs/2311.04913. arXiv:2311.04913.

[21] M. A. K. Raiaan, M. S. H. Mukta, K. Fatema, N. M. Fahad, S. Sakib, M. M. J. Mim, J. Ahmad, M. E. Ali, S. Azam, A review on large language models: Architectures, applications, taxonomies, open issues and challenges, IEEE Access 12 (2024) 26839–26874. doi:10.1109/ACCESS.2024.3365742.

[22] Y. Hua, H. Na, Z. Li, F. Liu, X. Fang, D. Clifton, J. Torous, A scoping review of large language models for generative tasks in mental health care, npj Digital Medicine 8 (2025) 230.

[23] Y. Li, S. Wang, H. Ding, H. Chen, Large language models in finance: A survey, 2024. URL: https://arxiv.org/abs/2311.10723. arXiv:2311.10723.

[24] N. Maksoud, H. AlJassmi, L. Ali, A. R. Masoud, Applications of large language models and generative ai in transportation: A systematic review and bibliometric analysis, Transportation Research Interdisciplinary Perspectives 34 (2025) 101699. URL: https://www.sciencedirect.com/science/article/pii/S2590198225003781. doi:https://doi.org/10.1016/j.trip.2025.101699.

[25] T. M. Mitchell, Machine Learning, 1 ed., McGraw-Hill, Inc., New York, NY, USA, 1997.

[26] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016. http://www.deeplearningbook.org.

[27] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 2014.

[28] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016) 770–778.

[29] F. Chollet, Xception: Deep learning with depthwise separable convolutions, 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017) 1800–1807.

[30] Y. Kim, Convolutional neural networks for sentence classification, CoRR abs/1408.5882 (2014). URL: http://arxiv.org/abs/1408.5882. arXiv:1408.5882.

[31] D. Chen, C. D. Manning, A fast and accurate dependency parser using neural networks, in: EMNLP, 2014.

[32] T. Bansal, D. Belanger, A. McCallum, Ask the GRU: Multi-task learning for deep text recommendations, in: proceedings of the 10th ACM Conference on Recommender Systems, 2016, pp. 107–114.

[33] A. Graves, A. Mohamed, G. Hinton, Speech recognition with deep recurrent neural networks, in: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, 2013, pp.

6645–6649. doi:10.1109/ICASSP.2013.6638947.

[34] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, B. Kingsbury, Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups, IEEE Signal Processing Magazine 29 (2012) 82–97. doi:10.1109/MSP.2012.2205597.

[35] J. Menick, N. Kalchbrenner, Generating high fidelity images with subscale pixel networks and multidimensional upscaling, in: International Conference on Learning Representations, 2019. URL: https://openreview.net/forum?id=HylzTiC5Km.

[36] T. Karras, S. Laine, T. Aila, A style-based generator architecture for generative adversarial networks, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 4396–4405. doi:10.1109/CVPR.2019.00453.

[37] G. Pagnotta, D. Hitaj, F. De Gaspari, L. V. Mancini, Passflow: guessing passwords with generative flows, in: 2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), IEEE, 2022, pp. 251–262.

[38] J.-H. Jacobsen, A. Smeulders, E. Oyallon, i-RevNet: Deep Invertible Networks, in: ICLR 2018 - International Conference on Learning Representations, Vancouver, Canada, 2018. URL: https://hal.archives-ouvertes.fr/hal-01712808.

[39] F. De Gaspari, D. Hitaj, L. V. Mancini, Have you poisoned my data? defending neural networks against data poisoning, in: European Symposium on Research in Computer Security, Springer, 2024, pp. 85–104.

[40] D. Hitaj, G. Pagnotta, F. De Gaspari, S. Ruko, B. Hitaj, L. V. Mancini, F. Perez-Cruz, Do you trust your model? emerging malware threats in the deep learning ecosystem, arXiv preprint arXiv:2403.03593 (2024).

[41] F. De Gaspari, D. Hitaj, G. Pagnotta, L. De Carli, L. V. Mancini, The naked sun: Malicious cooperation between benign-looking processes, in: International Conference on Applied Cryptography and Network Security, Springer, 2019, pp. 254–274.

[42] D. P. Kingma, M. Welling, An introduction to variational autoencoders, Foundations and Trends in Machine Learning 12 (2019) 307–392. URL: http://dx.doi.org/10.1561/2200000056. doi:10.1561/2200000056.

[43] A. Oussidi, A. Elhassouny, Deep generative models: Survey, in: 2018 International Conference on Intelligent Systems and Computer Vision (ISCV), 2018, pp. 1–8. doi:10.1109/ISACV.2018.8354080.

[44] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial networks, Communications of the ACM 63 (2020) 139–144.

[45] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, F. Huang, et al., A tutorial on energy-based learning, Predicting structured data 1 (2006).

[46] J. Ngiam, Z. Chen, P. W. Koh, A. Y. Ng, Learning deep energy models, in: Proceedings of the 28th international conference on machine learning (ICML-11), 2011, pp. 1105–1112.

[47] D. Rezende, S. Mohamed, Variational inference with normalizing flows, in: International conference on machine learning, PMLR, 2015, pp. 1530–1538.

[48] I. Kobyzev, S. J. Prince, M. A. Brubaker, Normalizing flows: An introduction and review of current methods, IEEE transactions on pattern analysis and machine intelligence 43 (2020) 3964–3979.

[49] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, S. Ganguli, Deep unsupervised learning using nonequilibrium thermodynamics, in: International conference on machine learning, pmlr, 2015, pp. 2256–2265.

[50] J. Ho, A. Jain, P. Abbeel, Denoising diffusion probabilistic models, Advances in neural information processing systems 33 (2020) 6840–6851.

[51] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, I. Polosukhin, Attention is all you need, in: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, volume 30, Curran Associates, Inc., 2017.

[52] A. Radford, K. Narasimhan, Improving language understanding by generative pre-training, 2018.

URL: https://api.semanticscholar.org/CorpusID:49313245.

[53] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, CoRR abs/1810.04805 (2018). URL: http://arxiv.org/abs/1810.04805. arXiv:1810.04805.

[54] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al., Language models are unsupervised multitask learners, OpenAI blog 1 (2019) 9.

[55] B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, et al., Language models are few-shot learners, arXiv preprint arXiv:2005.14165 1 (2020) 3.

[56] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, G. Lample, Llama: Open and efficient foundation language models, 2023. URL: https://arxiv.org/abs/2302.13971. arXiv:2302.13971.

[57] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, D. Kiela, Retrieval-augmented generation for knowledge-intensive nlp tasks, in: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems, volume 33, Curran Associates, Inc., 2020, pp. 9459–9474. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf.

[58] P. P. Mishra, K. P. Yeole, R. Keshavamurthy, M. B. Surana, F. Sarayloo, A systematic framework for enterprise knowledge retrieval: Leveraging llm-generated metadata to enhance rag systems, 2025. URL: https://arxiv.org/abs/2512.05411. arXiv:2512.05411.

[59] A. Tellache, A. A. Korba, A. Mokhtari, H. Moldovan, Y. Ghamri-Doudane, Advancing autonomous incident response: Leveraging llms and cyber threat intelligence, 2025. URL: https://arxiv.org/abs/2508.10677. arXiv:2508.10677.

[60] Z. Liu, A. Anwar, Autobnb-rag: Enhancing multi-agent incident response with retrieval-augmented generation, 2025. URL: https://arxiv.org/abs/2508.13118. arXiv:2508.13118.

[61] P. Inc., Postman test automation, 2025. URL: https://www.postman.com/solutions/test-automation/.

[62] Anthorpic, Claude 3 sonnet, 2025. URL: https://www.anthropic.com/news/claude-3-5-sonnet.

[63] L. Zhang, T. Jia, M. Jia, Y. Wu, A. Liu, Y. Yang, Z. Wu, X. Hu, P. S. Yu, Y. Li, A survey of aiops for failure management in the era of large language models, 2024. URL: https://arxiv.org/abs/2406.11213. arXiv:2406.11213.

[64] X. Zhou, Z. Sun, G. Li, Db-gpt: Large language model meets database, Data Science and Engineering 9 (2024) 102–111.

[65] X. Zhou, X. Zhao, G. Li, Llm-enhanced data management, 2024. URL: https://arxiv.org/abs/2402.02643. arXiv:2402.02643.

[66] P. T. Isaza, M. Nidd, N. Zheutlin, J. wook Ahn, C. A. Bhatt, Y. Deng, R. Mahindru, M. Franz, H. Florian, S. Roukos, Retrieval augmented generation-based incident resolution recommendation system for it support, 2024. URL: https://arxiv.org/abs/2409.13707. arXiv:2409.13707.

[67] Y. Chen, H. Xie, M. Ma, Y. Kang, X. Gao, L. Shi, Y. Cao, X. Gao, H. Fan, M. Wen, et al., Automatic root cause analysis via large language models for cloud incidents, in: Proceedings of the Nineteenth European Conference on Computer Systems, 2024, pp. 674–688.

[68] T. Zhang, Z. Jiang, S. Bai, T. Zhang, L. Lin, Y. Liu, J. Ren, Rag4itops: A supervised fine-tunable and comprehensive rag framework for it operations and maintenance, 2025. URL: https://arxiv.org/abs/2410.15805. arXiv:2410.15805.