RAG for Root Cause Analysis

Krintanpreet Singh (40257186)
Department of Electrical and Computer Engineering
Concordia University
Montreal, Quebec

Muhammad Hussnain Uz Zaman (40258916)
Department of Electrical and Computer Engineering
Concordia University
Montreal, Quebec

***Abstract*—This project employs Retrieval-Augmented Generation (RAG) for root cause analysis in microservices environments. By transforming datasets representing issues across various traffic scenarios and storing them in Pinecone, we efficiently retrieve relevant data for analysis. A JSON file detailing service dependencies helps us understand the microservices architecture. The training datasets stored in Pinecone serve as inputs to the RAG model, which analyzes issues and identifies root causes. Our evaluation includes three scenarios: analysis without RAG, with RAG using training data in Pinecone, and with RAG using both synthetic and training data in Pinecone. The generated analysis provides insights into root causes, impacts on dependencies, pathways of influence, affected metrics, and mitigation strategies. Our results demonstrate the effectiveness of RAG in root cause analysis for microservices, emphasizing the significance of data-driven decision-making in managing complex systems.**

***Index Terms*—RAG (Retrieval Augmented Generation), Focused Evaluation, Synthetic datasets, Extensive Evaluation, RCA (Root Cause Analysis), LLM (Large Language Model)**

I. Introduction

In today's rapidly evolving technological landscape, microservices architectures have become a cornerstone for building scalable and resilient applications. However, the complexity and interdependencies inherent in microservices can lead to challenges in diagnosing and resolving performance issues. Traditional root cause analysis methods often fall short due to the distributed nature of these systems. To address this, our project leverages Retrieval-Augmented Generation (RAG) for root cause analysis. By transforming datasets from various traffic scenarios and storing them in Pinecone, we enable efficient retrieval and analysis of relevant data. Utilizing a JSON file that maps service dependencies and dependents, we gain a comprehensive understanding of the microservices architecture. Our approach integrates training datasets with RAG, facilitating the identification of root causes and impacted nodes. We evaluate our method using focused and extensive prompts across different scenarios, demonstrating the effectiveness of RAG in enhancing root cause analysis for microservices. This project underscores the importance of data-driven approaches in managing complex systems and offers a robust framework for improving system reliability and performance.

II. Background

A. Understanding the Root Cause Analysis (RCA)

Root Cause Analysis (RCA) is a methodological approach for identifying the fundamental reasons behind faults, failures, or adverse events within a system. By understanding the root causes, organizations can implement measures to prevent future occurrences. RCA involves several steps, including data collection, cause identification, and the implementation of corrective actions. This approach is critical in various industries, including manufacturing, healthcare, and IT, to ensure continuous improvement and reliability.

By leveraging AI to understand the root causes, organizations can implement proactive and effective measures to prevent future occurrences. RCA involves several steps, including automated data collection, advanced data analysis, AI-driven cause identification, and the implementation of data-informed corrective actions. This approach is critical in various industries, such as manufacturing, healthcare, and IT, to ensure continuous improvement, enhanced reliability, and optimal performance.

B. Challenges for Performance Issue Analysis in Microservices

- **Complexity:** Microservices architecture decentralizes services, which means that each service operates independently. This independence, while beneficial for modularity and scalability, complicates performance analysis because issues can arise from interactions between services rather than within a single monolithic application. There services often depend on each other, forming a web of interdependencies. Performance issues in one service can cascade and affect others, making it difficult to pinpoint the root cause of a problem without comprehensive tracing and analysis tools.rewrite this in better and consciese way
- **Distributed Nature of Systems:** Since requests could be hopping from server to server, performance bottlenecks can emerge from various sources such as network latency, resource contention, or misconfigurations across geographically dispersed systems or servers.
- **Scalability:** Microservices architecture faces scalability challenges addressed by horizontal and vertical scaling. Horizontal scaling adds instances to enhance scalability but complicates load balancing and state management. Vertical scaling increases per-instance resources like CPU and memory, posing complexities in resource allocation and performance prediction. Both approaches are vital for optimizing microservices performance.

- **Inter-Service Communication:** Communication between microservices over the network introduces latency, which can significantly impact performance. Analyzing and optimizing network latency requires a deep understanding of the network infrastructure and the communication patterns between services.
- **Data Consistency:** Data consistency in microservices often relies on eventual consistency, which can result in temporary data discrepancies. Balancing performance and ensuring timely data synchronization pose significant challenges. Moreover, ensuring data replication across services and regions impacts performance, necessitating thorough monitoring and bottleneck identification in replication processes.
- **Monitoring Overhead:** The process of collecting, storing, and analyzing performance metrics from numerous services can be resource-intensive and may itself impact performance.

C. Advantages of Large Language Models (LLMs)

- **Natural Language Understanding:** LLMs excel at processing and interpreting complex technical documents, logs, and incident reports written in natural language.
- **Automation:** They can automate the analysis of large volumes of data, reducing the need for extensive manual intervention and speeding up the RCA process.
- **Scalability:** They can handle large amounts of data and compute power efficiently, making them suitable for processing vast datasets and complex tasks.
- **Pattern Recognition:** LLMs can detect patterns and anomalies within large datasets that might be overlooked by human analysts, facilitating more accurate RCA.

D. Disadvantages of Large Language Models (LLMs)

- **Biases:** LLMs can reflect biases present in the training data, leading to biased outputs or reinforcing societal prejudices if not carefully managed.
- **Interpretability:** Understanding how LLMs arrive at their outputs can be challenging, raising concerns about transparency and accountability in decision-making processes.
- **Computational Resources:**Training and deploying LLMs require significant computational resources, which can be costly and environmentally impactful.
- **Ethical Concerns:** There are ethical implications regarding the responsible use of LLMs, particularly in generating misleading content or misinformation.

E. Choice of LLMs

The selection of NousResearch/Hermes-2-Pro-Mistral-7B as our Large Language Model (LLM) is crucial for root cause analysis in microservices architectures. This model excels in understanding and generating natural language, making it ideal for analyzing complex microservice interactions. Its advanced contextual comprehension and large parameter count enable it to handle extensive prompts and generate detailed insights. By leveraging Hermes-2-Pro-Mistral-7B, we can produce precise analyses and actionable recommendations, enhancing the reliability and performance of microservices systems. This choice underscores our commitment to using state-of-the-art tools to address the challenges of root cause analysis in distributed environments.

III. The Architecture Pipeline of RCA

A. Data Preparation and Embedding

- **Data Collection:** This involves gathering relevant data from various sources such as logs, performance metrics, and incident reports.
- **Preprocessing:** Cleaning and normalizing data to remove noise and ensure consistency across the dataset.
- **Embedding and Vectorization:** Converting textual data into numerical vectors using embedding techniques like BERT embeddings to make it comprehensible for LLMs. Textual data is converted into high-dimensional vectors that preserve semantic meaning and enable efficient similarity searches. This transformation supports rapid retrieval and analysis during anomaly detection and root cause identification phases.

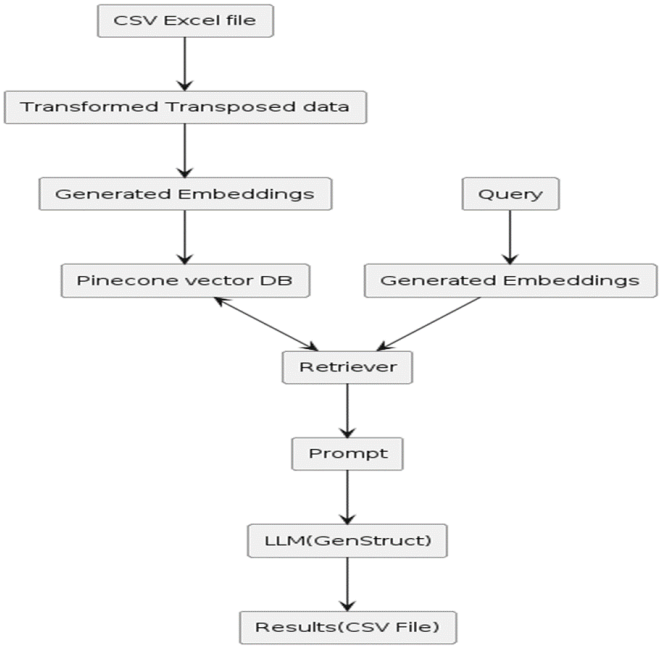

Fig. 1. Architecture diagram

The embedded data vectors serve as inputs to LLMs, which leverage these representations to generate insightful responses. By integrating embeddings with language models, the RCA pipeline enhances its capability to diagnose complex issues. This structured approach from data collection through embedding ensures that the RCA pipeline is well-equipped to handle diverse datasets and extract valuable insights efficiently.

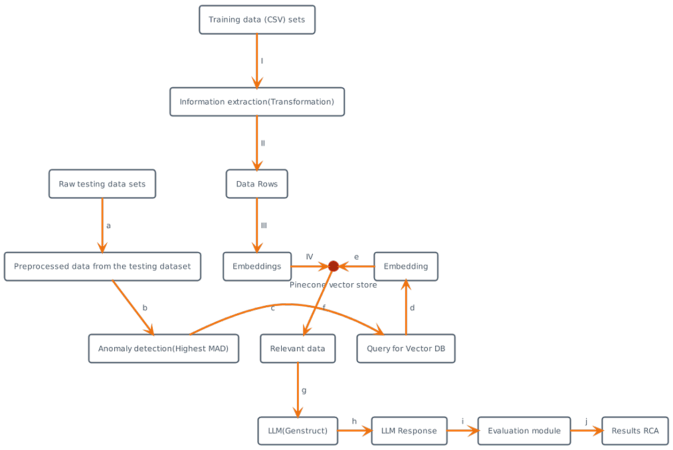

Fig. 2. RAG pipeline

B. Architecture Patterns: Without RAG

Using LLMs directly for analyzing data and generating insights without the augmentation of external knowledge sources. This method is straightforward and often quicker to deploy since it doesn't require integration with external databases or knowledge sources. However, it may limit the depth and breadth of insights generate.

C. Architecture Patterns: With RAG

The architecture for root cause analysis can be significantly enhanced with the use of Retrieval-Augmented Generation (RAG). This involves enhancing LLMs with a retrieval mechanism that fetches relevant information from external databases to support and augment the analysis. This retrieval mechanism fetches relevant information based on the input query or context before generating a response. By incorporating external data, RAG improves the depth and accuracy of insights produced by LLMs. ChatGPT In the RAG process, data undergoes initial transformation, and embeddings are created using models such as BAAI/bge-large-en-v1.5. These embeddings are stored in a vector database like Pinecone to enable quick retrieval. Anomalies detected in test data generate queries that are compared against the vector database to fetch relevant data points. These retrieved data points are used to formulate detailed prompts, which are fed into a language model to generate responses.

IV. The Strategies for Constructing Prompts and Prompt Refinement

Strategies for constructing initial prompts involve defining clear objectives to focus on extracting specific insights from data. Using precise and unambiguous language tailored to the data's context and domain helps avoid varied interpretations and inaccurate responses.

- Initial Prompts: Crafting initial questions or prompts designed to extract specific information from the data.
- Refinement: Iteratively improving the prompts based on feedback and results to enhance clarity and relevance, ensuring that the LLM generates more accurate responses.

It's crucial to include key parameters and balance simplicity with complexity to capture the depth of analysis required for instance taking into consideration the microservice information, service dependency related data, historical data etcetera . Iterative testing with a subset of data helps refine prompts for effectiveness. Prompt refinement strategies include evaluating response quality, gathering feedback from professsor and research scholars, and analyzing performance metrics to enhance clarity, specificity, and relevance. Adapting prompts based on results with defined section for root cause node analysis, target node analysis, MAD score, and suggesting mitigation strategies etcetera ensures the language model consistently deliver accurate and insightful responses.

V. The Retrieval Patterns and/or Augmentation Patterns (Path Dependencies)

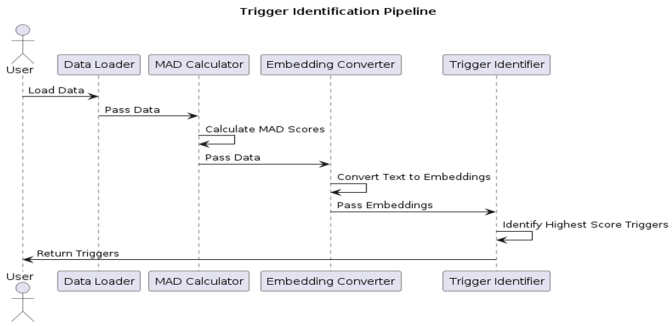

Fig. 3. Trigger identification pipeline

The Retrieval and Augmentation Patterns are designed to ensure that the Root Cause Analysis (RCA) process is comprehensive and effective. Within the Trigger Identification Pipeline, the process begins with the Data Loader importing the necessary data. The MAD Calculator then computes Median Absolute Deviation (MAD) scores to identify outliers, highlighting significant deviations within the data. This information is crucial for focusing on the most relevant data points. The data is then transformed into vector embeddings by the Embedding Converter, which captures the semantic meaning of the text for easier analysis. The embeddings are analyzed by the Trigger Identifier to pinpoint the highest score triggers based on the MAD scores. To ensure a focused and relevant

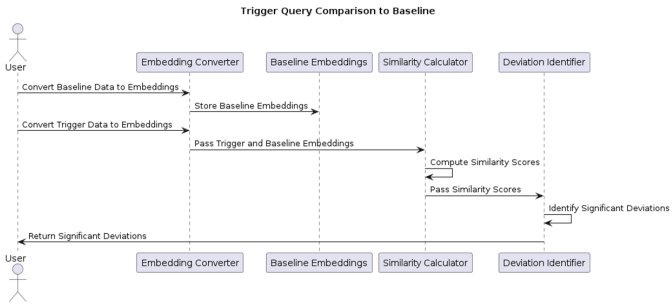

Fig. 4. Trigger query identification

analysis, queries are executed to retrieve data related to these anomalies. In the Retrieval-Augmented Generation (RAG) Pipeline, embeddings are utilized to query a vector store, such as Pinecone, to find data points that are semantically similar to the query. This retrieval process enriches the context for further analysis by providing detailed and relevant data. The retrieved data helps in constructing detailed prompts for the language model, ensuring that the generated responses are well-informed and pertinent. By leveraging this combination of outlier detection, data retrieval, and semantic analysis, the retrieval and augmentation patterns enhance the RCA process, making it both thorough and effective.

VI. LLMs Configuration and Response Generation

Configuring the Language Model (LLM) involves several steps to optimize its performance for root cause analysis (RCA) tasks. The first step is model selection, choosing an appropriate pre-trained model that can handle the complexity of RCA tasks. For instance, the model 'BAAI/bge-large-en-v1.5' is used for generating embeddings, while 'NousResearch/Hermes-2-Pro-Mistral-7B' is utilized for text generation. Fine-tuning the model on domain-specific data enhances its ability to understand and generate relevant responses. Parameter tuning is crucial for balancing response quality and computational efficiency. We use a top(k) value of '0.8', a temperature of '0.9', a maximum token size of '1200', and a return sequence of '1'. Ensuring seamless integration with the data retrieval system allows the model to effectively use the retrieved data to generate responses.

VII. The Augmentation of Synthetic Data

Augmenting synthetic data can significantly enhance the training process and improve the robustness of the model. Synthetic data generation involves creating datasets that mimic real-world scenarios but cover a broader range of conditions, helping train the model to handle diverse and rare cases. Integrating synthetic data with actual training data provides a more comprehensive training set, improving the model's ability to generalize from the training data.

In our project, we use Pinecone as the vector database to efficiently store and retrieve embeddings. For generating embeddings, we utilize 'BAAI/bge-large-en-v1.5' for training data and 'sentence-transformers/all-MiniLM-L6-v2' for synthetic data. Scenario simulation with synthetic data ensures the model can handle extreme or rare scenarios that may not be well-represented in the actual data. Validating and testing the model using synthetic data ensures it performs well across different conditions and accurately identifies and analyzes anomalies.

By following these structured approaches in data preparation, architecture design, prompt construction, retrieval patterns, LLM configuration, and synthetic data augmentation, the RCA process becomes more efficient, accurate, and robust.

VIII. The Evaluation

A. Definition of Confusion Matrix

A confusion matrix is a fundamental tool used to evaluate the performance of classification models. It provides a comprehensive summary of prediction results by comparing actual and predicted classifications. Framing the confusion matrix differs in each of the evaluation scenarios given below. In focused evaluation, the primary goal is to verify the correctness of both the root cause node and the target node against the paths specified in the `paths_json` file. The confusion matrix is calculated based on the following rules:

- **True Positive (TP)**: Both the root cause node and the target node generated in the responses are correct and present in the path.
- **False Positive (FP)**: One of the nodes (root cause or target node) is present in the path but not both.
- **False Negative (FN)**: Neither node is in the path.
- **True Negative (TN)**: Considered as 0 in all cases.

This approach ensures that the evaluation focuses on the critical nodes involved in the anomaly detection and root cause analysis process.

Evaluation Metrics

The resulting confusion matrix for focused evaluation is used to calculate the following metrics:

Precision

$$\text{Precision} = \frac{TP}{TP + FP} \tag{1}$$

Recall

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2}$$

F1-score

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

Matthews Correlation Coefficient (MCC)

Since TN is considered 0 in all cases, the MCC formula simplifies accordingly:

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{4}$$

B. Focused Evaluation(With RAG)

In focused evaluation, the goal is to validate the accuracy of both the root cause node and the target node against predefined paths in the 'paths_json' file. The confusion matrix distinguishes True Positives (TP) when both nodes are correct and present in the path, False Positives (FP) when only one node appears in the path, False Negatives (FN) when neither node is in the path, and True Negatives (TN), which are consistently zero. This approach prioritizes the

critical nodes in anomaly detection and root cause analysis. The resulting confusion matrix metrics—precision, recall, F1-score, and Matthews Correlation Coefficient (MCC)—offer a detailed assessment of the model's accuracy and reliability in navigating complex system interactions.

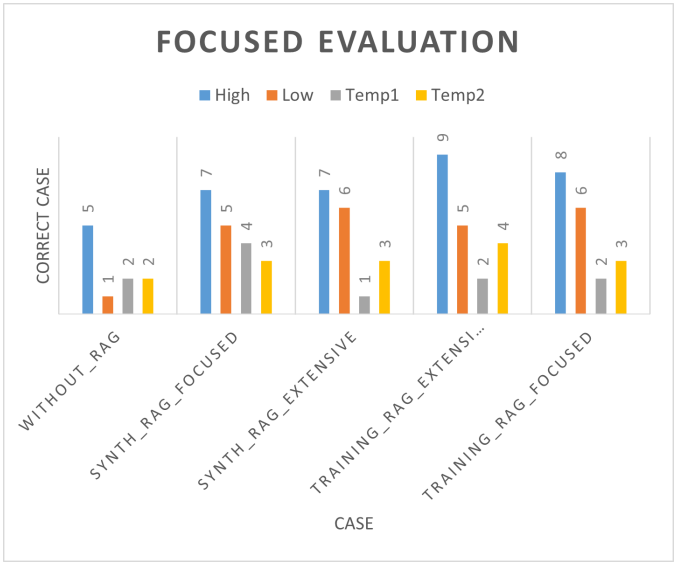

Fig. 5. Focused Evaluation Graph

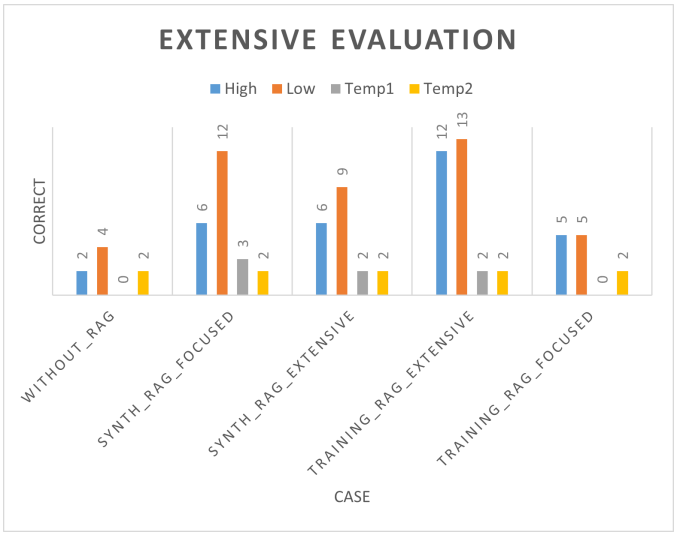

Fig. 6. Extensive Evaluation Graph

C. Extensive Evaluation

Extensive evaluation looks at response quality across several areas: discussing dependencies, dependents, paths, metrics, and mitigation efforts. Each area gets a score of 0 or 1, adding up to a maximum of 5. The confusion matrix then categorizes results based on these scores: True Positive (TP) when the score is 5, False Positive (FP) for scores between 2 and 4, False Negative (FN) for scores below 2, and True Negative (TN) always at 0. This method gives a thorough look at how well responses match up with anomaly situations.

In our evaluation, scenarios without Retrieval-Augmented Generation (RAG) showed very scores, suggesting they provide some insights but miss major details. Using synthetic data with RAG improved scores, especially in least common traffic scenarios, shows the model can handle differ situations better. With extensive training data and RAG, scores were highest across all scenarios, proving the model can consistently give accurate and detailed responses. This shows RAG combined with good training data makes the model strong in spotting and explaining anomalies effectively.

D. Strict Evaluation

In strict evaluation, confusion matrices are generated for each of the root cause nodes and target nodes by comparing them with ground truth files. However, this method is often not accurate as it does not account for the complexity and interconnectedness of the microservices architecture, leading to potentially misleading results. Due to these limitations, strict evaluation is not preferred. Instead, focused and extensive evaluations are employed for a more nuanced and accurate assessment of the model's performance.

E. Text Similarity Analysis

Evaluating the similarity between generated text and reference text is crucial for understanding model performance. This study uses cosine similarity and BLEU scores to measure the similarity between hypotheses generated from different prompting strategies. The analysis focuses on five scenarios, aiming to determine the consistency and variation in generated text.

1) Cosine Similarity Calculation: TF-IDF vectorization was applied to the text data, and cosine similarity matrices were computed. The cosine similarity matrix indicates how similar each hypothesis is to every other hypothesis within the same sheet. The values range from 0 to 1, where 1 indicates identical text.

2) BLEU Score Calculation: The BLEU score, which measures the overlap of n-grams, was calculated for each hypothesis against its corresponding prompt. The BLEU scores provide a measure of the similarity between each hypothesis and its corresponding prompt.

IX. Summary of Best Practices

Effective data preparation and embedding are crucial for successful RCA in microservices architectures. This process begins with robust data transformation techniques to convert raw data into a structured format. Utilizing pre-trained models such as 'BAAI/bge-large-en-v1.5' for generating embeddings captures the semantic meaning of data points, enabling precise querying and retrieval. Storing these embeddings in a vector database like Pinecone ensures efficient and scalable data retrieval, essential for timely RCA. Integrating RAG into RCA enhances analysis accuracy and depth. The project's evaluation showed RAG's superiority in handling complex dependencies and providing insightful root cause analysis. Effective

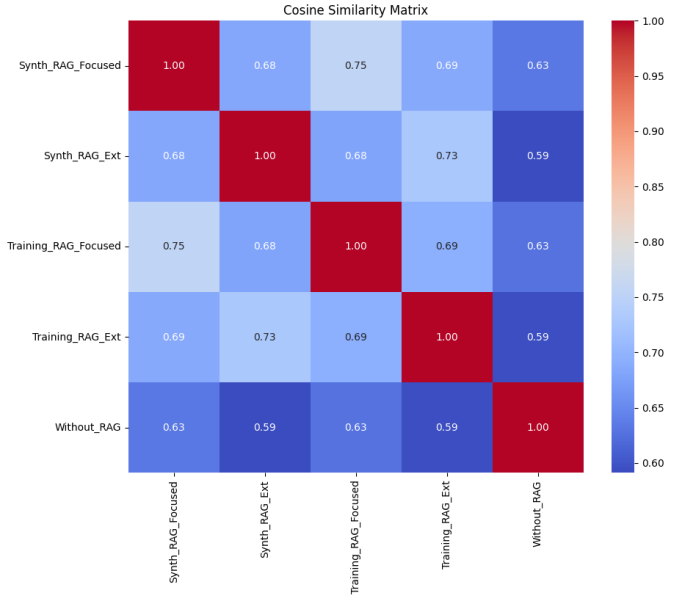

Fig. 7. High Cosine Matrix

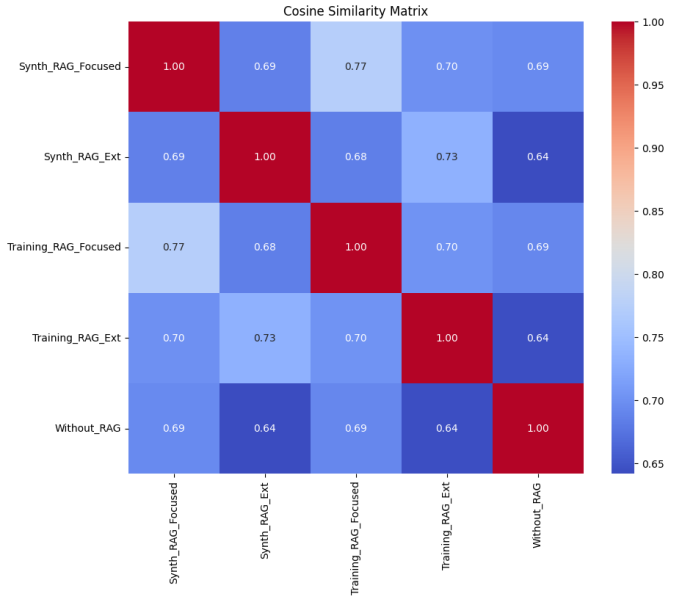

Fig. 8. Low Cosine Matrix

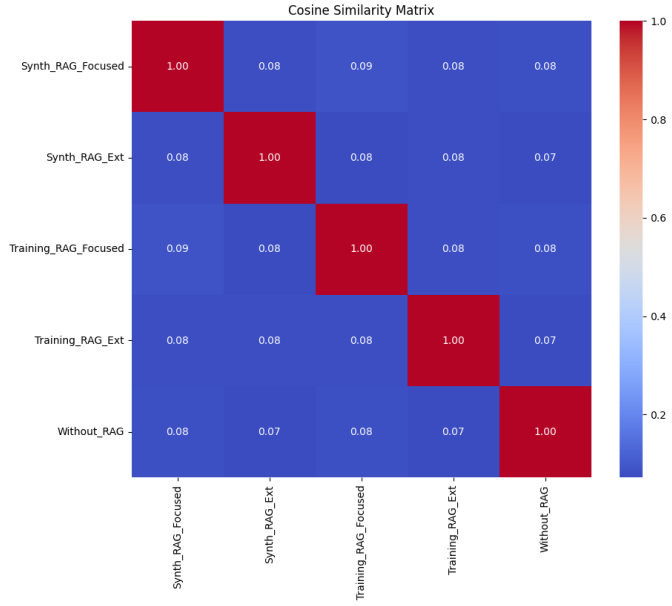

Fig. 9. Temp1 Cosine Matrix

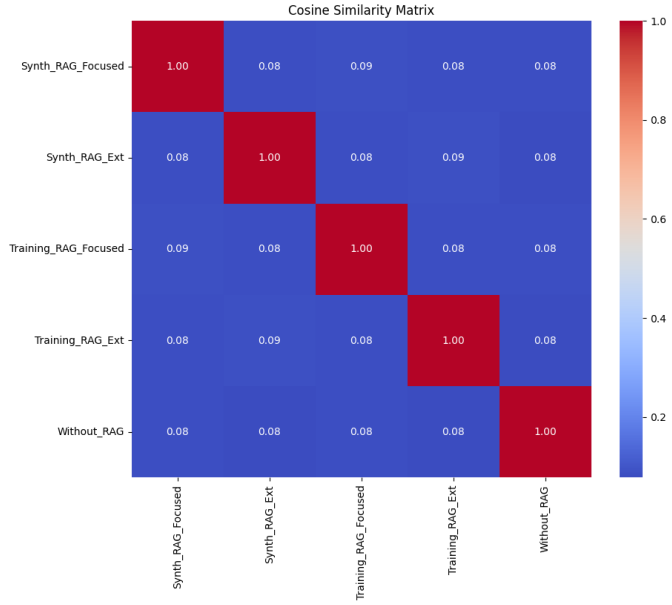

Fig. 10. Temp2 Cosine Matrix

prompt construction is key, involving specific prompts with comprehensive background information. Iterative refinement of prompts based on model responses improves output quality. Retrieval patterns using embeddings from the Pinecone vector store ensure semantically similar data points are retrieved, providing rich context for analysis. Configuring language models like 'BAAI/bge-large-en-v1.5' for embeddings and 'NousResearch/Hermes-2-Pro-Mistral-7B' for text generation involves fine-tuning on domain-specific data and adjusting parameters for optimal performance. Synthetic data augmentation enhances training, enabling the model to handle diverse scenarios. Creating synthetic datasets that mimic real-world conditions helps the model generalize better. Integrating synthetic with actual training data ensures the model is well-prepared for various situations, improving reliability and accuracy. Following these best practices in data preparation, architecture design, prompt construction, retrieval patterns, LLM configuration, and synthetic data augmentation ensures an efficient, accurate, and robust RCA process.

By following these best practices in data preparation, architecture design, prompt construction, retrieval patterns, LLM configuration, and synthetic data augmentation, the RCA

process becomes more efficient, accurate, and robust. This comprehensive approach ensures that microservices architecture can effectively manage and mitigate anomalies, leading to improved system reliability and performance.

X. Conclusion

In conclusion, the integration of Retrieval-Augmented Generation (RAG) for Root Cause Analysis (RCA) within microservices environments represents a significant advancement in managing and diagnosing performance issues in complex systems. By leveraging advanced embedding techniques and a robust retrieval process using Pinecone's vector store, this approach enhances the precision and depth of analysis. Effective data preparation and embedding are crucial for successful RCA in microservices architectures. This process begins with robust data transformation techniques to convert raw data into a structured format. Utilizing pre-trained models such as BAAI/bge-large-en-v1.5 for generating embeddings captures the semantic meaning of data points, enabling precise querying and retrieval. Storing these embeddings in a vector database like Pinecone ensures efficient and scalable data retrieval, essential for timely RCA. Integrating RAG into RCA enhances analysis accuracy and depth. The project's evaluation showed RAG's superiority in handling complex dependencies and providing insightful root cause analysis. Effective prompt construction is key, involving specific prompts with comprehensive background information. Iterative refinement of prompts based on model responses improves output quality. Retrieval patterns using embeddings from the Pinecone vector store ensure semantically similar data points are retrieved, providing rich context for analysis. Configuring language models like BAAI/bge-large-en-v1.5 for embeddings and NousResearch/Hermes-2-Pro-Mistral-7B for text generation involves fine-tuning on domain-specific data and adjusting parameters for optimal performance. Synthetic data augmentation enhances training, enabling the model to handle diverse scenarios. Creating synthetic datasets that mimic real-world conditions helps the model generalize better. Integrating synthetic with actual training data ensures the model is well-prepared for various situations, improving reliability and accuracy. Overall, the adoption of RAG in RCA underscores the importance of data-driven methodologies in enhancing the reliability and performance of distributed systems. By following these best practices in data preparation, architecture design, prompt construction, retrieval patterns, LLM configuration, and synthetic data augmentation, the RCA process becomes more efficient, accurate, and robust. This comprehensive approach ensures that microservices architecture can effectively manage and mitigate anomalies, leading to improved system reliability and performance. Github Repository Link:https://github.com/hussnainzaman/RAG-for-RCA.git

References

[1] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," arXiv preprint arXiv:2005.11401, 2020. Available: https://arxiv.org/abs/2005.11401.

[2] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang, "REALM: Retrieval-Augmented Language Model Pre-Training," arXiv preprint arXiv:2002.08909, 2020. Available: https://arxiv.org/abs/2002.08909.

[3] W. Shi, S. Min, M. Yasunaga, M. Seo, R. James, M. Lewis, L. Zettlemoyer, and W. Yih, "REPLUG: Retrieval-Augmented Black-Box Language Models," arXiv preprint arXiv:2301.12652, 2023. Available: https://arxiv.org/abs/2301.12652.

[4] K. Shuster, S. Poff, M. Chen, D. Kiela, and J. Weston, "Retrieval Augmentation Reduces Hallucination in Conversation," arXiv preprint arXiv:2104.07567, 2021. Available: https://arxiv.org/abs/2104.07567.

[5] M. Hardt, W. R. Orchard, P. Blöbaum, E. Kirschbaum, and S. P. Kasiviswanathan, "The PetShop Dataset — Finding Causes of Performance Issues across Microservices," arXiv preprint arXiv:2311.04806, 2024. Available: https://arxiv.org/abs/2311.04806.

[6] GitHub - amazon-science/petshop-root-cause-analysis. Available: https://github.com/amazon-science/petshop-root-cause-analysis.

[7] GitHub - gayathritela/RCA-of-Cloud-Microservices. Available: https://github.com/gayathritela/RCA-of-Cloud-Microservices.

[8] GitHub - pranaysood/Root-Cause-Analysis. Available: https://github.com/pranaysood/Root-Cause-Analysis.