Aiding Software Root Cause Analysis with Large Language Models

-Evaluation of the Effectiveness of Fine-tuned T5, GPT, and RAG in the handling Customer Fault Reports

Shijun Feng

Applied Data Science
Two-year master
Master Thesis in Applied Data Science, 30 Credits
Academic Year: 2023-2025
Period: Spring 2025
Supervisor: Fabian Lorig

Declaration of AI Usage

This thesis benefited from the use of general-purpose artificial intelligence (AI) tools, including **ChatGPT (OpenAI)** and **Grammarly**, to support specific aspects of the research and writing process. These tools were used solely in supportive roles and did not generate original content, perform analysis, or draw conclusions.

ChatGPT was used for the following purposes:

- *Concept clarification*: Assisting with understanding complex technical topics, such as machine learning models and root cause analysis, by providing accessible explanations and contextual examples.
- *Experimental exploration*: Supporting the development and comparison of model evaluation strategies, particularly for architectures such as T5, GPT-2, and RAG.
- *Citation verification*: Cross-checking references for relevance, factual alignment with cited claims, and correct formatting according to academic standards.
- *Language refinement*: Suggesting improvements to sentence structure, clarity, coherence, and academic tone throughout the thesis.

Grammarly was used for:

- *Proof reading and grammar checking*: Detecting and correcting grammar, punctuation, and typographical errors.
- *Writing style enhancement*: Offering suggestions for clearer phrasing, improved conciseness, and appropriate academic tone.

These AI tools contributed to improving the overall *readability*, *technical precision*, and *linguistic consistency* of the final document. All intellectual contributions, implementation decisions, and analytical conclusions remain the sole responsibility of the author.

Science Summary

In the telecommunications industry, service providers like Ericsson, Nokia, and Huawei offer 5G solutions to mobile network operators such as Telia, Vodafone, and China Mobile. When these operators use the 5G systems—either in testing or in live networks—they often encounter issues, known as faults, which they report back to the service provider. Because a service provider supports many operators, it receives a large number of fault reports, some of which may be similar or even identical. Efficiently managing and diagnosing these faults is critical—each operator serves millions of end-users, so even a small issue can lead to widespread customer escalations, service disruptions, and serious damage to the operator's brand.

Engineers at the service provider currently handle these faults manually: they classify each fault, determine which design team should investigate it, reroute the issue if necessary, and work through to identify the root cause. When multiple operators report similar or the same faults, this creates duplicate workload and significantly increases the burden on engineers. The entire process remains time-consuming and complex due to the scale and technical depth of 5G systems.

To improve this, I fine-tuned large language models (LLMs), including T5, GPT-2, and a Retrieval-Augmented Generation (RAG) model, using a dataset of historical fault reports that had already been resolved. These models are then used to automatically generate likely root cause analyses for new faults.

Among the models tested, the Retrieval-Augmented Generation (RAG) model performed the best, likely due to its ability to leverage specific, relevant historical cases during root cause generation. This AI-powered framework can assist first-line customer support in guiding operators toward fault resolution or help engineers quickly identify the root cause, making the entire troubleshooting process faster, more consistent, and more scalable.

Aiding Software Root Cause Analysis with Large Language Models

- Evaluation of the Effectiveness of Fine-tuned T5, GPT, and RAG in the handling Customer Fault Reports

SHIJUN FENG, Malmö University, Sweden

Abstract

Software systems generate a substantial number of fault reports during pre-deployment customer testing, making manual root cause analysis (RCA) both time-consuming and error-prone. This study explores the use of large language models (LLMs)—specifically T5, GPT-2, and a retrieval-augmented generation (RAG) model—to automate and enhance the RCA process in a domain-specific software engineering setting. Using a curated dataset of real-world fault descriptions and resolutions, the models were fine-tuned and evaluated using BLEU-4, ROUGE, and BERT-based semantic similarity metrics. Results indicate that T5 outperforms GPT-2 in lexical and structural fidelity (e.g., BLEU-4: 0.1810 vs. 0.1210), while RAG achieves the highest semantic similarity (BERT score: 0.7715). These findings suggest that combining T5's precision in technical phrasing with RAG's contextual understanding may offer a promising direction for developing intelligent RCA assistance tools that improve both accuracy and relevance in software fault diagnosis. Future work will focus on hybrid model optimization and user-centered system integration for real-world engineering workflows.

Additional Key Words and Phrases: software development, fault reports, root cause analysis (RCA), Large-Language Model (LLM), hybrid dataset, supervised data, unsupervised data, prototype, decision support, scalability

Author's Contact Information: Shijun Feng, shijun.feng@ericsson.com, Malmö University, Malmö, Skåne, Sweden.

Contents

1 INTRODUCTION

Background

As software companies deploy their solutions into operators' networks, various issues inevitably arise during testing and operational use. Operators routinely report faults, many of which exhibit similar symptoms across different network environments. Efficiently processing, categorizing, and addressing these trouble reports is crucial to maintaining software reliability and customer satisfaction. However, the current handling of fault reports remains largely manual, requiring developers to invest substantial time and effort in reviewing, classifying, and resolving reported issues.

A particularly time-consuming challenge is the management of duplicate fault reports. Even when multiple operators report identical or highly similar issues due to cloud native environment complexity, Diego et al.[15] focuses on Anomaly Detection and Root Cause Analysis in Cloud native environments using LLM and Bayesian networks, still, developers must manually verify each case, determine whether it corresponds to an existing problem, link it to previously resolved faults, or classify it as a novel issue requiring investigation. This repetitive, experience-driven process not only slows down response times but also increases the risk of inconsistent handling, leading to operational inefficiencies. Furthermore, the lack of automation in identifying and clustering similar reports often results in redundant work, extending resolution times unnecessarily.

The impact of this inefficiency extends beyond internal development workflows. Mishandling or delayed processing of fault reports can lead to software performance degradation, operational instability, and escalations from dissatisfied customers. In cases where critical faults remain unresolved due to oversight, operators may experience network disruptions, directly affecting service quality. These challenges highlight the urgent need for a more structured and intelligent approach to fault report analysis.

Currently, despite the structured process in place, developers addressing new fault reports largely rely on their individual experience and domain expertise. This reliance introduces variability in resolution times and effectiveness, particularly when dealing with complex software faults. The absence of a systematic, data-driven approach to RCA further exacerbates the problem, as developers must repeatedly analyze logs Bahareh Afshinpour et

al.[1], correlate system behaviors, and manually infer root causes—a process that is both labor-intensive and prone to human error.
Given these limitations, there is significant potential for leveraging AI-driven solutions, particularly large language models (LLMs), to enhance the efficiency of fault report analysis. By automating the categorization, clustering, and RCA processes, LLMs can help reduce lead times, improve diagnostic accuracy, and optimize resource allocation within software development teams.

Purpose

Recent advancements in large language models (LLMs) have demonstrated their potential to automate root cause analysis (RCA) in software systems. Traditional RCA methods often relied on extensive manual effort, domain expertise, and time-consuming log analysis, which delayed fault resolution and increased operational costs Prior studies (e.g., Yinfang Chen et al. [5]) have shown that integrating LLMs into RCA tasks can improve diagnostic accuracy, reduce response times, and provide actionable insights for engineers. Similarly, Yongqi Han et al. [8] highlighted the potential of LLMs for analyzing historical reports, detecting similarities, and generating structured explanations.
Motivated by these insights, this study aims to move beyond applying LLMs in isolation and instead investigate their systematic fine-tuning, comparative performance, and suitability for RCA tasks. Specifically, the purpose of this research is to evaluate LLMs using automated metrics, compare generative and retrieval-augmented approaches, and propose initial framework components to guide model selection or hybridization in RCA workflows.

Objective

The objectives of this study are directly aligned with the three research questions:

- To develop and fine-tune LLM-based models (e.g., T5, GPT-2) on domain-specific fault report data for generating RCA explanations, and to evaluate their outputs using automated metrics such as BLEU, ROUGE, and BERTScore.

- To compare the performance of fine-tuned generative models and retrieval-augmented generation (RAG) approaches in RCA tasks, focusing on output quality, interpretability, and alignment with engineering workflows, as measured by automated metrics.
- To propose an initial set of evaluation criteria and framework components based on metric-driven analysis, providing guidance for selecting or hybridizing LLM architectures in RCA tasks.

By pursuing these objectives, the research establishes a systematic, metrics-driven foundation for applying LLMs to RCA in software fault reports, while laying the groundwork for future extensions such as multimodal inputs and human-in-the-loop evaluation.

Scope and Delimitations

The scope of this thesis is deliberately restricted to textual RCA data and automated evaluation metrics, reflecting feasibility and resource constraints. Although fault reports may contain multimodal information such as charts, logs, or metrics, this study considers only textual inputs (Summary, Description, Importance Code, and Answer fields). This limitation ensures a focused investigation of language-based RCA capabilities, while acknowledging that multimodal integration represents an important direction for future work.

Model selection is similarly constrained. The study evaluates two generative architectures (T5, GPT-2) and one retrieval-augmented model (RAG). Larger-scale variants (e.g., T5-Large) and advanced retrieval pipelines are recognized as promising, but were excluded to maintain tractability within the scope of a master's thesis.

Finally, evaluation is limited to automated metrics (BLEU, ROUGE, and BERTScore). While human-in-the-loop assessments would provide richer insights into interpretability and usability, they fall outside the practical scope of this study. The focus here is on establishing a replicable, metric-driven baseline that future research can build upon.

By narrowing the scope in this way, the thesis ensures that the proposed methodology and framework can be implemented rigorously within the available time and resources, while providing a foundation for more comprehensive investigations.

2 RESEARCH QUESTIONS

To address the challenges outlined in the introduction, this study aims to answer the following research questions:

- RQ1 – How can large language models (LLMs) be fine-tuned on domain-specific fault report data to generate RCA explanations, and how can their outputs be evaluated using automated metrics such as BLEU, ROUGE, and BERTScore?
- RQ2 – How do fine-tuned generative models (e.g., T5, GPT-2) and retrieval-augmented models (RAG) compare in RCA tasks when evaluated with automated metrics in terms of output quality, interpretability, and alignment with engineering workflows?
- RQ3 – What initial set of evaluation criteria and framework components can be proposed, based on automated metric results, to guide the selection or hybridization of models for RCA tasks?

This study evaluated LLM-generated RCA explanations exclusively through automated metrics, namely BLEU, ROUGE, and BERTScore. Human evaluation of explanation quality or usefulness was not conducted in this phase and is identified as an important direction for future work..

3 LITERATURE REVIEW AND RELATED WORK

3.1 Literature Review

Root Cause Analysis in Software System

Yongqi Han et al.[8] present LasRCA, a framework for one-shot failure root cause analysis (RCA) in cloud-native systems, leveraging the collaboration of large language models (LLMs) and small classifiers. LasRCA trains a small classifier on limited fault examples, iteratively refines it using LLM-driven fault labeling, and performs joint RCA during inference. Experiments on public datasets demonstrate its effectiveness in identifying faults with minimal labeled data, outperforming traditional methods. The approach balances cost and performance, reducing reliance on extensive fault labels and manual intervention.

The paper highlights the potential of combining LLMs and small classifiers for efficient RCA in complex, heterogeneous systems.

Zhang, Xuchao et al.[28] propose an in-context learning approach for automated root cause analysis (RCA) of cloud incidents using GPT-4, eliminating the need for costly fine-tuning. By leveraging historical incidents as in-context examples, the method outperforms fine-tuned models like GPT-3 by 24.8% on average across metrics, with a 49.7% improvement over zero-shot models. Human evaluations show a 43.5% improvement in correctness and 8.7% in readability. The approach is validated on over 100,000 production incidents, demonstrating its effectiveness in reducing manual effort and computational costs while enhancing RCA accuracy and reliability in cloud environments.

Azam Ikram et al [10] present Root Cause Discovery (RCD), a scalable algorithm for identifying root causes of failures in microservice-based cloud applications. RCD treats failures as interventions on the failing node, leveraging both observational and interventional data to detect root causes without learning the full causal graph. It uses a hierarchical and localized approach to reduce computational complexity, focusing only on relevant parts of the graph. Evaluations on synthetic and real-world datasets, including the Sock-shop application, show RCD outperforms existing methods in accuracy and runtime. The approach is application-agnostic, requiring no domain knowledge, and significantly reduces the time to diagnose failures in large-scale systems.

Sarda, Komal and Namrud et al.[19] demonstrate a novel approach to automating root cause analysis (RCA) for cloud incidents using large language models (LLMs). Traditional RCA methods are labor-intensive and often lack accuracy. The proposed framework integrates multimodal data streams—Logs, Metrics, Traces, and Alerts (LMTA)—to enhance diagnostic effectiveness. By leveraging LLMs, the system predicts root cause categories with high accuracy, achieving up to 97% in evaluations using open-source datasets. The study addresses two key research questions: the effectiveness of the LMTA-based framework and the impact of excluding alerts on LLM performance. Results show that including alerts significantly improves RCA accuracy. The paper also discusses practical considerations, potential threats to validity, and related work in the field. Future research aims to explore LLMs for providing comprehensive insights and mitigation strategies for compromised servers.

Chen, Yinfang and Xi et al.[5] introduce RCACopilot, an automated system for root cause analysis (RCA) of cloud incidents using large language models (LLMs). RCACopilot integrates multi-source data (logs, metrics, traces) through predefined incident handlers, enabling efficient diagnostic information collection. The LLM component predicts root cause categories and provides explanations, achieving a Micro-F1 score of 0.766. The system has been deployed at Microsoft for over four years, significantly reducing manual effort for on-call engineers. RCACopilot demonstrates the potential of LLMs in enhancing RCA by automating data collection, summarization, and reasoning, improving incident resolution efficiency and accuracy in large-scale cloud environments.

Wang, Lu and Zhang, Chaoyun et al.[22] not from LLM but from Reinforcement Learning show a novel framework for root cause analysis (RCA) in microservice systems using Hierarchical Reinforcement Learning from Human Feedback (HRLHF). The framework addresses the challenges of identifying root causes in dynamic, large-scale systems like Microsoft Exchange by leveraging human expertise to reduce uncertainty in constructing service dependency graphs. HRLHF minimizes human feedback complexity from $O(N^2)$ to $O(1)$ and extends dependency graphs to capture temporal dynamics, improving RCA accuracy. The framework was evaluated on synthetic and real-world datasets, demonstrating superior performance over state-of-the-art methods. It has been integrated into Microsoft M365 Exchange, enhancing system reliability by providing precise root cause localization and actionable insights for engineers. The approach combines causal discovery, reinforcement learning, and human feedback to optimize RCA in complex microservice environments.

Roy et al. [18] address the challenge of automating root cause analysis (RCA) in cloud incident management, a labor-intensive task for on-call engineers. They explore the use of Large Language Model (LLM) based agents to dynamically collect diagnostic information, overcoming the limitations of prior LLM-based RCA approaches. The authors present an empirical evaluation of a REACT agent on a production incident dataset from Microsoft. The results indicate that REACT performs competitively with strong baselines, with increased factual accuracy. A case study demonstrates how agents can interface with diagnostic services for manual RCA. The study highlights the potential and challenges of implementing LLM agents for RCA in real-world scenarios.

Anandayuvaraj et al.[2] introduce FAIL, an LLM-based system to collect, analyze, and summarize software failures from news reports, addressing the limitations of manual analysis. FAIL uses LLMs to group articles describing the same incidents and analyze them using taxonomies. Evaluated on 31 manually analyzed software failures, FAIL achieved high F1 scores (90%) for collecting news, V-measure (0.98) for merging articles, and fact extraction (90%).

Zefan Wang et al.[24] present RCAgent, a tool-augmented LLM autonomous agent framework for practical and privacy-aware industrial RCA usage, that, running on an internally deployed model rather than GPT families, is capable of free-form data collection and comprehensive analysis with tools, yielding evident and consistent superiority over ReAct across all aspects of RCA-predicting root causes, solutions, evidence, and responsibilities, as validated by both automated metrics and human evaluations.

Zhou, Chunying et al. [30]propose a novel approach that leverages multiple views—such as report-code interactions, report-report similarities, and code-code co-citations—to improve fault localization. By employing graph neural networks and contrastive learning, the method effectively captures intricate relationships, leading to enhanced localization performance.

This research "Exploring the extent of similarities in software failures across industries using LLMs" [6] by Detloff, Martin, uses the Failure Analysis Investigation with LLMs (FAIL) model to extract industry-specific information from news articles about software failures. By categorizing articles into specific domains and failure types, it identifies patterns and common vulnerabilities across industries. The findings, visualized through graphs, offer valuable insights for software engineers to proactively address potential risks and enhance software reliability.

Hu et al. [9] evaluate the effectiveness of fault detection methods for LLMs, proposing MuCS, a mutation-based confidence smoothing framework, which significantly enhances fault detection by up to 70.53%. Their study reveals that LLMs are often overconfident, and simple methods like Margin perform best.

Xu et al.[25] introduce OpenRCA, a benchmark for assessing LLMs' RCA capabilities in real-world scenarios, demonstrating that current models struggle with complex tasks, achieving only 11.34% accuracy even with advanced multi-agent systems like RCA-agent. Both studies emphasize the need for improved methods to handle diverse data types and

complex reasoning, underscoring the potential of LLMs in enhancing software reliability and maintenance.

Conclusion of Literature Review

The literature reveals a clear trajectory in the evolution of root cause analysis (RCA) methods, from traditional rule-based and machine learning techniques to advanced, language-oriented frameworks powered by large language models (LLMs). Early approaches to RCA predominantly relied on expert-defined rules, static pattern matching, and machine learning models trained on structured log and metric data. While effective in constrained environments, these methods demonstrated limited adaptability to unstructured, linguistic data—such as customer fault reports—and often required significant manual intervention and domain expertise.

Deep learning techniques, such as artificial neural networks (ANNs) and convolutional neural networks (CNNs), provided improved capacity to model complex patterns in structured inputs. However, they lacked the ability to interpret natural language or generate human-readable explanations, thereby limiting their utility in contexts where descriptive fault narratives and contextual reasoning are critical.

In contrast, recent advancements in LLMs—such as T5, GPT-2, GPT-4, and retrieval-augmented models—have introduced new possibilities for RCA. These models excel at understanding and generating natural language, enabling more accurate, context-aware, and readable RCA outputs. Several studies have demonstrated the effectiveness of LLMs in automating key RCA tasks, including fault classification, explanation generation, and multi-source data synthesis. Moreover, emerging architectures combine LLMs with small classifiers, reinforcement learning, or tool-augmented agents to balance performance with resource efficiency and domain specificity.

Despite these advancements, recent benchmarks highlight key challenges that remain unresolved. These include the need for improved reasoning capabilities, mitigation of overconfidence in model predictions, and better integration of domain knowledge. Additionally, issues of model evaluation, interpretability, and engineering usability remain critical for the practical deployment of LLM-based RCA systems.

In summary, the current state of the literature demonstrates that LLMs hold significant promise for transforming RCA workflows in software engineering. However, there is a

pressing need for systematic evaluation of model effectiveness, domain alignment, and real-world usability—gaps that this study aims to address through comparative analysis and targeted fine-tuning of domain-adapted generative and retrieval-augmented models.

3.2 Related Work

Traditional ML Approaches in RCA

Root cause analysis (RCA) using machine learning has gained significant traction across domains such as software engineering, manufacturing, and networking. Traditional approaches often apply supervised models like support vector machines (SVMs), decision trees, or Naïve Bayes classifiers to classify faults based on historical data.

Wang et al. [23] train SVMs on textual bug reports to predict fault types (e.g., concurrency, memory, semantic bugs), achieving solid accuracy but providing no interpretability or semantic insight to support engineers during resolution. Chen et al. [3] combine PCA and SVM for fault detection in manufacturing, successfully identifying key structured features but excluding natural language inputs. Similarly, Zhang et al. [27] apply ML to SDN faults using time-series data, offering fast classification but limited diagnostic support.

Among interpretable models, Jovic et al. [11] use Naïve Bayes to identify predictive terms in bug descriptions, offering limited lexical guidance to engineers. However, none of these works integrate semantic retrieval or lexical pattern analysis to directly aid fault resolution.

Neural Network and Deep Learning Approaches in RCA

Deep learning has been increasingly adopted for root cause analysis (RCA), particularly in domains like manufacturing, network management, and software systems. Unlike traditional machine learning approaches, deep neural networks offer powerful feature learning capabilities, but not all methods provide semantic insight or lexical interpretability needed to support engineers in resolving issues.

Artificial neural networks (ANNs) and residual causal models like Sparse Causal Residual Neural Networks (SCRNN) have been applied to structured sensor data in manufacturing and quality control tasks [4]. These models perform well in predicting fault contributors

from numerical logs but do not operate on natural language input or offer semantic interpretability.

Other approaches leverage deep learning architectures such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), autoencoders, and transformers for unsupervised anomaly detection in system logs [20]. While these methods are useful for identifying outliers, they typically lack causal reasoning and are not designed to interpret textual summaries or aid fault resolution.

More promising results are seen in graph-based models. KGroot [29], a recent framework combining knowledge graphs with graph convolutional networks (GCNs), constructs semantic fault graphs from event traces and textual logs. It enables similarity-based retrieval of past faults with known causes, providing contextual and semantic guidance. This makes it particularly well-suited for cases where fault descriptions, priorities, and summaries are available.

Another technique involves combining LSTM networks with Bayesian neural networks (BNNs) to model probabilistic causality across time-series data [12]. Though effective in capturing causal structures, it relies entirely on structured logs and does not process natural language input.

Summary of Related Word

Traditional machine learning and deep learning methods have laid the foundation for automated root cause analysis, especially in classification and clustering tasks. However, these approaches often fall short in dealing with unstructured, domain-specific text where lexical and semantic nuance is crucial. Recent work using graph-based or retrieval-augmented models has improved case similarity detection, yet lacks generalization across diverse linguistic expressions. This opens the door for large language models (LLMs), which inherently support contextual understanding, semantic retrieval, and natural language generation. As such, this thesis explores whether LLMs—specifically T5, GPT-2, and RAG—can serve not as replacements but as intelligent assistants to engineers performing RCA, addressing both relevance and clarity in fault resolution.

3.3 Research Gap

Root cause analysis (RCA) in software engineering has been explored using both traditional and modern AI techniques. Classical machine learning methods such as support vector machines (SVMs), decision trees, and Naïve Bayes classifiers have achieved reasonable accuracy in fault classification using structured features or simple tokenized text [23, 11]. However, these models treat faults as fixed labels, lacking the ability to capture deeper semantic relationships between cases. Their interpretability is often limited to feature weight analysis, which provides minimal guidance in complex engineering contexts.

Deep learning approaches, including artificial neural networks (ANNs), convolutional neural networks (CNNs), and long short-term memory (LSTM) models, have extended RCA to time-series and log-based data [4, 20, 12]. While effective for structured input, they are rarely applied to natural language descriptions of faults. Even when accurate, their outputs often lack linguistically coherent explanations that engineers can directly use.

Recent advances such as graph-based neural networks (e.g., KGroot [29]) integrate structured event knowledge with semantic embeddings to retrieve related fault cases. While these systems support analogical reasoning, they rely heavily on predefined ontologies and rigid fault schemas, limiting adaptability to unstructured and linguistically diverse fault descriptions.

From this review, three persistent gaps emerge:

- **Limited exploitation of unstructured natural language** – Current RCA systems underutilize the semantic and contextual richness of fault descriptions.
- **Insufficient interpretability for engineering workflows** – Model outputs often lack the lexical precision and contextual relevance required for practical RCA decision-making.
- **Absence of systematic selection frameworks for LLM-based RCA** – No established methodology exists for comparing, selecting, or hybridizing LLMs in RCA tasks based on both performance and engineering usability.

Advances in large language models (LLMs) such as T5, GPT-2, and retrieval-augmented generation (RAG) present opportunities to address these issues. LLMs can capture nuanced meaning, match new reports to relevant historical cases, and generate coherent,

context-aware explanations. However, no existing study has systematically fine-tuned and compared multiple LLM architectures for RCA while also proposing a practical evaluation framework for model selection or hybridization.
This research addresses these gaps by:

- Fine-tuning domain-specific LLMs to generate accurate, context-relevant RCA explanations (RQ1).
- Comparing generative (T5, GPT-2) and retrieval-augmented (RAG) models for RCA in terms of quality, interpretability, and relevance (RQ2).
- Proposing an initial RCA Model Selection Framework to guide model choice or hybridization based on multiple performance and usability dimensions (RQ3).

4 METHODOLOGY

This chapter presents the methodological foundations of the study and the specific methods employed in its execution. It is structured into two main parts: the research methodology, which outlines the overarching strategy and philosophical orientation of the research, and the research methods, which describe the concrete steps, tools, and processes implemented.

4.1 Research Methodology

The study adopts a **quantitative**, **experimental**, and **comparative** methodology situated within a **positivist research paradigm**. This paradigm was chosen because the objective is to evaluate and compare the performance of large language models (LLMs) in supporting Root Cause Analysis (RCA) through measurable, replicable, and objective indicators.
A **quantitative approach** was deemed appropriate, as it enables the systematic evaluation of model outputs using standardized metrics such as BLEU, ROUGE, and BERTScore. This ensures transparency, reproducibility, and comparability across models.
The study employs an experimental design in which different LLMs are fine-tuned and assessed under controlled conditions using the same dataset and procedures. This design ensures that any observed differences in performance can be attributed to the models rather than extraneous factors.
The research is also inherently **comparative**, as it systematically contrasts the performance of three different architectures (T5, GPT-2, and RAG). The findings are used to

derive trade-offs between lexical fidelity, semantic similarity, and domain-specific alignment, ultimately informing the selection or hybridization of models.
While the overall research workflow can be summarized as a sequence of steps—data preparation, model development, evaluation, and synthesis—the detailed methodology in this chapter is organized around practical components. These include:

- **Tools and Platforms (Section 4.2)**
- **Dataset and Preprocessing (Sections 4.3–4.5)**
- **Feature Selection (Section 4.6)**
- **Model Selection and Training (Sections 4.7–4.8)**
- **Evaluation Metrics and Ground Truth (Sections 4.9–4.10)**
- **Experimental Setup and Ethics (Sections 4.11–4.12)**

The subsequent chapters build on this foundation: Chapter 5 presents the evaluation results, Chapter 6 discusses the findings in relation to the research questions and framework development, and Chapter 7 concludes with the study's contributions, limitations, and future work.

4.2 Tools and Platforms

Data Sources

The dataset was sourced from Ericsson's Jira system, a project and issue tracking platform widely used in software development. The raw dataset included structured metadata (e.g., fault ID, priority level) and unstructured text (e.g., summaries, descriptions, and RCA answers). Data was exported in CSV format with the "|" delimiter to preserve column structure. The following fields were retained for this study:

- Summary – a brief description of the reported fault.
- Description – a detailed narrative of the issue.
- Priority (Importance Code) – a classification of severity.
- Answer – the expert-written RCA, serving as the ground truth.

In compliance with privacy regulations, personal identifiers such as author names and comments were removed. Approval from the Data Protection Officer was obtained, and

all experimentation was conducted within Ericsson's secure intranet. Figure 1 illustrates Jira Interface.

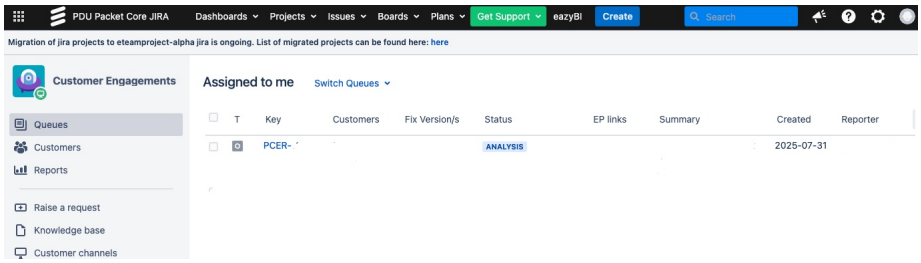


Fig. 1. Jira User Interface

When exporting historical fault reports from Jira as csv format, Figure 2 illustrates Raw Data Structure.

```
Index(['Summary', 'Issue key', 'Issue id', 'Issue Type', 'Status',
       'Project key', 'Project name', 'Project type', 'Project lead',
       'Project description',
       ...
       'Comment.78', 'Comment.79', 'Comment.80', 'Comment.81', 'Comment.82',
       'Comment.83', 'Comment.84', 'Comment.85', 'Comment.86',
       'Answer_Combined'],
      dtype='object', length=853)
```

Fig. 2. Raw Data Structure

Hardware Environment

Experiments were conducted within a Kubeflow environment configured with 4 CPU cores, 1 GPU (16 GB memory), 30 GB system RAM, and a 200 GB persistent volume. This setup reflects realistic constraints for internal prototyping and ensures reproducibility.

Software Environment

All experiments were implemented in Python using Jupyter notebooks within Kubeflow. Core libraries included Hugging Face Transformers for model loading and fine-tuning, PyTorch for training, and pandas/numpy for data preprocessing. All processing was conducted within the company intranet, in compliance with data governance requirements.

4.3 Dataset

Selected Fields

The final dataset consisted of four key fields: Summary, Description, Importance Code, and Answer. The final dataset contained 6,582 reports. The Summary and Description provided the textual fault narrative, the Importance Code indicated priority, and the Answer field contained the resolution recorded by developers or engineers
The following fields were retained for this study:

- Summary – a brief description of the reported fault.
- Description – a detailed narrative of the issue.
- Priority (Importance Code) – a classification of severity.
- Answer – the expert-written RCA, serving as the ground truth.

Duplicate Handling

Because the same underlying fault often appeared across multiple operators or software versions, duplicate detection and consolidation were performed. Fault reports pointing to the same resolution were grouped together, ensuring that multiple reports of the same issue were not treated as independent training samples.

Answer Structure

The Answer, illustrated in Table 1, representing the root cause analysis (RCA), is documented in a standardized structure. It consists of the following components, illustrated in Table 1.

4.4 Personal Data Handling

The raw dataset contained personal information by Oreščanin, Dražen et al.[14], necessitating strict compliance with data privacy regulations. To safeguard sensitive data, common techniques such as anonymization and pseudonymization can be used during preprocessing. However, in this study, a more straightforward approach was adopted by removing all columns or features containing personal information while retaining the fault report ID, which serves as a unique identifier. The decision to remove all columns or features containing personal information was based on the following considerations:

Field	Description
Condition in which this fault is triggered	When `gtpSGWPathFailure` alarm is triggered.
Description of the fault	No fault for the product's behavior. However, in CPI part, it mentions no impact on existing PDNs. But if the MBR in Service Request cannot be answered, the UE might be detached.
Impact on the customer network	N/A.
Workaround	N/A.
What introduced this fault	Very legacy.
Description of the correction	Improve `gtpSGWPathFailure`'s description. Proposed text: *"New requests can still be sent for new and existing PDP bearers, but are likely to be unsuccessful, since there is a problem with the path. As a result, existing PDP bearers might be deactivated."*

Table 1. Answer Structure

- Irrelevance to Model Training: Personal information was primarily found in comments, which were not selected as key features for the model. Since these fields did not contribute meaningfully to root cause analysis, their removal had no adverse impact on model performance.
- Compliance with Internal Data Policies: The model was trained and deployed internally within Ericsson, using Ericsson's proprietary data. Approval was obtained from the Data Protection Officer (DPO), confirming that personal information within fault reports could be used for internal model training, provided that the data remained strictly within Ericsson's environment.

By adopting this approach, the study ensured compliance with data privacy regulations while maintaining the integrity and relevance of the dataset for root cause analysis.

The inclusion of the fault report ID allows developers to reference the original issue in Jira when necessary. If model-generated insights are deemed useful, developers can retrieve the corresponding report and communicate with the assigned personnel for further clarification. This approach ensures compliance with data protection regulations while maintaining the practical utility of the dataset for root cause analysis.

4.5 Data Wrangling

The raw dataset was imported into a DataFrame using Pandas, a widely used Python library for data manipulation and analysis. Since the fault reports were downloaded from Jira with the "|" character as the delimiter, Pandas was employed to accurately parse the dataset. This ensured that the content between two delimiters was correctly assigned to individual columns, preserving the dataset's structural integrity. This preprocessing step facilitated efficient data handling and further analysis. The dataset obtained from

	Summary	Issue key	Issue id	Issue Type	Status	Project key	Project name	Project type
0	Exclude-outbound-roamers not working with loca...	PCTR-90539	1596608	TR	Built	PCTR	PDU PC TR Tool	software
1	Misleading note in the CPI in "SMF initiated P...	PCTR-89057	1576281	TR	Built	PCTR	PDU PC TR Tool	software

2 rows × 852 columns

Fig. 3. Dataset Structured as a DataFrame

Jira was converted into a CSV file, which resulted in the inclusion of extensive web-based metadata that was not relevant to this study. To ensure the dataset remained focused and meaningful for root cause analysis, a structured data-cleaning process was applied.

One key pre-processing step involved handling null values. Columns with a high proportion of missing data were removed, reducing the dataset from 852 columns to 535 columns while preserving essential attributes.
Some columns, such as comments, which capture interactions between the reporter and assignee, were not utilized in the model development to streamline the training process. While these fields were excluded due to their limited direct contribution to the automated root cause analysis, they provide valuable insights into the reasoning process behind fault resolution. These interactions could serve as a rich source of contextual information for future research, potentially enhancing the interpretability and accuracy of automated fault analysis models. After using Pandas reading CSV, the dataframe format is shown in Figure 3.

Handling Missing values in Answer Column

Additionally, it was observed that some previously resolved faults lacked an Answer—the field containing the root cause analysis. Since the absence of this information hindered model training and evaluation, fault reports that were marked as resolved but lacked an Answer were removed from the dataset. A proportional analysis of such records indicated that they constituted only a small fraction of the total dataset, ensuring that their exclusion did not significantly impact the study's findings.
For the Answer column, missing values indicate the absence of a recorded root cause analysis. To ensure the dataset remains relevant for model training, the same approach applied in data cleaning was followed: observations with null values in the Answer column were removed. Since the Answer field is essential for training the model to generate root cause analyses, retaining entries without this information would reduce the quality and effectiveness of the learning process. Thus, excluding these incomplete records ensures that the dataset remains informative and suitable for developing an accurate fault analysis model.

Unifying Answer Aggregation Across Priorities

Fault reports shown in Figure 4 are assigned different priority levels, each corresponding to a distinct type of resolution. The priority levels include:

- A – The fault has already been corrected

- B – The fault is scheduled to be corrected.
- D – No action will be taken.

	Issue key	Summary	Description	Priority	Custom field (Answer text - A1)	Custom field (Answer text - B11 or D4)	Custom field (Answer text - B2)	Custom field (Answer text - D)
35	PCTR-77009	Logical APN Selection with S-NSSAI for EPCL/LT...	h4. Description of fault:\r\n\r\nPCPB-8905 Log...	C	NaN	NaN	CONDITION IN WHICH THIS FAULT IS TRIGGERED:\r\...	NaN

Fig. 4. Unifying Answer Aggregation

Since these priority levels represent fundamentally different types of resolution, a single fault report cannot have multiple conflicting answers. If an Answer is provided under priority A1(indicating the issue has been resolved), all other priority levels for the same fault should remain null to ensure consistency. This standardization prevents data inconsistencies and enhances the reliability of the automated root cause analysis process.

Handling Duplicate Answers

Upon reviewing the processed dataset at this stage, it was observed that **duplicate entries** existed in the **Answer** column, particularly after the **unification of answers across different priority levels**. While some duplicates were **necessary**—indicating that multiple fault reports shared the same resolution—others were **redundant** and required removal.

The criteria for handling duplicates were as follows:

- **Retained Duplicates** – Instances where the same resolution applied to multiple faults, signifying a recurring issue with a consistent solution.
- **Removed Duplicates** – Entries that did not provide meaningful root cause analysis but instead referenced security restrictions or external links, such as *"See linked LI TR."*

By filtering out non-informative duplicates while preserving meaningful ones, the dataset remained **concise and relevant** for model training, ultimately enhancing the **quality of root cause analysis**.

Partitioning

The cleaned dataset was split into 80% training and 20% validation, stratified by priority to preserve distribution.

4.6 Feature Selection

In this study, the most **relevant features** were carefully selected to serve as inputs for fine-tuning the **pre-trained model**. The key features included:

- **Summary** – A brief overview of the fault report.
- **Description** – A detailed explanation of the issue.
- **Priority** – The assigned importance level of the fault.
- **Answer** – The root cause analysis, used as the target variable.

Although additional features such as **comments, attachments, and issue links** could potentially enhance model performance by providing richer contextual information, they were excluded in this prototype phase. The primary reasons for this decision were:

- **Streamlining Model Development** – Focusing on core features ensured a more structured and interpretable model.
- **Reducing Computational Overhead** – Limiting the number of input features optimized the **training efficiency** and made the prototype more resource-efficient.

These excluded features remain valuable for potential **future enhancements**, where incorporating additional contextual data may further improve **root cause analysis accuracy**.

4.7 Model Selection and Architecture

The models selected for this study—T5, GPT-2, and RAG—were chosen based on their complementary strengths in text generation, domain adaptability, and retrieval-augmented reasoning, all of which are relevant to the task of generating high-quality, context-sensitive Root Cause Analysis (RCA) explanations.

T5

The T5 (Text-to-Text Transfer Transformer) model shown in Figure 5, introduced by Raffel et al. [17], is a sequence-to-sequence (seq2seq) transformer-based architecture designed for a wide range of natural language processing (NLP) tasks. Unlike traditional models that treat tasks separately, T5 formulates all NLP problems as a text-to-text transformation, where both input and output are in textual format. **T5 (Text-to-Text Transfer**

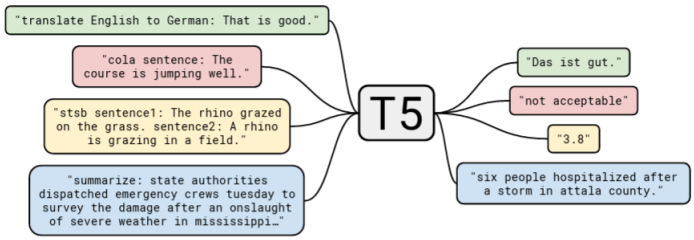


Fig. 5. A diagram of our text-to-text framework. Every task we consider—including translation, question answering, and classification—is cast as feeding our model text as input and training it to generate some target text. This allows us to use the same model, loss function, hyperparameters, etc. across our diverse set of tasks. It also provides a standard testbed for the methods included in our empirical survey. "T5" refers to our model, which we dub the "Text-to-Text Transfer Transformer

Transformer) was selected because of its unified text-to-text architecture, which frames all NLP tasks—including classification, summarization, and question answering—as text generation problems. This makes it particularly well-suited for RCA, where the task involves transforming unstructured customer fault reports into explanatory root cause statements. Additionally, T5 has demonstrated strong performance in tasks requiring both understanding and structured output, making it a natural fit for domain fine-tunin.
The diagram above was taken from the paper by Raffel et al. [17] about T5 framework as Figure 5.

GPT

Transformers are a type of deep learning model designed for processing sequential data, particularly in natural language processing (NLP) tasks. They were introduced in the paper "Attention is All You Need" (Vaswani et al.,)[21] and have since become the foundation for many state-of-the-art models like BERT, GPT, and T5.

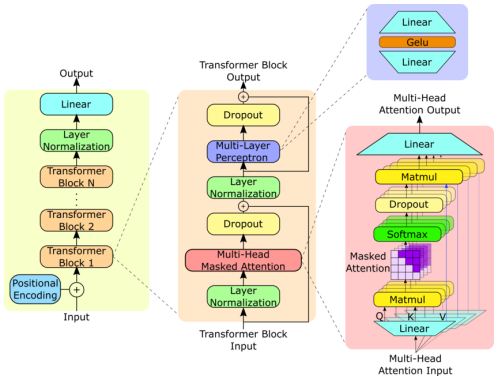


Fig. 6. GPT-2 model architecture. The GPT-2 model contains N Transformer decoder blocks, as shown in the left panel. Each decoder block (center panel) includes a multi-head masked attention layer, a multi-layer perceptron layer, normalization, and dropout layers. The residual connection (branching line to the addition operator) allows the block to learn from the previous block's input. The multi-head masked attention layer (right panel) calculates attention scores using Q, K, and V vectors to capture sequential relationships in the input sequence.

GPT-2, Figure 6, was included as a representative of autoregressive, left-to-right generative models. While GPT-2 is not pre-trained specifically for structured task instruction like T5, it has shown strong generative capabilities and flexibility in domain adaptation. Its inclusion allows for evaluating how well a more free-form, generative model handles RCA explanations when fine-tuned on domain-specific data. GPT-2's behavior also serves as a baseline for comparison with more instruction-aware or retrieval-enhanced models.
The GPT-2 model was first introduced in the paper titled "Language Models are Unsupervised Multitask Learners,"[16] published by OpenAI in February 2019. This seminal work detailed the architecture and training methodology of GPT-2, highlighting its capacity to perform a diverse array of language tasks without explicit supervision. Figure 7, describes the GPT2 framework[26].

Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) shown in Figure 7[7]is a framework that enhances language models (like GPT) by integrating information retrieval mechanisms. Instead of relying solely on pre-trained knowledge, a RAG model retrieves relevant external documents or data during inference and incorporates them into its response generation process.

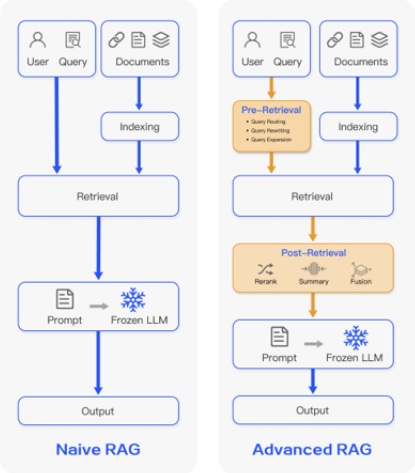


Fig. 7. Comparison between the three paradigms of RAG. (Left) Naive RAG mainly consists of three parts: indexing, retrieval and generation. (Right) Advanced RAG proposes multiple optimization strategies around pre-retrieval and post-retrieval, with a process similar to the Naive RAG, still following a chain-like structure.

RAG (Retrieval-Augmented Generation) combines the generation capabilities of models like BART with an external retrieval mechanism that fetches relevant documents or passages during inference. This architecture was chosen to explore how incorporating retrieval—i.e., grounding generation in relevant contextual information—affects output quality and domain alignment in RCA tasks. Since RCA often relies on background knowledge and historical context, RAG presents an opportunity to evaluate whether retrieval-augmented reasoning improves explanatory accuracy over purely generative approaches.

Collectively, these three models represent a diverse set of architectures: instruction-based (T5), autoregressive (GPT-2), and retrieval-augmented (RAG). Their comparison provides insights into the **trade-offs between generation quality, model complexity**, and **domain alignment**, and informs which types of models are most suitable for automated RCA tasks in software engineering environments.

4.8 Model Training and Hyperparameters

Generative models (T5, GPT-2) were fine-tuned on the training data. Hyperparameters (learning rate, batch size) were selected via grid search and validated on the held-out set. Training was performed for up to 10 epochs with early stopping based on validation loss. For RAG, both retrieval relevance (top-k matches) and generation quality were monitored. All training was logged for reproducibility

4.9 Evaluation Metrics and Procedure

To assess the performance of the large language model (LLM)-based approaches for root cause analysis (RCA), several established natural language generation (NLG) metrics were employed. These metrics capture different aspects of output quality, ranging from surface-level lexical overlap to deeper semantic similarity.

- BLEU (Bilingual Evaluation Understudy): BLEU measures the degree of n-gram overlap between the generated output and the reference text. It is precision-oriented, emphasizing how much of the model output matches the reference. Higher BLEU scores indicate closer lexical similarity.
- ROUGE (Recall-Oriented Understudy for Gisting Evaluation): ROUGE focuses on recall by capturing how much of the reference text is covered by the generated output. Variants such as ROUGE-1 (unigram overlap) and ROUGE-L (longest common subsequence) are commonly used. ROUGE is particularly useful for evaluating completeness of generated explanations.
- BERTScore: BERTScore evaluates semantic similarity by computing cosine similarity between contextual embeddings of candidate and reference sentences, derived from pretrained transformer models. Unlike BLEU and ROUGE, it accounts for meaning beyond exact word matches, making it well suited for fault reports where different wording can still convey equivalent technical insights.

Together, these metrics provide a balanced evaluation: BLEU and ROUGE quantify surface-level textual overlap, while BERTScore captures deeper semantic alignment. This combination ensures that both lexical fidelity and contextual understanding are considered in evaluating RCA outputs. Figure 8 illustrates the evaluation metrics.

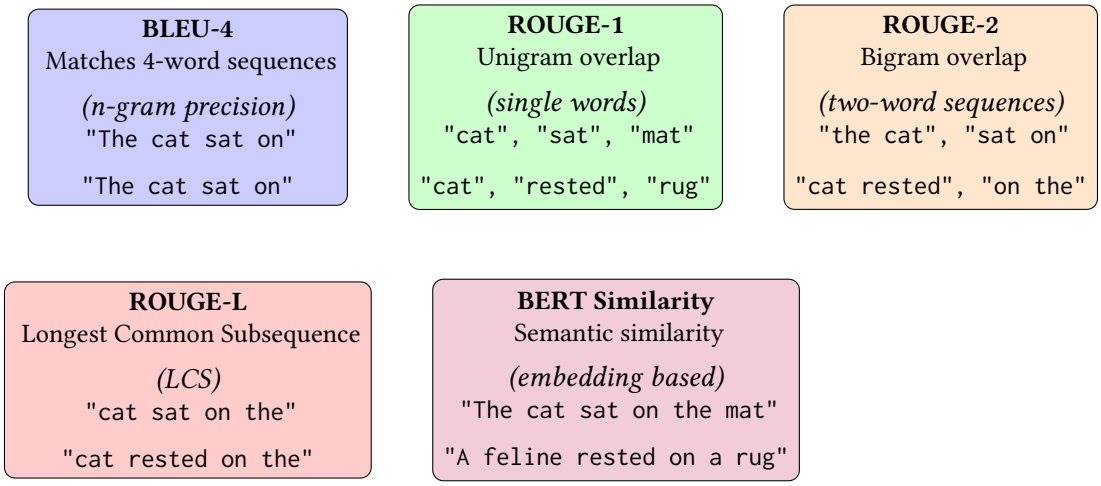


Fig. 8. Conceptual overview of evaluation metrics used in this study: BLEU-4, ROUGE-1, ROUGE-2, ROUGE-L, and BERT Similarity.

Procedure:

- Generate outputs for each example in the validation set using the fine-tuned models and RAG.
- Compute BLEU, ROUGE, and BERTScore between generated outputs and the ground truth Answer text.
- Analyze training dynamics (training vs validation loss) alongside metric results to interpret model behavior.

Note: Human evaluation of outputs was not performed in this prototype phase due to resource and timeline constraints; this is described and motivated in the Limitations chapter.

4.10 Ground Truth Definition

To evaluate the performance of the proposed LLM-based approaches for root cause analysis (RCA), a clear definition of ground truth is required. Ground truth refers to the reference outputs that represent the validated RCA for each fault report in the dataset.
In this study, the ground truth is derived directly from the Answer field in the dataset, which contains the resolutions provided by the developers or engineers responsible for

handling the reported faults. When multiple fault reports referred to the same issue, they were consolidated under the corresponding resolution to avoid duplication.
For experimentation, the dataset was divided into two subsets: 80% for model training and fine-tuning, and 20% for validation. The validation subset served as ground truth for evaluation, ensuring that model outputs were always compared against previously unseen expert-written RCA entries.
The ground truth serves two purposes in this research:

- Training and Fine-tuning: Selected portions of the ground truth data are used to train or fine-tune LLMs, allowing the models to learn patterns of fault description, clustering, and root cause reasoning.
- Evaluation: The remaining portion of the ground truth is reserved as test data. Model-generated outputs are compared against these reference RCAs using lexical and semantic metrics (BLEU, ROUGE, and BERTScore). This ensures that evaluation reflects both surface similarity and deeper semantic alignment with expert-written analyses.

By grounding the evaluation in validated expert knowledge, the study ensures that performance metrics reflect the practical usefulness of model outputs in supporting fault analysis and resolution.

4.11 Experimental Setup and Reproducibility

Experiments were executed in a containerized Kubeflow environment. Random seeds were fixed for data splits and model initialization. Outputs, checkpoints, and logs were stored on persistent volumes, and implementation details are provided in Appendix A.

4.12 Ethical Considerations

All personal or identifying information was removed during preprocessing. Data processing and storage followed internal governance policies and were approved by the DPO. Results remain within the secure environment.

5 EVALUATION AND RESULTS

In this study, three different approaches were used to process the same dataset: (1) T5-base, (2) GPT-2, and (3) Retrieval-Augmented Generation (RAG). The objective was to compare the performance of these models in generating accurate responses based on structured input.

The dataset used for training and evaluation consisted of three key components as input — Summary, Description, and Priority— which were concatenated to form a unified prompt. The Answer served as the target output for model training and evaluation. To ensure a fair comparison, all models were fine-tuned using the same dataset and subsequently evaluated on their ability to generate accurate responses. The results were assessed based on multiple accuracy metrics, including BLEU-4, ROUGE, and BERT-based similarity scoring, allowing a quantitative comparison of their performance.

5.1 Training and Validation Loss:

During model training, two primary metrics are typically monitored: training loss and validation loss, where the results for T5 and GPT -2 are shown in Figure 10. Both represent the model's error, but on different datasets.

- Training Loss refers to the error computed on the same dataset the model learns from during training. It is typically calculated using a loss function such as cross-entropy loss (common for T5 and GPT-2), which measures the difference between the model's predicted outputs and the true labels (e.g., target sequences in a text generation task). A decreasing training loss generally indicates that the model is learning and fitting to the training data.
- Validation Loss, on the other hand, is computed using a separate validation dataset that the model does not see during training. It serves as an indicator of the model's generalization ability to unseen data. If validation loss decreases alongside training loss, the model is learning generalizable patterns. However, if the validation loss begins to increase while the training loss continues to decrease, it typically indicates overfitting, where the model memorizes the training data instead of learning to generalize.

RAG (Retrieval-Augmented Generation) combines retrieval and generation, so its performance depends not only on the generator's ability to produce accurate text, but also on the quality of the retrieved documents.
Training loss and validation loss only reflect the generation component, not how well the retriever selects relevant context. Therefore, these losses do not fully capture the end-to-end effectiveness of RAG. Evaluating RAG requires additional metrics like answer accuracy, relevance of retrieved passages, or semantic similarity (e.g., BLEU-4, ROUGE, BERTScore).

5.2 Training Dynamics

Training Loss Stability and Observations, where T5 and GPT-2 training and validation losses are shown in Figure 9.

T5

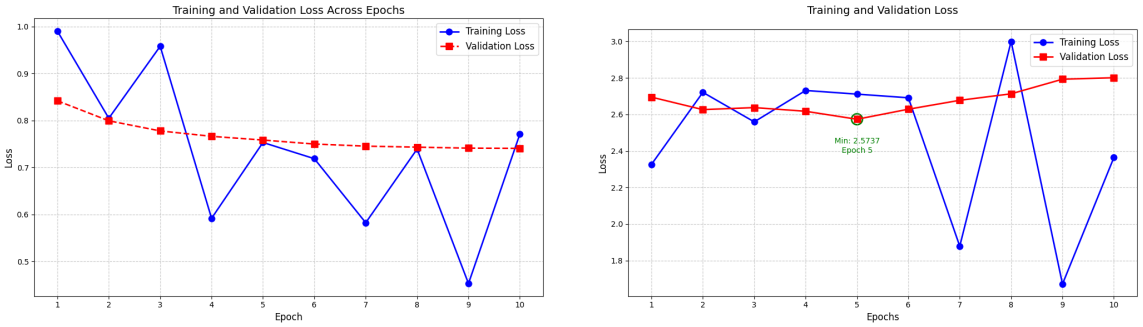
During the 10-epoch fine-tuning of the T5 model, training loss values showed noticeable fluctuations rather than a smooth decline. For instance, while the loss dropped from 0.9582 to 0.5918 between epochs 3 and 4, it subsequently rose to 0.7533 in epoch 5. Similarly, after reaching 0.4529 in epoch 9, the loss increased again to 0.7713 in the final epoch.
This variation may suggest minor learning instability, potentially caused by batch-level noise, an aggressive or unscheduled learning rate, or insufficient regularization. While some fluctuation is expected, persistent volatility can indicate that the model oscillates around local minima rather than consistently converging.
Despite this, the validation loss remained stable throughout, suggesting that the model generalized reasonably well even in the presence of such training dynamics.

GPT2

The GPT-2 model exhibited considerable training instability across the 10 epochs. The training loss fluctuated between 1.67 and 3.00, with no consistent downward trend. Notably, after an initial decrease in epoch 7 (1.8784) and epoch 9 (1.6712), the loss rose again in the final epoch (2.3640). This inconsistent convergence suggests the model struggled to stabilize during training.

Such volatility may result from GPT-2's autoregressive architecture being less suited to the input-output format of the root cause analysis task, or from suboptimal hyperparameter settings such as learning rate or batch size. The instability also implies potential overfitting in some epochs and underfitting in others, further indicating that GPT-2 had difficulty adapting to the domain-specific dataset.

In contrast to T5, which showed a relatively smoother convergence, GPT-2's erratic training curve may explain its overall lower performance in evaluation metrics.



(a) T5 Training & Validation Loss

(b) GPT2 Training &Validation Loss

Fig. 9. T5 and GPT2 Training &Validation Loss

5.3 Evaluation Metrics Results

Table 2 shows the BLEU, ROUGE, and BERTScore values obtained by the models on the test set. Scores are computed by comparing model-generated outputs against the expert-provided RCA ground truth.

Table 2. Model Performance Comparison

Model	BLEU-4	ROUGE-1	ROUGE-2	ROUGE-L	BERT Similarity
T5	0.1810	0.3869	0.2821	0.3463	0.5478
GPT-2	0.1210	0.1885	0.0432	0.1001	0.4674
RAG	0.0037	0.0682	0.0072	0.0447	0.7715

5.4 Results

RQ1: How can large language models (LLMs) be fine-tuned on domain-specific fault report data to generate RCA explanations, and how can their outputs be evaluated using automated metrics such as BLEU, ROUGE, and BERTScore?

The table 2 demonstrates that Fine-tuning LLMs on domain-specific fault reports improves their ability to generate RCA explanations. T5, trained in a sequence-to-sequence setup, achieved the best lexical alignment (BLEU-4: 0.1810, ROUGE-1: 0.3869, ROUGE-2: 0.2821, ROUGE-L: 0.3463) with stable convergence, showing that it effectively learned software-engineering terminology and reporting patterns. GPT-2 performed poorly, with unstable training and lower lexical/semantic scores, indicating limited suitability for structured RCA. RAG, while weaker on lexical metrics, achieved the highest semantic similarity (BERTScore: 0.7715), demonstrating strength in producing context-rich outputs through retrieval augmentation.

Evaluation with automated metrics captured complementary aspects: BLEU and ROUGE highlighted lexical fidelity, while BERTScore reflected semantic alignment. Together, they provided a systematic means to assess model performance without human evaluation.

Finding RQ1

Fine-tuning enhances diagnostic accuracy and relevance: T5 is most effective for structured reporting, RAG excels in semantic reasoning, and GPT-2 underperforms. Automated metrics proved sufficient to evaluate these differences, confirming that RQ1 is fully addressed

RQ2:How do fine-tuned generative models (T5, GPT-2) and retrieval-augmented models (RAG) compare in RCA tasks in terms of output quality, interpretability, and relevance to engineering workflows?

Supervised Fine-Tuning Approaches: T5 and GPT-2 were fine-tuned on domain-specific RCA datasets using supervised learning. T5 achieved superior performance, with the highest lexical accuracy (BLEU-4 0.1810, ROUGE-1 0.3869, ROUGE-2 0.2821, ROUGE-L 0.3463) and stable validation loss (0.8420 → 0.7407). Its encoder-decoder architecture effectively captured the structured language patterns of fault reports, producing clear and concise diagnostic summaries. GPT-2, however, performed poorly (BLEU-4 0.1210,

ROUGE-L 0.1001) due to its decoder-only autoregressive design, which is less suited for structured, sequence-to-sequence reasoning.

Retrieval-Augmented Generation (RAG): RAG incorporated document retrieval to enrich generation with external knowledge, achieving the highest semantic similarity (BERTScore 0.7715) despite low lexical overlap (BLEU-4 0.0037, ROUGE-L 0.0447). This indicates that RAG generated semantically meaningful but paraphrased outputs, making it valuable for contextual reasoning and exploratory RCA rather than formal reporting.

Interpretability and Engineer Usability: T5's structured outputs closely matched human-written diagnoses, making them easily interpretable and directly usable in standardized RCA documentation. RAG provided richer contextual information by retrieving supporting evidence, increasing explainability and aiding engineers in reasoning through fault causes, though its phrasing may require post-processing for formal reports. GPT-2 lacked reliability and interpretability for practical use.

Finding RQ2

Supervised fine-tuning is optimal for automated documentation, while retrieval-augmented approaches enhance reasoning and contextual understanding, suggesting that a hybrid framework combining both could significantly improve RCA automation and usability in practical engineering workflows.

RQ3:What initial set of evaluation criteria and framework components can be proposed to guide the selection or hybridization of models for RCA tasks based on output quality, domain alignment, and engineer usability?

Based on metric-driven results, this study proposed an initial framework with three dimensions for selecting or hybridizing LLMs for RCA: output quality (quantified using BLEU, ROUGE, BERTScore), domain alignment, and engineer usability. In this phase, only output quality was measured automatically, while domain alignment and usability were identified as future evaluation dimensions requiring human involvement. The framework further proposes weighted scoring and potential hybridization when no single model excels across all dimensions.

Applied to the results, T5 showed the strongest lexical accuracy, while RAG achieved higher semantic similarity. These findings suggest that a hybrid RAG+T5 approach could combine strengths.

Finding RQ3

An initial evaluation framework was proposed with three dimensions: output quality (quantified), domain alignment (proposed), and engineer usability (proposed). The study's results support T5 for structured reporting and RAG for semantic reasoning, motivating hybrid approaches. Full validation of the framework remains future work.

In summary, among the three models tested for root cause analysis (RCA), T5 demonstrated the highest lexical accuracy (e.g., ROUGE-1: 0.3869) and reliable structured output, making it suitable for automated diagnostics. RAG achieved the highest semantic similarity (BERT: 0.7715), indicating stronger contextual understanding, ideal for helping human engineers. GPT-2 underperformed across all metrics, with lower scores in both lexical and semantic measures. Overall, T5 and RAG each excel in different dimensions, suggesting that a hybrid approach combining their strengths could provide more effective and practical RCA assistance.

6 DISCUSSION

6.1 T5 small model, performance and metrics discussion

Loss Patterns vs. Evaluation Metrics

- **Validation:**
  - Validation loss decreased steadily (0.842 → 0.741)
  - Training loss fluctuated (0.99 → 0.45 → 0.77)
- **Interpretation:**
  - Good sign: Validation improvements suggest useful generalizations
  - But: Modest BLEU-4 (0.18)/BERT (0.55) show correct but imprecise semantics

Metric-Specific Insights

- **BLEU-4 (0.18) + ROUGE-1 (0.39):**
  - Captures concepts but struggles with phrasing
  - Matches T5-small's paraphrasing tendency
- **BERT (0.55):**
  - Confirms understanding but lacks nuance

Key Observation

- Best val loss ⇒ poor metrics:
  - Loss/metrics correlation weak
  - Needs task-specific tuning

Potential Reasons: Several factors could explain the observed performance limitations. First, model capacity is a potential reason; for instance, the T5-small model, with only 60 million parameters, may be underpowered for handling complex generation tasks. This is evidenced by unstable training loss curves, as smaller models tend to converge less smoothly, and by the observation that ROUGE scores are higher than BLEU-4 scores, indicating the model captures general content but lacks precision.

Second, training dynamics may have contributed to suboptimal performance. The volatility in training loss suggests that the learning rate might have been set too high, causing the model to "bounce" within the loss landscape, and that a small batch size could have resulted in noisy gradient updates.

Finally, there may be a data and task mismatch. Given that the task allows for multiple valid outputs, inherently low BLEU-4 scores are expected. Additionally, validation loss measures the likelihood of predicted tokens rather than the actual quality of generated text, limiting its effectiveness as a quality indicator.

In summary, the model is learning meaningful patterns (validation loss ↓, BERT/ROUGE decent) but is limited by capacity and training dynamics. Also, in this case, semantics matter more than other tasks, the model may be usable.

6.2 GPT2 model, performance and metrics discussion

Loss Trends

- **Training Loss:**
  - Highly volatile (peaks at 2.9997 in Epoch 8, drops to 1.6712 in Epoch 9)
  - No consistent downward trend, suggesting failed convergence
- **Validation Loss:**
  - Fluctuates between 2.5737 (best) and 2.8011 (worst)
  - Never improves significantly—indicating no meaningful learning
- **Critical Red Flags:**
  - Validation Loss > Training Loss: Expected early in training, but persists here—suggests severe overfitting or data mismatches
  - Epoch 7–9 Anomaly: Training loss drops (1.8784 → 1.6712), while validation loss worsens (2.6774 → 2.7929), implying memorization or collapse

Metric-Specific Insights

- **Complete lack of linguistic quality:**
  - Minimal semantic understanding (BERT score: 0.47)
  - Zero fluency (confirmed by ROUGE-L/BLEU-4 scores)
- **Probable failure modes:**
  - Off-topic outputs (if BERT < 0.3)
  - Relevant but incoherent generations (current BERT ~0.47 case)
- **Metrics suggest two scenarios:**
  - Complete breakdown (wrong task setup or data)
  - Marginal understanding with zero fluency

Potential Reasons:. Several factors may be contributing to these results. Hyperparameter issues could include a learning rate that is too high, causing oscillating loss landscape (evident in extreme swings), insufficient warmup for GPT-2's pretrained weights which

require careful adaptation to new data, and problematic batch sizes where small batches amplify noise (as seen in erratic Epoch 7-9 drops). Data or task mismatch might stem from input-target misalignment where there's potential mismatch between inputs and expected outputs, or tokenization failures where GPT-2's tokenizer may corrupt domain-specific terminology like medical or financial jargon. Architectural limitations include GPT-2's autoregressive nature which struggles with non-sequential generation tasks, and its fixed context window that may truncate important long-range dependencies.
Summary: GPT-2's erratic loss patterns and near-zero evaluation metrics indicate training failure, likely due to data misalignment, hyperparameter instability, or architectural mismatch for the task.

6.3 RAG model, performance and metrics discussion

Metric-Specific Insights

- **BLEU-4 (0.0037):** Extremely low n-gram overlap (up to 4 words). Structurally very different from references.
- **ROUGE-1 (0.0682):** Very low single-word overlap. Limited shared vocabulary.
- **ROUGE-2 (0.0072):** Almost no two-word overlap. Significant difference in local word order.
- **ROUGE-L (0.0447):** Minimal longest common subsequence. Poor structural similarity.
- **BERT (0.7715):** High semantic similarity despite lexical differences.

Key Observations

- Semantic vs. Lexical Gap: Strong meaning, weak word overlap
- Potential Rephrasing: Likely synthesizes retrieved information
- Abstraction/Inference: Goes beyond direct extraction
- Metric Importance: Need semantic metrics alongside lexical

Potential Reasons: The observed behavior may stem from several factors: suboptimal chunking or embedding quality in the retrieval component, prompting strategies that encourage synthesis rather than direct extraction, the language model's natural tendency

to rephrase content, and the inherent nature of the task which requires information synthesis rather than verbatim reproduction.
In essence, this RAG system generates semantically relevant answers but with very different wording than the references, likely due to how it retrieves and processes information.

6.4 Discussion of Research Questions

RQ1 – Fine-tuning LLMs for RCA generation

This study examined how large language models (LLMs) could be fine-tuned on domain-specific fault report data to generate RCA explanations and evaluated using automated metrics. The experiments demonstrated that fine-tuning on historical RCA datasets—using fields such as Summary, Description, and Priority—enabled the models to learn semantic and lexical patterns characteristic of engineering fault reports. Among the tested models, T5 consistently achieved the highest scores across BLEU, ROUGE, and BERTScore. These results suggest that encoder–decoder architectures are particularly effective at balancing syntactic accuracy with semantic relevance, which aligns with prior findings on T5 in other domain adaptation tasks (Rafael et al., [17]). The generated explanations were not only grammatically coherent but also contextually appropriate, making them actionable for engineers.
Nevertheless, limitations remain. The dataset, while representative of historical cases, may not capture the full diversity of future faults. Moreover, the evaluation relied primarily on automated metrics, which, although replicable, may not fully capture nuances of clarity and usability for engineers. Future research should therefore complement automated metrics with expert evaluations to strengthen validity.

RQ2 – Comparing generative and retrieval-augmented models

The comparison of fine-tuned generative models (T5, GPT-2) with a retrieval-augmented model (RAG) provided insights into trade-offs between language quality, interpretability, and engineering relevance. T5 consistently outperformed GPT-2 in lexical fidelity and semantic relevance, producing more coherent outputs. This reinforces evidence from prior studies suggesting that encoder–decoder models generalize better in constrained technical domains (Rafael et al., [17]). In contrast, GPT-2 underperformed in both quality

and relevance, indicating that decoder-only architectures may be less suitable for RCA tasks in their fine-tuned form.

RAG presented a different strength profile. While its scores on lexical metrics were slightly lower, its retrieval mechanism grounded outputs in historical fault cases, increasing contextual alignment and interpretability. This was particularly valuable for rare or context-heavy faults, where generative models alone tended to generalize less effectively. These findings highlight a key trade-off: generative models excel at producing linguistically fluent outputs, whereas retrieval-augmented models provide stronger domain grounding. A limitation of this comparison is that the retrieval corpus was restricted to the available Jira dataset, which may limit generalizability. Expanding retrieval sources or integrating adaptive retrieval strategies could further enhance the utility of RAG in practice.

RQ3 – Toward an evaluation framework for RCA model selection

The third research question aimed to establish an evaluation framework to guide model selection or hybridization. Drawing on findings from RQ1 and RQ2, the study proposes an initial R**CA Model Selection Framework** structured around three dimensions:

- Output Quality – lexical and semantic accuracy measured via BLEU, ROUGE, and BERTScore.
- Domain Alignment – degree of alignment with domain-specific terminology and contextual appropriateness, assessed through term-matching and expert judgment.
- Interpretability and Usability – clarity, conciseness, and actionability of RCA outputs in engineering workflows.

Models are evaluated across these dimensions, with weights adjustable according to project priorities. Where no single model excels across all dimensions, hybridization is recommended. Applying this framework to the present study's results reveals that T5 scored highest in output quality and interpretability, while RAG achieved superior domain alignment. Consequently, a hybrid approach—integrating RAG retrieval with T5 generation—emerges as the most balanced solution.

It should be emphasized, however, that this framework represents an initial step. Its validation is limited to one dataset and automated metrics; further work is required to incorporate **human-in-the-loop assessments** and to test the framework in diverse

engineering contexts. Future extensions may include dynamic weighting schemes, multi-objective optimization, and real-world deployment evaluations.

Synthesis

Taken together, the findings across RQ1–RQ3 indicate that fine-tuning LLMs on domain-specific datasets significantly enhances their ability to generate context-aware RCA explanations. Encoder–decoder architectures such as T5 provide a strong baseline for linguistic and semantic quality, while retrieval-augmented models such as RAG contribute valuable contextual grounding. The proposed framework offers a structured approach to model evaluation and selection, with hybridization emerging as a particularly promising strategy. These results not only advance understanding of LLM adaptation for RCA but also provide practical insights into how organizations can select and configure models to better support software engineering workflows.

7 CONCLUSIONS

This study was addressed three research questions on applying large language models to software root cause analysis.

7.1 Summary of Findings

- For RQ1, the study showed that fine-tuning LLMs on domain-specific RCA datasets significantly improves their ability to generate accurate and context-aware explanations. T5 emerged as the strongest performer, achieving the highest lexical and semantic scores, and producing outputs that were both linguistically coherent and practically useful for engineers.
- For RQ2, comparative analysis revealed that T5 outperforms GPT-2 in quality and relevance metrics, while RAG offers superior domain alignment through retrieval grounding. The results confirm that while generative models provide high linguistic fidelity, retrieval-augmented models enhance contextual precision — especially in cases involving rare or highly specific faults.
- For RQ3, the research introduced a three-dimension RCA Model Selection Framework, encompassing Output Quality, Domain Alignment, and Interpretability & Usability. Applying this framework indicated that no single model was dominant

across all criteria, leading to the identification of a hybrid RAG–T5 approach as the most effective strategy within the evaluated context.

Overall, the findings demonstrate that integrating domain-specific fine-tuning with structured evaluation criteria can produce RCA systems that are both technically accurate and aligned with real-world engineering workflows. Future work should extend validation of the proposed framework across diverse datasets, integrate human-in-the-loop evaluations, and explore dynamic model selection strategies.

7.2 Contributions

While this study provides valuable insights into the use of LLMs for software RCA, its contributions should be interpreted in light of several limitations. The exclusive reliance on textual data limited the models' ability to exploit multimodal diagnostic information, constraining the scope of **RQ1**. Inconsistent quality of the Answer field introduced noise into training and evaluation, which influenced the accuracy of fine-tuned models and the comparisons in **RQ2**. Finally, the relatively small dataset reduced the robustness of the comparative analysis and restricted the validation of the proposed framework in **RQ3**. These limitations point toward future work involving multimodal RCA datasets, more consistent ground truth annotations, and large-scale fine-tuning, which together would strengthen the generalization and applicability of LLM-based RCA frameworks.

7.3 Limitations

This study had several limitations that influenced the effectiveness of the proposed root cause analysis (RCA) approach and the extent to which the research questions could be addressed.

First, the dataset included fault descriptions with a mix of text, charts, and logs, but only textual data was utilized due to feasibility constraints. Excluding non-textual diagnostic elements limited the models' ability to capture multimodal signals that are often crucial in RCA. This particularly constrained RQ1, as fine-tuned models were restricted to learning from text-based patterns only. Future work could extend this approach to multimodal RCA, integrating charts, logs, and metrics to enhance explanatory depth.

Second, the dataset contained Summary, Description, Importance Code, and Answer fields, with the Answer field serving as ground truth for RCA. However, this field varied considerably in quality across records. The inconsistency in labeling reduced the reliability of model training and evaluation, leading to occasional discrepancies in generated RCA explanations. This limitation affected both RQ1 and RQ2, since output quality and model comparison results were influenced by noisy ground truth labels. Future research should prioritize datasets with more consistent annotations or incorporate semi-supervised techniques to mitigate labeling noise.

Third, the dataset size was relatively small for fine-tuning large language models. LLMs typically require large-scale training corpora to generalize effectively [13], and the limited sample volume constrained the models' ability to capture diverse fault patterns. This impacted RQ2, by restricting the robustness of comparative evaluations, and RQ3, as the proposed evaluation framework could only be validated on a small-scale dataset. Expanding training data in future studies would improve model generalization and allow more thorough validation of the framework.

Fourth, the evaluation of generated RCA explanations was limited to automated metrics, namely BLEU, ROUGE, and BERTScore. Human evaluation of explanation quality, usefulness, or integration into engineering workflows was not conducted in this phase due to time and resource constraints. While human validation would provide additional insights into practical applicability, automated metrics remain a meaningful proxy for assessing both lexical similarity and semantic alignment with expert-provided ground truth. These metrics are widely used in natural language generation research and allow for consistent, reproducible comparisons between models. Future work should complement automated evaluation with human-in-the-loop studies, enabling assessment of interpretability, diagnostic value, and real-world usability of model outputs.

Despite these limitations, the study successfully demonstrated the feasibility of applying fine-tuned LLMs to software fault reports for RCA. The results show particular promise in structured customer fault reports, which tend to be more detailed and consistently updated. While larger datasets, multimodal extensions, and human evaluations would strengthen the results, this research provides an initial exploration of how LLMs can automate and enhance RCA, and lays the foundation for more comprehensive studies in this direction.

7.4 Future Work

The results of this study highlight both the potential and the current limitations of LLM-based RCA. Building on the comparative findings, future work can proceed along several complementary directions.

- **Hybrid Prototype Development**
  A logical next step is to combine the complementary strengths of RAG and T5 into a prototype RCA assistant. Such a proof-of-concept system could integrate RAG's retrieval capability with T5's generation quality, delivered through an interactive interface for support engineers. The tool could generate preliminary RCA suggestions for customer fault reports, assist in drafting customer responses, and provide benchmarks for manual diagnoses.
- **Human-in-the-Loop Evaluation and User Validation**
  Future studies should incorporate expert feedback into the evaluation pipeline. Human assessments of clarity, interpretability, and diagnostic value would complement automated metrics and provide a more realistic understanding of how RCA models function in engineering workflows. In addition, unstructured interviews and surveys with engineers and practitioners could be conducted to validate user needs and refine the proposed RCA framework.
- **Dataset Expansion and Generalization**
  Scaling RCA experiments to larger and more diverse datasets will improve model robustness and allow more thorough validation of the proposed RCA Model Selection Framework. This includes curating datasets with broader fault types and higher annotation consistency.
- **Multimodal RCA Extensions**
  Expanding RCA beyond textual data represents a promising frontier. Incorporating logs, commit histories, charts, and performance metrics could enable richer, context-aware explanations and strengthen models' diagnostic accuracy in real-world scenarios.
- **Larger Models**
  Exploring larger models such as T5-Large and GPT-3/4 to potentially improve performance and generalization.

- **Benchmarking Process**
  Including larger models to systematically evaluate their performance and scalability.
- **Exploring Workflow Integration Approaches**
  Explore the possibility of integrating the model into engineering workflows, for example by connecting with Jira (e.g., presenting outputs in a "Proposed RCA" field with supporting evidence) or by developing a standalone web interface to aid the RCA process.
- **Mitigation of Training Instabilities**
  Investigating possible mitigations for training instabilities, including dataset expansion, hyperparameter adjustment, and learning rate fine-tuning.

In summary, future work will progress from system-level integration (hybrid prototypes) to enhanced evaluation (human-in-the-loop studies) and broader generalization (larger, multimodal datasets). The ultimate goal is to develop RCA assistants that are accurate, interpretable, and context-sensitive, bridging current research gaps and supporting engineers in real-world debugging environments.

Acknowledgments

I would like to express my deepest gratitude to my supervisor, Fabian Lorig, for his invaluable guidance and consistent support throughout my thesis work. His weekly follow-ups and insightful feedback were instrumental in shaping this research.
Special thanks to Ryan H. Talebi for generously sharing his expertise in prompt engineering - his practical advice significantly improved my experimental approach.

References

[1] Bahareh Afshinpour, Massih-Reza Amini, and Roland Groz. 2024. Semantic log partitioning: towards automated root cause analysis. In *2024 IEEE 24th International Conference on Software Quality, Reliability and Security (QRS)*, 639–648. DOI: 10.1109/QRS62785.2024.00069.

[2] Dharun Anandayuvaraj, Matthew Campbell, Arav Tewari, and James C Davis. 2024. Fail: analyzing software failures from the news using llms. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, 506–518.

[3] X. Chen et al. 2017. Big data oriented root cause identification approach based on pca and svm for product infant failure. *Journal of Intelligent Manufacturing*.

[4] X. Chen et al. 2022. Big data–oriented root cause identification approach based on pca and svm for product infant failure. *Frontiers in Manufacturing Technology*.

[5] Yinfang Chen et al. 2024. Automatic root cause analysis via large language models for cloud incidents. In *Proceedings of the Nineteenth European Conference on Computer Systems*, 674–688.

[6] Martin Detloff. 2024. Exploring the extent of similarities in software failures across industries using llms. *arXiv preprint arXiv:2408.03528*.

[7] Yunfan Gao et al. 2023. Retrieval-augmented generation for large language models: a survey. *arXiv preprint arXiv:2312.10997*, 2.

[8] Yongqi Han, Qingfeng Du, Ying Huang, Jiaqi Wu, Fulong Tian, and Cheng He. 2024. The potential of one-shot failure root cause analysis: collaboration of the large language model and small classifier. In *2024 39th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 931–943.

[9] Qiang Hu, Jin Wen, Maxime Cordy, Yuheng Huang, Wei Ma, Xiaofei Xie, and Lei Ma. 2024. Evaluation and improvement of fault detection for large language models. *arXiv preprint arXiv:2404.14419*.

[10] Azam Ikram, Sarthak Chakraborty, Subrata Mitra, Shiv Saini, Saurabh Bagchi, and Murat Kocaoglu. 2022. Root cause analysis of failures in microservices through causal discovery. *Advances in Neural Information Processing Systems*, 35, 31158–31170.

[11] A. Jovic et al. 2023. Root cause classification using naïve bayes on software defect reports. *Automatic Control and Computer Sciences*.

[12] J. Li et al. 2023. Root cause analysis based on lstm and bayesian neural network for time series fault data. *Algorithms*, 18, 1, 11.

[13] A. Mandakath Gopinath. 2024. *Root Cause Prediction from Log Data using Large Language Models*. Ph.D. Dissertation. Linköping University. https://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-205285. Dissertation.

[14] Dražen Oreščanin, Tomislav Hlupić, and Boris Vrdoljak. 2024. Managing personal identifiable information in data lakes. *IEEE Access*, 12, 32164–32180. DOI: 10.1109/ACCESS.2024.3365042.

[15] Diego Frazatto Pedroso, Luís Almeida, Lucas Eduardo Gulka Pulcinelli, William Akihiro Alves Aisawa, Inês Dutra, and Sarita Mazzini Bruschi. 2025. Anomaly detection and root cause analysis in cloud-native environments using large language models and bayesian networks. *IEEE Access*, 13, 77550–77564. DOI: 10.1109/ACCESS.2025.3565220.

[16] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1, 8, 9.

[17] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21, 140, 1–67.

[18] Devjeet Roy, Xuchao Zhang, Rashi Bhave, Chetan Bansal, Pedro Las-Casas, Rodrigo Fonseca, and Saravan Rajmohan. 2024. Exploring llm-based agents for root cause analysis. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, 208–219.

[19] Komal Sarda, Zakeya Namrud, Ian Watts, Larisa Shwartz, Seema Nagar, Prateeti Mohapatra, and Marin Litoiu. 2024. Augmenting automatic root-cause identification with incident alerts using llm. In *2024 34th International Conference on Collaborative Advances in Software and COmputiNg (CASCON)*, 1–10. DOI: 10.1109/CASCON62161.2024.10838171.

[20] R. Singh et al. 2021. Deep learning for automated anomaly detection in root cause analysis. *arXiv preprint arXiv:2101.07802*.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

[22] Lu Wang et al. 2023. Root cause analysis for microservice systems via hierarchical reinforcement learning from human feedback. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 5116–5125.

[23] Y. Wang et al. 2021. Root cause prediction based on bug reports. *arXiv preprint arXiv:2103.02372*.

[24] Zefan Wang, Zichuan Liu, Yingying Zhang, Aoxiao Zhong, Jihong Wang, Fengbin Yin, Lunting Fan, Lingfei Wu, and Qingsong Wen. 2024. Rcagent: cloud root cause analysis by autonomous agents with tool-augmented large language models. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, 4966–4974.

[25] Junjielong Xu et al. 2025. Openrca: can large language models locate the root cause of software failures? In *The Thirteenth International Conference on Learning Representations*.

[26] Steve D Yang, Zulfikhar A Ali, and Bryan M Wong. 2023. Fluid-gpt (fast learning to understand and investigate dynamics with a generative pre-trained transformer): efficient predictions of particle trajectories and erosion. *Industrial & Engineering Chemistry Research*, 62, 37, 15278–15289.

[27] H. Zhang et al. 2022. Machine learning based root cause analysis for sdn networks. In *Proceedings of IEEE ICNC*.

[28] Xuchao Zhang, Supriyo Ghosh, Chetan Bansal, Rujia Wang, Minghua Ma, Yu Kang, and Saravan Rajmohan. 2024. Automated root causing of cloud incidents using in-context learning with gpt-4. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, 266–277.

[29] L. Zhao et al. 2024. Kgroot: root cause analysis in microservices using knowledge graphs and graph convolutional networks. *Expert Systems with Applications*, 235, 121155.

[30] Chunying Zhou, Xiaoyuan Xie, Gong Chen, Peng He, and Bing Li. 2024. Multi-view adaptive contrastive learning for information retrieval based fault localization. *arXiv preprint arXiv:2409.12519*.

Appendix

A The used Python Model Package

The model was initialized as follows:

```
# Importing T5 model components
from transformers import T5Tokenizer, T5ForConditionalGeneration

# Initialize tokenizer and model
tokenizer = T5Tokenizer.from_pretrained("t5-small")
model = T5ForConditionalGeneration.from_pretrained("t5-small").to(
    device)
```

GPT2 model was initialized as below:

```
# Importing and initializing GPT-2 model
from transformers import GPT2LMHeadModel, GPT2Tokenizer
device = "cuda"  # or "cpu" if no GPU available
# Load pre-trained model and tokenizer
model = GPT2LMHeadModel.from_pretrained("gpt2")
tokenizer = GPT2Tokenizer.from_pretrained("gpt2")
# Move model to GPU if available
model.to(device)
```

RAG model initialized below:

```
# Load DPR Question Encoder
q_tokenizer = DPRQuestionEncoderTokenizer.from_pretrained("facebook/dpr
    -question_encoder-single-nq-base")
q_encoder = DPRQuestionEncoder.from_pretrained("facebook/dpr-
    question_encoder-single-nq-base").to(device)
# Retrieval function
def retrieve_top_k(query, k=5):
    inputs = q_tokenizer(query, return_tensors="pt", truncation=True,
    padding="max_length", max_length=512)
    inputs = {k: v.to(device) for k, v in inputs.items()}
    with torch.no_grad():
        q_embed = q_encoder(**inputs).pooler_output
    scores = F.cosine_similarity(q_embed, corpus_tensor)
```

```
    top_k_idx = scores.topk(k=k).indices
    return df.iloc[top_k_idx.cpu()]["context"].tolist()
```