Journal of Information Systems Engineering and Management
2025, 10(13s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

LLM for Automated Root Cause Analysis in Microservices Architectures

Amruta Mhatre , Afrid Shaikh
[1]*John College of Engineering & Management (SJCEM) Palghar, Mumbai, India*
[2]*John College of Engineering & Management (SJCEM) Palghar, Mumbai, India*
amruta.mhatre@vcet.edu.in

ARTICLE INFO

Received: 27 Nov 2024

Revised: 05 Jan 2025

Accepted: 30 Jan 2025

ABSTRACT

The adoption of microservices architectures has introduced significant challenges in diagnosing and resolving system issues efficiently, as multiple services handling millions of requests generate vast volumes of exceptions, including business errors and critical runtime failures. Traditional manual approaches for error analysis and Root Cause Analysis (RCA) are time-consuming, error-prone and lack scalability. Existing tools often aggregate exceptions but fail to effectively classify or diagnose root causes, leading to prolonged system downtimes and reduced developer productivity. This research proposes an automated solution leveraging fine-tuned Large Language Models (LLMs), combined with an Exception Classifier powered by Natural Language Processing (NLP) and machine learning techniques. The system adopts a layered architecture where exceptions are aggregated through a Kafka cluster. The Exception Classifier preprocesses error messages, extracts contextual information and categorizes exceptions into business and runtime errors. Classified runtime errors are forwarded to the RCA service, where fine-tuned LLMs perform detailed diagnostics by analyzing exception stack trace and tokenized code repository. The solution targets over 90% precision for business exceptions and 89.6% recall for runtime exceptions in Exception Classification and less than a second for RCA diagnostics per exception and gives over 85% accuracy in human qualitative evaluation. By automating error classification and RCA, the proposed system promises faster fault resolution, improved RCA accuracy and enhanced developer productivity, contributing to more resilient and efficient microservices environments.

Keywords: Classification, Exception, Fine Tuning, Microservice, Large Learning Model, Root Cause Analysis.

INTRODUCTION

Microservices architectures have become the backbone of modern software development, enabling organizations to address complex business challenges by decomposing applications into independently deployable and scalable services. This modularity, while offering numerous advantages, introduces the challenge of managing a vast number of system exceptions [1]. These exceptions range from predictable business errors to critical runtime failures that can disrupt service operations and negatively impact user experiences [2]. Analysing and resolving such exceptions manually is labour-intensive, error-prone and incapable of keeping up with the dynamic nature of microservices environments. Traditional tools primarily focus on exception aggregation but lack automated, granular classification and diagnostic capabilities, resulting in prolonged system downtimes and decreased developer productivity [3].

To address these limitations, this project proposes an automated solution that integrates Large Language Models (LLMs) fine-tuned for Root Cause Analysis (RCA) and a machine learning-powered Exception Classifier for efficient error categorization. Leveraging Kafka as a centralized exception aggregation hub, the system will preprocess error messages, extract critical contextual information and classify exceptions into business and runtime categories. The classified runtime errors will be forwarded to the RCA service, where LLMs will perform detailed diagnostics by analysing system logs and traces.

This innovative architecture aims to achieve key technical performance objectives, including a classification precision and recall of over 90% and an RCA diagnostic latency of less than 10 seconds per exception. By ensuring real-time processing, scalability and modularity, the solution will enable faster fault resolution and improved operational efficiency.

The importance of this project extends beyond technical benefits. Developers will gain from reduced cognitive load and increased focus on critical tasks, while organizations will experience improved system reliability, reduced operational costs and enhanced service delivery. By automating exception classification and RCA, this project addresses a critical gap in existing microservices solutions and offers a scalable, AI-driven approach to maintaining resilient and efficient software systems.

PROBLEM DEFINITION

As software development increasingly adopts microservices architectures for scalable and modular applications, diagnosing and resolving system issues has become a critical challenge [1]. The complexity of microservices environments, with millions of requests per minute and vast amounts of operational data, including errors and exceptions, makes efficient issue identification and resolution difficult [4]. These errors, which can be categorized into business exceptions and runtime exceptions, are exacerbated by the distributed nature of microservices and their interdependencies [2].

Traditional manual approaches to Root Cause Analysis (RCA) are no longer feasible due to the overwhelming volume of logs, the diversity of error types, and the need for real-time diagnostics. Current solutions, such as rule-based anomaly detection systems and agent-based cloud incident response tools, lack scalability and do not effectively automate RCA or accurately classify exceptions [5]. Furthermore, these methods fail to provide actionable, real-time insights, resulting in prolonged system downtime, decreased service reliability, and reduced developer productivity.

To overcome these challenges, there is a need for an automated and scalable RCA framework that integrates advanced AI models, such as Large Language Models (LLMs), and an Exception Classifier powered by Natural Language Processing (NLP) and machine learning algorithms. This framework must enable real-time processing, adaptive diagnostics, and precise exception categorization to improve system resilience and operational efficiency.

LITERATURE SURVEY

Shilin He et al. (2024) provides a comprehensive review of Root Cause Analysis (RCA) methodologies in microservice environments, characterized by their dynamic dependencies and complex fault propagation. It highlights the evolution from manual operations to DevOps and AIOps, where AI techniques are employed for automated fault diagnosis. The survey categorizes various diagnostic approaches using metrics, logs, traces, and multi-model data to identify system failures. It emphasizes the critical role of RCA in mitigating system disruptions and ensuring rapid recovery. The authors also explore current challenges, such as the difficulty in tracing faults across microservices and the need for more sophisticated AI-driven techniques, while offering insights into future research directions to enhance fault tolerance and system resilience [6].

Luan Pham et al. (2024) presented BARO, an end-to-end framework for root cause analysis in microservices that integrates anomaly detection with advanced statistical methods. Their approach leverages Multivariate Bayesian Online Change Point Detection to effectively model dependencies within complex multivariate time-series data, enabling timely detection of abrupt changes that signal potential anomalies. In addition, BARO employs a novel nonparametric hypothesis testing technique that enhances the robustness of change point identification without assuming a fixed data distribution. Experiments on benchmark microservice systems demonstrated that BARO outperforms current state-of-the-art methods in both anomaly detection and root cause identification, though further work is needed to assess its performance in real-time, large-scale environments [4].

Cheng-Ming Lin et al. (2024) proposed RUN, a novel approach for root cause analysis in microservices that utilizes neural Granger causal discovery enhanced by contrastive learning. Their method integrates contextual time-series data with deep forecasting models to capture complex temporal patterns in service metrics. By applying a modified PageRank algorithm—with an added personalization vector—to the forecasted data, RUN efficiently ranks potential root causes. Extensive experiments on synthetic and real-world datasets showed that RUN achieves superior diagnostic performance compared to existing techniques, although additional validation across varied operational conditions is required [7].

Ruomeng Ding et al. (2023) explored TraceDiag, an adaptive and interpretable framework for root cause analysis in large-scale microservice systems. TraceDiag uses reinforcement learning to automatically learn a pruning policy for service dependency graphs derived from real-time traces and logs, thereby reducing the search space and focusing on the most critical components. This approach not only enhances diagnostic efficiency and accuracy but also maintains high interpretability, enabling engineers to understand the underlying reasoning. Evaluations on data from systems like Microsoft Exchange demonstrated significant improvements over traditional RCA approaches, even as the complexity introduced by reinforcement learning suggests a need for further simplification for broader deployment [8].

Yidan Wang et al. (2024) introduced MRCA, a metric-level root cause analysis framework that leverages multi-modal data—including traces, logs, and performance metrics—to construct causal graphs that capture the sequential occurrence of anomalies across services. A key innovation in MRCA is its reward mechanism, which effectively terminates unnecessary expansion of the causal graph once sufficient evidence is gathered, thus improving both efficiency and accuracy. Experiments on widely recognized microservice benchmarks revealed that MRCA outperforms state-of-the-art approaches in diagnostic accuracy and processing speed, although its performance in real-time scenarios and scalability to larger environments warrants further investigation [3].

Zhouruixing Zhu et al. (2024) presented HeMiRCA, a fine-grained root cause analysis framework specifically designed for microservices that harnesses heterogeneous data sources to deliver detailed diagnostic insights. By employing statistical techniques such as Spearman correlation along with anomaly-aware monotonic correlation analysis, HeMiRCA effectively ranks suspicious metrics at both service and metric levels. This comprehensive integration of logs, traces, and performance metrics allows for precise identification of root causes, and evaluations on real-world datasets indicate that HeMiRCA outperforms several current state-of-the-art methods. Nonetheless, additional research is needed to validate its robustness across diverse microservices environments and operational conditions [9].

Toufique Ahmed (2023) investigates the application of LLMs, specifically GPT-3.x models, to assist on-call engineers in diagnosing and resolving cloud incidents. Conducted at Microsoft, the study evaluates over 44,000 incidents across 1,759 cloud services. The authors explore various configurations, including zero-shot, fine-tuned, and multi-task learning, to assess model performance in root cause and mitigation step recommendation. Human validation with incident owners confirms the efficacy of the models, with fine-tuned GPT-3.x significantly outperforming encoder-decoder models. The paper underscores the potential of AI-driven solutions for reducing manual effort and improving cloud service reliability [5].

Xuanming Zhang (2024) addresses challenges in exception handling in software development, which often leads to fragile and unreliable code. The authors propose SEEKER, a multi-agent framework inspired by expert developer strategies. It utilizes five agents—Scanner, Detector, Predator, Ranker, and Handler—to detect, capture, and resolve exceptions. By integrating LLMs with exception handling best practices, SEEKER enhances code robustness and mitigates common pitfalls such as insensitive detection and inaccurate exception capture. The framework demonstrates significant improvements in exception handling performance, providing valuable insights for future advancements in reliable code generation [10].

Devjeet Roy (2024) investigates the use of LLM-based agents for RCA in cloud environments, where incident diagnosis is increasingly complex. The authors evaluate a ReAct agent equipped with retrieval tools for handling out-of-distribution production incidents. The agent demonstrates competitive performance compared to baseline models while achieving higher factual accuracy. A case study with a Microsoft team shows the practical application of these agents, highlighting their ability to interact with diagnostic services and dynamically collect new information. The study emphasizes the potential of LLM-based agents to improve RCA by reducing the manual burden on engineers and enhancing diagnostic accuracy [11].

Yingying Zhang et al (2021) presented CloudRCA, a comprehensive framework for automated root cause analysis tailored to cloud computing platforms. CloudRCA integrates multiple data sources—including logs, metrics, and traces—to create a multi-dimensional view of system behavior and applies machine learning techniques to detect anomalies and correlate them with potential failure causes. The framework was evaluated in cloud environments, where it significantly reduced the time required to diagnose incidents and improved overall system reliability. While CloudRCA demonstrates strong potential in streamlining incident resolution, the authors highlight the need for enhanced scalability and real-time processing to better support large-scale deployments [12].

Álvaro Brandón et al (2022) explored a graph-based approach for root cause analysis in service-oriented and microservice architectures by constructing dependency graphs that model the interactions between services. Their method applies network analysis techniques to traverse these graphs and identify critical nodes and fault propagation paths that contribute to system failures. Evaluations on real-world case studies indicate that the approach enhances the accuracy and speed of root cause identification compared to traditional correlation-based methods. Despite its promising results, the study acknowledges challenges in scaling the technique for very large and dynamic systems, suggesting the need for further optimization [13].

Haixuan Guo et al (2021) presented LogBERT, a transformer-based framework for log anomaly detection that leverages the BERT architecture to capture contextual and semantic patterns in system logs. Their approach involves pre-training BERT on large-scale log data and subsequently fine-tuning it on labeled datasets for anomaly detection, enabling robust identification and classification of anomalous events. Experimental results on benchmark log datasets demonstrate that LogBERT achieves improved precision and recall compared to traditional log analysis methods, highlighting its potential to enhance real-time log monitoring. However, further research is needed to address scalability challenges and facilitate seamless integration into production environments [14].

Supriyo Ghosh (2022) investigates 152 high-severity production incidents in Microsoft Teams to understand their causes, detection, and mitigation strategies. The authors identify software bugs, infrastructure issues, and manual errors as primary root causes. The study reveals that over 90% of incidents are mitigated without code changes, using strategies such as service rollbacks and configuration adjustments. The findings emphasize the importance of automated monitoring and better detection mechanisms to reduce response time. The study offers practical insights into optimizing incident response and highlights opportunities for automation to improve cloud service reliability [15].

Wei Zhang (2024) introduces MABC, a novel multi-agent framework inspired by blockchain principles for RCA in microservices architectures. The framework integrates seven specialized agents that collaborate through a decentralized decision-making process to identify root causes of faults. By leveraging blockchain-inspired voting and limiting steps to avoid circular dependencies, MABC enhances fault detection and resolution. Experimental results demonstrate superior performance in RCA and effective resolution generation compared to traditional methods. The study highlights the importance of agent collaboration and decentralized decision-making for robust RCA in complex microservices environments [16].

Mathav Raj J (2024) provides practical guidelines for fine-tuning LLMs to meet enterprise needs, focusing on proprietary datasets for documentation and code. The authors discuss the advantages and limitations of fine-tuning compared to Retrieval-Augmented Generation (RAG) methods, highlighting scenarios where fine-tuning offers superior accuracy despite higher resource costs. Techniques such as Low-Rank Adaptation (LORA) and data preprocessing strategies are recommended to optimize training. The study offers insights into configuring LLMs for enterprise applications, guiding beginners on dataset preparation, GPU requirements, and efficient fine-tuning techniques [17].

Pawan Kumar Sarika (2023) focuses on automating test failure analysis in microservices environments using Kubernetes cluster logs. As manual classification of failures becomes increasingly time-consuming, the authors evaluate five machine learning algorithms, including Support Vector Machines, Random Forest, and Gradient Boosting Classifier, to determine their efficiency in diagnosing failures. The results highlight Random Forest as a top performer, achieving high accuracy with low computational resource requirements. The paper underscores the benefits of automation in reducing analysis time and improving the reliability of microservices, offering valuable insights for organizations managing large-scale distributed systems [18].

Ahmed Saeed Alsayed (2024) presents MicroRec, a novel microservice recommendation framework designed to help developers discover relevant microservices in large ecosystems. Leveraging Large Language Models (LLMs), the system utilizes a dual-encoder architecture combining contrastive and semantic similarity learning to enhance recommendation accuracy. The framework integrates Stack Overflow posts, Dockerfile, and README information to better understand queries and available microservices. Empirical evaluations demonstrate significant improvements in recommendation precision and relevance compared to existing tools like Docker Hub. MicroRec is 14 times more accurate than traditional methods, highlighting its practical value in microservice discovery, evaluation, and compatibility [19].

Sayar Ul Hassana (2022) presents a detailed comparative analysis of various machine learning algorithms for text classification, including Support Vector Machine (SVM), k-Nearest Neighbor (k-NN), Logistic Regression (LR), Multinomial Naïve Bayes (MNB), and Random Forest (RF). The authors evaluate these algorithms on two different datasets: IMDB for sentiment analysis and a SPAM dataset for message categorization. The study assesses performance based on metrics like accuracy, precision, recall, and F1-score. The results indicate that Logistic Regression and SVM excel for the IMDB dataset, while kNN outperforms others for the SPAM dataset. The paper emphasizes the importance of selecting appropriate algorithms based on dataset characteristics and highlights the growing role of text classification in automating data analysis for enterprise applications [20].

Iman Kohyarnejadfard (2022) addresses the challenges of anomaly detection in microservice environments, which are prone to performance anomalies due to their complexity and distributed architecture. The authors propose an NLP-based approach that leverages distributed tracing data to analyze event sequences and detect anomalies without requiring prior system knowledge. The approach also identifies release-over-release regressions, providing valuable insights for performance monitoring. Extensive experiments on real-world datasets demonstrate high accuracy, with an F-score of 0.9759. The study showcases how visualization tools can further expedite root cause analysis, ultimately enhancing system resilience and reducing manual diagnostic efforts [21].

Shilin He (2021) explores the significance of automated log analysis as a critical tool for ensuring system reliability in cloud services. It emphasizes best practices in log management, such as adhering to logging standards, maintaining appropriate verbosity levels, and ensuring proper log aggregation. A key challenge highlighted is tracing log cycles across interdependent services, which can hinder efficient problem diagnosis. The study discusses the importance of using event IDs and centralized log storage to facilitate effective analysis. The authors recommend safeguarding logs due to the sensitive information they may contain and outline emerging research directions for improving automated log analysis systems, ultimately aiming to enhance reliability through better fault detection and faster RCA [6].

Baptiste Rozière (2024) presents Code Llama, a family of large language models (LLMs) by Meta AI designed for code generation, completion, and documentation tasks. Based on the Llama 2 architecture, it offers general, Python-optimized, and instruction-following variants with parameter sizes up to 70B and support for 100,000-token input contexts. Trained on up to 1 trillion tokens, the models utilize fine-tuning techniques such as infilling and instruction tuning, achieving state-of-the-art performance on benchmarks like HumanEval and MBPP. Released under a permissive license, Code Llama aims to enhance developer productivity and foster innovation in AI-driven software development [22].

Muhammad Waseem (2023) investigates the issues, causes, and solutions encountered in microservices systems. The authors collected data from 2,641 issues in 15 open-source microservices projects, conducted 15 interviews, and surveyed 150 practitioners. They developed comprehensive taxonomies categorizing 19 types of issues, 8 categories of causes, and 177 types of solutions. Key findings highlight technical debt, exception handling, and invalid configurations as prevalent challenges. The study provides actionable insights for developers and researchers to improve microservices design and maintenance by addressing recurring issues and adopting effective fixing strategies [2].

Xuchao Zhang (2024) explores the application of GPT-4 for automated RCA in large-scale cloud services using in-context learning. The authors present an innovative approach that eliminates the need for fine-tuning by leveraging historical incident examples directly during inference. Extensive evaluations across over 100,000 production incidents demonstrate a 24.8% improvement over fine-tuned GPT-3 models and a 49.7% improvement over zero-shot models. Human evaluations validate the approach, with a 43.5% increase in correctness and an 8.7% boost in readability. The study underscores the cost-efficiency and adaptability of in-context learning for RCA, offering a scalable solution for incident diagnosis in complex cloud environments [23].

4.1 Comparative study table

Table 4.1.1: comparative study table

Sr. No	**Title of Paper**	**Author(s)**	**Year**	**Methodology & Technology Used**	**Outcome**	**Gap Identified**
1	A Comprehensive Survey on Root Cause Analysis in (Micro) Services: Methodologies, Challenges, and Trends	Tingting Wang et al	2024	Literature survey on RCA methods; AI/ML techniques (anomaly detection, dependency mapping, multi-model data analysis)	Categorized RCA techniques and highlighted AI-driven methods	Limited automated RCA solutions for microservices with comprehensive exception handling
2	BARO: Robust Root Cause Analysis for Microservices via Multivariate Bayesian Online Change Point Detection	Luan Pham et al	2024	End-to-end RCA using Multivariate Bayesian Online Change Point Detection and nonparametric hypothesis testing for anomaly detection and RCA	Regularly surpasses cutting-edge techniques across three benchmark microservice systems.	Further evaluation needed in real-time, large-scale microservices environments
3	Root Cause Analysis in Microservices Using Neural Granger Causal Discovery	Cheng-Ming Lin et al	2024	Neural Granger causal discovery enhanced by contrastive learning; integration of time-series forecasting and modified PageRank for top-k root cause ranking	Exceeds the performance of advanced RCA methods on both synthetic and real-world datasets.	Requires further validation in diverse microservices architectures and operational conditions
4	TraceDiag: Adaptive, Interpretable, and Efficient Root Cause Analysis on Large-Scale Microservice Systems	Ruomeng Ding et al	2023	An end-to-end RCA approach that uses reinforcement learning to develop a pruning policy for service dependency graphs based on real-time traces and logs.	Surpasses leading RCA approaches on real data from Microsoft Exchange and is integrated into Microsoft M365 Exchange.	Reinforcement learning adds complexity; further exploration needed for adaptability to other

						microservices environments
5	MRCA: Metric-level Root Cause Analysis for Microservices via Multi-Modal Data	Yidan Wang et al	2024	Metric-level RCA using multi-modal data (traces, logs, metrics) to construct causal graphs with a reward mechanism for terminating unnecessary expansion	Achieves higher accuracy and efficiency than state-of-the-art approaches across two microservice benchmarks.	Effectiveness in real-time analysis and scalability in larger systems require further investigation
6	HeMiRCA: Fine-Grained Root Cause Analysis for Microservices with Heterogeneous Data Sources	Zhouruixing Zhu et al	2024	Hierarchical RCA using Spearman correlation and anomaly-aware monotonic correlation to rank suspicious metrics from heterogeneous data sources	Surpasses state-of-the-art methods in detecting root causes at both the service and metric levels.	Additional research is required to evaluate performance across various microservice environments and its integration with current monitoring tools.
7	Recommending Root-Cause and Mitigation Steps for Cloud Incidents using Large Language Models	Toufique Ahmed et al	2023	GPT-3.x models (Curie, Davinci); fine-tuning for RCA and mitigation; BLEU-4 evaluation metrics	Improved RCA and mitigation with fine-tuned models	Absence of microservices-specific exception classification and RCA optimization
8	SEEKER: Enhancing Exception Handling in Code with LLM-Based Multi-Agent Approach	Xuanming Zhang et al	2024	Multi-agent framework (Scanner, Detector, Predator, Ranker, Handler agents) with Retrieval-Augmented Generation (Deep-RAG) for exception handling	Improved exception handling and code robustness	Lack of scalability and optimization for microservices exception classification
9	Exploring LLM-based Agents for Root Cause Analysis	Devjeet Roy et al	2024	ReAct agent framework; retrieval tools for RCA; comparison with baseline models	Competitive RCA performance with high factual accuracy	Insufficient use of fine-tuned LLMs for RCA diagnostics involving exception patterns
10	CloudRCA: A Root Cause Analysis Framework for Cloud Computing Platforms	Yingying Zhang et al.	2021	Integrates multi-dimensional data (logs, metrics, traces) with machine learning techniques to correlate	Significantly reduces incident diagnosis time and improves system reliability	Enhanced scalability and real-time processing capabilities are needed for large-scale deployments

				anomalies and identify root causes in cloud computing platforms		
11	Graph-based Root Cause Analysis for Service-Oriented and Microservice Architectures	Álvaro Brandón et al.	2020	Constructs dependency graphs to model service interactions; applies network analysis techniques for fault propagation analysis and root cause identification	Enhanced accuracy and speed of RCA compared to conventional correlation-based methods	Challenges in scaling to very large, dynamic systems require further optimization
12	LogBERT: Log Anomaly Detection via BERT	Haixuan Guo et al.	2021	A transformer-based approach utilizing the BERT architecture with pre-training on large-scale log data followed by fine-tuning for anomaly detection.	Achieved improved precision and recall on benchmark log datasets compared to traditional log analysis methods	Scalability challenges and production integration require further research
13	How to Fight Production Incidents: An Empirical Study on a Large-scale Cloud Service	Supriyo Ghosh et al.	2022	Empirical analysis of production incidents; mitigation strategies (rollback, configuration updates)	Over 90% incidents mitigated without code changes	No integration of automated exception classification with RCA workflows
14	MABC: Multi-Agent Blockchain-inspired Collaboration for Root Cause Analysis	Wei Zhang et al.	2024	A multi-agent framework incorporating a blockchain-inspired voting mechanism for root cause analysis (RCA).	Improved RCA efficiency with reduced hallucination issues	Complexity in adapting multi-agent frameworks for microservices exception handling
15	Fine-Tuning LLMs for Enterprise: Practical Guidelines and Recommendations	Mathav Raj J et al.	2024	LORA, Retrieval-Augmented Generation (RAG); fine-tuning techniques for proprietary datasets	Identified best practices for fine-tuning and RAG in enterprise applications	Limited exploration of fine-tuning LLMs for exception classification in microservices
16	Automating Microservices Test Failure Analysis	Pawan Kumar Sarika et al	2023	ML algorithms (Random Forest, SVM, Gradient Boosting); log	Automated failure classification with high	Lack of NLP-based exception classification for production micro-

	using Kubernetes Cluster Logs			analysis on Kubernetes clusters	accuracy using Random Forest	services environments
17	MicroRec: Leveraging Large Language Models for Microservice Recommendation	Ahmed Saeed Alsayed et al.	2024	Dual-encoder architecture; contrastive learning; LLM-based semantic search using Stack Overflow and Dockerfile data	14× more accurate recommenda-tions than Docker Hub	Limited evaluation on microservice compatibility with RCA diagnostics
18	Analytics of Machine Learning-based Algorithms for Text Classification	Sayar Ul Hassan et al	2023	Comparison of ML algorithms (SVM, k-NN, Logistic Regression, Naive Bayes, Random Forest); evaluation using precision, recall, F1-score metrics	Identified optimal algorithms for dataset-specific text classification	Absence of ML models focused on classifying microservice-related exception logs
19	Anomaly Detection in Microservice Environments Using Distributed Tracing Data Analysis and NLP	Iman Kohyar nejadfard et al	2023	NLP techniques for sequence analysis; distributed tracing data; regression analysis for anomaly detection	Achieved high anomaly detection accuracy (F-score of 0.9759)	Limited focus on exception classification and detailed RCA for micro-services
20	A Survey on Automated Log Analysis for Reliability	Shilin He et al	2023	Log aggregation, pattern recognition, and event correlation techniques	Identified best practices and challenges for automated log analysis	Lack of efficient real-time log classification and RCA integration for micro-services
21	Code Llama: Open Foundation Models for Code	Baptiste Rozière et al	2023	Fine-tuned LLMs (7B, 13B, 70B); infilling training; instruction tuning; long-context optimization	State-of-the-art code generation and completion; superior benchmark performance on HumanEval and MBPP	Limited exploration of RCA diagnostics or exception classification in code-focused LLMs
22	Understanding the Issues, Their Causes, and Solutions in Microservices Systems: An Empirical Study	Muhammad Waseem et al.	2023	Mixed-methods approach; taxonomy development from issue tracking, interviews, and surveys	Developed taxonomies for issues, causes, and solutions	Need for automated RCA and exception classification solutions for large-scale microservices systems
23	Automated Root Causing of Cloud Incidents Using In-Context Learning with GPT-4	Xuchao Zhang et al	2024	In-context learning with GPT-4; historical incident analysis; semantic similarity evaluation	43.5% improvement in RCA correctness over fine-tuned models	No solution for real-time RCA diagnostics for microservices exceptions

4.2 Key insights from comparative study:

- **Automated RCA:** Limited solutions fully automate RCA for large-scale microservices exception analysis.
- **Exception Classification:** Lack of robust systems to distinguish business and runtime errors in microservices.
- **Real-time Diagnostics:** Existing solutions face scalability and latency challenges in high-velocity environments.
- **Scalability Gaps:** Multi-agent frameworks struggle with scalability in decentralized microservices architectures.
- **Centralized Aggregation:** Limited focus on technologies like Kafka for real-time exception aggregation.
- **Latency:** Many current methods struggle with scalability and low-latency diagnostics when deployed in large-scale systems.

METHODOLOGY USED

This project employs a phased approach to tackle the challenges of exception classification and root cause analysis (RCA) in microservices architectures. The methodology involves the following steps:

Data Preprocessing

- Exception stack trace data is collected from various microservices.
- Data comprises of 52% of Runtime Exceptions and 48% of Business Exceptions
- Exception stack traces and code repositories are also processed using NLP.
- Data is parsed, tokenized and embedded for each stack trace [20].
- Code from repositories is embedded with CodeBERT and embeddings are stored for future use in LLM.

Exception Aggregation

- A Kafka Cluster is utilized as the central platform for real-time aggregation of exceptions generated by various microservices.
- Services send exceptions to Kafka topics, which act as scalable and reliable queues for handling high volumes of data.
- The system ensures fault tolerance and enables seamless integration of distributed microservices.

Exception Classification

- A dedicated Exception Classifier Service is designed to use text classifier built using NLP.
- Preprocessed data is then fed to chosen classification models such as SVM, Naïve Baye's and XGBoost [20].
- Classification models are then compared against each other with metrices like Precision Recall and F1-Score.

Root Cause Analysis (RCA)

- RCA is performed by an LLM fine-tuned using exception traces and codebert embeddings of code repositories of microservices.
- CodeLlama and Mistral are chosen for fine tuning using LoRA for their ability to generate text on provided context [17], [22].
- Runtime exceptions stack traces categorized by the Exception Classifier are sent to the RCA Service via API integration.
- The LLM-based RCA system analyses exception received from kafka to identify underlying issues and provide actionable insights.

Performance Evaluation and Optimization

- Key performance metrics, including RCA accuracy, classification precision, recall, and latency are measured.
- Human qualitative evaluation is also done on various sample of exception traces.

ARCHITECTURE

The provided block diagram represents the architecture for the LLM-based Automated Root Cause Analysis (RCA) System designed for microservices architectures. It illustrates the interaction between services, exception handling mechanisms, and RCA components, as outlined below:

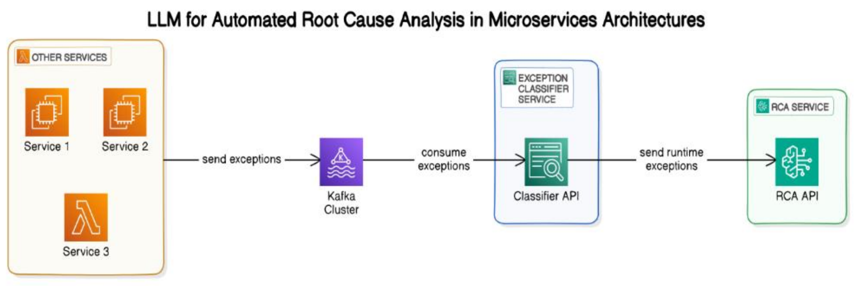

Fig 6.1 Architecture Diagram

Micro-Services (Service 1, Service 2, Service 3):

These represent various microservices within the architecture that generate errors and exceptions during runtime. These services are the primary producers of exception data, which could include business exceptions (expected errors due to business logic) and runtime exceptions (unanticipated system failures).

Kafka Cluster:

Acting as a central hub for exception aggregation, the Kafka Cluster receives exceptions from the microservices in real time. Kafka ensures that the system can handle high volumes of exceptions efficiently and reliably. The clustered design allows for scalability and fault tolerance, ensuring uninterrupted flow of data.

Exception Classifier Service:

This service consumes exception data from Kafka using its API. The Exception Classifier API processes the exceptions to classify them into two categories: Business Exceptions: Expected errors related to business logic, such as validation failures. Runtime Exceptions: Critical system errors, such as null pointer exceptions or database connectivity failures. This classification leverages Natural Language Processing (NLP) for extracting patterns and categorizing exceptions effectively [20].

RCA Service:

Once exceptions are classified, Runtime Exception stack traces are sent to the RCA Service for detailed analysis. The RCA Service uses Large Language Model (LLM) that has been LoRA fine-tuned to perform automated root cause analysis. This involves interpreting log messages from stack traces and CodeBERT embedding saved in data preparations step. RCA's produced by this service would aid addressing critical underlying issues in microservices.

RESULTS AND DISCUSSIONS:

The research started off with collection of exceptions stack traces data from various microservices. Data also included manually generated exceptions. There were roughly 52% of runtime exceptions and 48 % of business exceptions in data set. For preprocessing of data, various NLP techniques such as TF-IDF vectorization, tokenization were executed. Data was also embedded with CodeBERT for faster search. This search was needed while generating RCA using LLM. These embedding were saved and secured.

For exception classification, three models were chosen which usually works best with textual data. Models selected were SVM, Naïve Bayes and Gradient Boosting XGBoost. Mentioned models were fitted on train-test split of exception traces. Models were compared as per their performance metrices such as recall, precision and f1 score.

Table 7.1: Exception Classification Performance metrices

Model	**Runtime Exception Recall**	**Runtime Exception F1**	**Business Exception Precision**
SVM	72.40%	68.30%	88.20%
Naïve Bayes	85.10%	82.70%	91.50%
XGBoost	89.60%	87.40%	93.80%

Runtime exceptions are critical to detect and missing any would might lead to major issues in future. Runtime Exception recall indicates model's ability to recognize runtime exceptions. Similarly, F1 score indicated model's balance of recognizing runtime and business exceptions. XGBoost achieved the highest runtime recall (89.6%) and F1-score (87.4%) due to its ability to handle imbalanced data while modeling non-linear relationships.

RCA is done using fine-tuned LLM. CodeLlama and Mistral were first choice because of their ability for code generation. Preprocessed data was fed to models with specific number of epochs. There were many hyperparameters considered, following are some examples of parameter values.

Table 7.2: LLM Hyperparameters

Hyperparameter	**CodeLlama-7B**	**Mistral-7B**
Base Model	codellama/CodeLlama-7b-hf	mistralai/Mistral-7B-v0.1
LoRA Rank	64	32
LoRA Alpha	128	64
Learning Rate	3e-5	2e-5
Quantization	4-bit (GPTQ)	4-bit (NF4)
Loss Function	Weighted Cross-Entropy	Focal Loss

Models were fine-tuned with and without code snippets. CodeBERT embeddings were used in training to understand impact of code context on RCA accuracies. Following are the results with and without code snippets. Models were also evaluated with human qualitative judgement. Exceptions were caused with human intervention and expected RCA was validated against RCA generated by models.

Table 7.3: LLM Performance Metrices

Models	**With Code Context**	**Without Code Context**	**Human Qualitative Evaluation**
Mistral-7B	87.60%	68.20%	83%
CodeLlama-7B	91.50%	69.30%	91%

Here, CodeLlama has outperformed Mistral-7B because of its code-aware architecture, larger context window, and domain-specific pretraining. Fine-tuned CodeLlama is integrated with rest API to serve in microservices architecture.

Finally, kafka is used to bind all components together. Kafka acts as an exception aggregator accepting exceptions from multiple microservices. These exceptions are written to a topic which is then further read by exception classifier. Stack traces of runtime exceptions are then forwarded to RCA service for generating RCA.

CONCLUSION AND FUTURE SCOPE

This research showcased how Large Language Models (LLMs) can be effectively utilized to automate Root Cause Analysis (RCA) for microservices, with a focus on runtime exception classification and code-aware root cause identification. By comparing performances of classification models (XGBoost, SVM, Naïve Bayes) and fine-tuned LLMs (CodeLlama-7B, Mistral-7B) we observed that, XGBoost performed best among other classification models selected. CodeLlama-7B has significant provided precise and accurate RCAs for given stack traces. CodeLlama-7B has performed best when was given preprocessed code snippets while generating reports. The integration of exception aggregation, exception classification, code repository context and optimized hyperparameters for LLM has improved ability to find critical issues early and address them before any outage in microservices architecture.

This research can be further improved by aggregative environment logs such as resource consumption, disk/network usage etc. LLM models can be further distilled to achieve leaner and smaller models for cost effective deployments. In microservice architecture, services keep getting updated, new services gets added. A feedback loops can be added to tune the model at runtime. Currently, model is capable of performing RCA upon given code context and exception stack trace. This can be further extended to act upon the RCA and remediate the root cause and fix the issue.

ACKNOWLEDGEMENTS

We would like to express our sincere gratitude to all those who have supported and guided us throughout the completion of our project titled " LLM for Automated Root Cause Analysis in Microservices Architectures."

Firstly, we would like to thank our guide, **Dr. Amruta Mhatre**, for his invaluable guidance, encouragement, and unwavering support. His insights and expertise were instrumental in helping us conceptualize and execute this project successfully.
Our sincere appreciation goes to the **PG Head Dr. Manish Rana** and **Principal Dr. Kamal Shah** of **St. John College of Engineering & Management (SJCEM)**, Palghar, Mumbai, India, for their continuous encouragement, vision, and leadership. Their guidance provided us with the necessary resources and motivation to complete this project with great enthusiasm.
We would also like to acknowledge all the faculty members and staff at SJCEM for their support, and our peers for their valuable suggestions during the course of this research.
Lastly, we would like to thank our families for their unconditional love and support, which helped us stay focused and motivated throughout the project.
This research paper on project would not have been possible without the collective efforts of everyone mentioned above.

REFERENCES:

[1] T. Wang and G. Qi, "A Comprehensive Survey on Root Cause Analysis in (Micro) Services: Methodologies, Challenges, and Trends," 2024, *arXiv*. doi: 10.48550/ARXIV.2408.00803.

[2] M. Waseem *et al.*, "Understanding the Issues, Their Causes and Solutions in Microservices Systems: An Empirical Study," 2023, *arXiv*. doi: 10.48550/ARXIV.2302.01894.

[3] Y. Wang, Z. Zhu, Q. Fu, Y. Ma, and P. He, "MRCA: Metric-level Root Cause Analysis for Microservices via Multi-Modal Data," in *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, Sacramento CA USA: ACM, Oct. 2024, pp. 1057–1068. doi: 10.1145/3691620.3695485.

[4] L. Pham, H. Ha, and H. Zhang, "BARO: Robust Root Cause Analysis for Microservices via Multivariate Bayesian Online Change Point Detection," *Proc. ACM Softw. Eng.*, vol. 1, no. FSE, pp. 2214–2237, Jul. 2024, doi: 10.1145/3660805.

[5] T. Ahmed, S. Ghosh, C. Bansal, T. Zimmermann, X. Zhang, and S. Rajmohan, "Recommending Root-Cause and Mitigation Steps for Cloud Incidents using Large Language Models," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, Melbourne, Australia: IEEE, May 2023, pp. 1737–1749. doi: 10.1109/ICSE48619.2023.00149.

[6] S. He, P. He, Z. Chen, T. Yang, Y. Su, and M. R. Lyu, "A Survey on Automated Log Analysis for Reliability Engineering," *ACM Comput. Surv.*, vol. 54, no. 6, pp. 1–37, Jul. 2022, doi: 10.1145/3460345.

[7] C.-M. Lin, C. Chang, W.-Y. Wang, K.-D. Wang, and W.-C. Peng, "Root Cause Analysis In Microservice Using Neural Granger Causal Discovery," 2024, *arXiv*. doi: 10.48550/ARXIV.2402.01140.

[8] R. Ding *et al.*, "TraceDiag: Adaptive, Interpretable, and Efficient Root Cause Analysis on Large-Scale Microservice Systems," in *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, San Francisco CA USA: ACM, Nov. 2023, pp. 1762–1773. doi: 10.1145/3611643.3613864.

[9] Z. Zhu, C. Lee, X. Tang, and P. He, "HeMiRCA: Fine-Grained Root Cause Analysis for Microservices with Heterogeneous Data Sources," *ACM Trans. Softw. Eng. Methodol.*, vol. 33, no. 8, pp. 1–25, Nov. 2024, doi: 10.1145/3674726.

[10] X. Zhang, Y. Chen, Y. Yuan, and M. Huang, "Seeker: Enhancing Exception Handling in Code with LLM-based Multi-Agent Approach," 2024, *arXiv*. doi: 10.48550/ARXIV.2410.06949.

[11] D. Roy *et al.*, "Exploring LLM-Based Agents for Root Cause Analysis," in *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, Porto de Galinhas Brazil: ACM, Jul. 2024, pp. 208–219. doi: 10.1145/3663529.3663841.

[12] Y. Zhang *et al.*, "CloudRCA: A Root Cause Analysis Framework for Cloud Computing Platforms," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, Virtual Event Queensland Australia: ACM, Oct. 2021, pp. 4373–4382. doi: 10.1145/3459637.3481903.

[13] Á. Brandón, M. Solé, A. Huélamo, D. Solans, M. S. Pérez, and V. Muntés-Mulero, “Graph-based root cause analysis for service-oriented and microservice architectures,” *J. Syst. Softw.*, vol. 159, p. 110432, Jan. 2020, doi: 10.1016/j.jss.2019.110432.

[14] H. Guo, S. Yuan, and X. Wu, “LogBERT: Log Anomaly Detection via BERT,” in *2021 International Joint Conference on Neural Networks (IJCNN)*, Shenzhen, China: IEEE, Jul. 2021, pp. 1–8. doi: 10.1109/IJCNN52387.2021.9534113.

[15] S. Ghosh, M. Shetty, C. Bansal, and S. Nath, “How to fight production incidents?: an empirical study on a large-scale cloud service,” in *Proceedings of the 13th Symposium on Cloud Computing*, San Francisco California: ACM, Nov. 2022, pp. 126–141. doi: 10.1145/3542929.3563482.

[16] W. Zhang *et al.*, “mABC: multi-Agent Blockchain-Inspired Collaboration for root cause analysis in micro-services architecture,” 2024, *arXiv*. doi: 10.48550/ARXIV.2404.12135.

[17] M. R. J, K. VM, H. Warrier, and Y. Gupta, “Fine Tuning LLM for Enterprise: Practical Guidelines and Recommendations,” 2024, *arXiv*. doi: 10.48550/ARXIV.2404.10779.

[18] P. K. Sarika, D. Badampudi, S. P. Josyula, and M. Usman, “Automating Microservices Test Failure Analysis using Kubernetes Cluster Logs,” in *Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering*, Oulu Finland: ACM, Jun. 2023, pp. 192–195. doi: 10.1145/3593434.3593472.

[19] A. S. Alsayed, H. K. Dam, and C. Nguyen, “MicroRec: Leveraging Large Language Models for Microservice Recommendation,” in *Proceedings of the 21st International Conference on Mining Software Repositories*, Lisbon Portugal: ACM, Apr. 2024, pp. 419–430. doi: 10.1145/3643991.3644916.

[20] S. U. Hassan, J. Ahamed, and K. Ahmad, “Analytics of machine learning-based algorithms for text classification,” *Sustain. Oper. Comput.*, vol. 3, pp. 238–248, 2022, doi: 10.1016/j.susoc.2022.03.001.

[21] I. Kohyarnejadfard, D. Aloise, S. V. Azhari, and M. R. Dagenais, “Anomaly detection in microservice environments using distributed tracing data analysis and NLP,” *J. Cloud Comput.*, vol. 11, no. 1, p. 25, Aug. 2022, doi: 10.1186/s13677-022-00296-4.

[22] B. Rozière *et al.*, “Code Llama: Open Foundation Models for Code,” 2023, *arXiv*. doi: 10.48550/ARXIV.2308.12950.

[23] X. Zhang *et al.*, “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*, Porto de Galinhas Brazil: ACM, Jul. 2024, pp. 266–277. doi: 10.1145/3663529.3663846.

Notes on Contributors

Dr. Amruta Mhatre

Mr. Afrid Shaikh

MTech Scholar in Computer Engineering Department, St. John College of Engineering and Management

Qualification Detail: B.E (Information Technology, Palghar, Mumbai University, Maharashtra

Work Experience (Teaching / Industry): 6.6 years of industry experience

Area of specialization: Computer

ORCID

1] Dr. Amruta Mhatre

2] Mr. Afrid Shaikh https://orcid.org/0009-0004-7223-5607