



Universidade de Brasília

Instituto de Ciências Exatas
Departamento de Ciência da Computação

Detecção e Interpretação de Anomalias em Logs de Sistemas de TI por meio de Inteligência Artificial

Alysson Cristiano Estevam de Moura

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Orientador
Prof. Dr. Lucas Bondan

Brasília
2024

Ficha catalográfica elaborada automaticamente,
com os dados fornecidos pelo(a) autor(a)

dM929dd de Moura, Alysson Cristiano Estevam
Detecção e Interpretação de Anomalias em Logs de Sistemas
de TI por meio de Inteligência Artificial / Alysson
Cristiano Estevam de Moura; orientador Lucas Bondan. --
Brasília, 2024.
95 p.

Dissertação(Mestrado Profissional em Computação Aplicada)
-- Universidade de Brasília, 2024.

1. Detecção de Anomalias. 2. Aprendizado de Máquina Não
Supervisionado. 3. Processamento de Linguagem Natural . 4.
Modelos de Linguagem de Grande Escala. I. Bondan, Lucas,
orient. II. Título.

Dedicatória

In memoriam de meu pai, cuja dedicação e trabalho seguem me inspirando; à minha mãe, por sua presença e apoio em todas as etapas da minha vida; e à minha filha, cuja alegria e amor dão sentido a cada conquista.

Agradecimentos

Agradeço primeiramente a Deus, por me conceder forças, resiliência e sabedoria para enfrentar e superar os desafios desta jornada.

Expresso minha sincera gratidão ao corpo docente do Programa de Pós-graduação em Computação Aplicada (PPCA). Em especial, ao meu orientador, Prof. Dr. Lucas Bondan, por sua orientação dedicada, paciência e esforço incansáveis, que foram essenciais para a realização deste trabalho.

Meus agradecimentos também se estendem aos professores do grupo de estudos de infraestrutura, cujas valiosas contribuições foram fundamentais para o aprimoramento desta pesquisa.

Aos colegas do mestrado, sou grato pelas trocas de experiências, pelo apoio mútuo e pela parceria ao longo desta caminhada.

Sou igualmente grato aos meus colegas de trabalho, pelo suporte e pela compreensão que permitiram conciliar minhas responsabilidades profissionais e acadêmicas.

Por fim, minha gratidão se estende a todos que, de alguma forma, contribuíram para o desenvolvimento e conclusão deste projeto, incluindo a equipe administrativa do programa, cujo suporte eficiente foi essencial em toda a trajetória acadêmica.

Resumo

Este estudo explora a detecção de anomalias em registros de *log* de uma instituição financeira por meio de técnicas de aprendizado de máquina não supervisionado, combinadas com processamento de linguagem natural (*Natural Language Processing - NLP*) e Modelos de Linguagem de Grande Escala (*Large Language Models - LLMs*), utilizados para interpretação. A diversidade, o volume massivo e a natureza não estruturada dos *logs* representam desafios significativos para automatizar a detecção e a análise de anomalias. O trabalho propõe um fluxo de trabalho que utiliza aprendizado de máquina não supervisionado combinado com NLP para identificar anomalias e LLMs para interpretá-las, permitindo identificar causas prováveis de falhas e vulnerabilidades em sistemas de TI. O desempenho de oito modelos foi avaliado com *logs* do *Blue Gene/L* (BGL), sendo os mais eficazes validados com registros de *log* do *Microsoft Configuration Manager* (MCM), oriundos de uma instituição financeira. A validação prática desses modelos em cenários reais destaca a aplicabilidade das soluções propostas, demonstrando como técnicas de aprendizado de máquina e NLP podem ser implementadas eficazmente em ambientes complexos. Os resultados experimentais indicam que redes neurais, como *Self-Organizing Map* (SOM) e *Autoencoders* (AE), foram particularmente eficientes, atingindo *F1-Scores* de 0,86 e 0,80, respectivamente, quando aplicados aos *logs* do MCM. Este estudo não apenas comprova a eficácia dos modelos na detecção de anomalias, mas também apresenta uma abordagem prática e inovadora ao utilizar LLMs para interpretar os eventos detectados. Isso contribui para a melhoria da segurança e confiabilidade dos sistemas da instituição financeira, automatizando o monitoramento e acelerando a análise de falhas.

Palavras-chave: detecção de anomalia, PLN, log, aprendizado não supervisionado, LLM

Abstract

This study explores anomaly detection in log records from a financial institution using unsupervised machine learning techniques, combined with Natural Language Processing (NLP) and Large Language Models (LLMs) for interpretation. The diversity, massive volume, and unstructured nature of logs pose significant challenges for automating anomaly detection and analysis. The proposed workflow employs unsupervised machine learning with NLP to identify anomalies and LLMs to interpret them, enabling the identification of probable causes of failures and vulnerabilities in IT systems. The performance of eight models was evaluated using logs from Blue Gene/L (BGL), with the most effective models validated through log records from Microsoft Configuration Manager (MCM), collected from a financial institution. The practical validation of these models in real-world scenarios highlights the applicability of the proposed solutions, demonstrating how machine learning and NLP techniques can be effectively implemented in complex environments. The experimental results indicate that neural networks, such as the Self-Organizing Map (SOM) and Autoencoders (AE), were particularly efficient, achieving F1-Scores of 0.86 and 0.80, respectively, when applied to MCM logs. This study not only confirms the effectiveness of the models in detecting anomalies but also presents a practical and innovative approach by using LLMs to interpret detected events. This contributes to enhancing the security and reliability of the financial institution's systems by automating monitoring and accelerating failure analysis.

Keywords: anomaly detection, NLP, log, unsupervised learning, LLM

Sumário

1	Introdução	1
1.1	Contextualização	1
1.2	Definição do Problema	2
1.3	Hipótese de Pesquisa	3
1.4	Justificativa e Delimitação do Problema	3
1.5	Objetivo Geral	4
1.5.1	Objetivos Específicos	4
1.6	Contribuição para a Pesquisa	4
1.7	Estruturação dos Capítulos	5
2	Fundamentação Teórica	6
2.1	Contextualização	6
2.1.1	Registros de <i>Log</i> e <i>Log Parsing</i>	6
2.1.2	Extração de Características	7
2.1.3	Detecção de Anomalias	10
2.1.4	Interpretação de Anomalias com NLP	11
2.2	Trabalhos Relacionados	15
3	Proposta	19
3.1	Fluxo de Trabalho	19
3.1.1	Coleta de <i>Logs</i>	19
3.1.2	<i>Log Parsing</i>	20
3.1.3	Janelamento	21
3.1.4	Pré-Processamento	21
3.1.5	Extração de Características	22
3.1.6	Detecção de Anomalias	23
3.1.7	Interpretação de Anomalias	24
4	Experimentos	27
4.1	Metodologia e Critérios de Avaliação dos Modelos	27

4.1.1	Métricas de Desempenho para Detecção de Anomalias	27
4.1.2	Métricas para Comparação de Respostas dos LLMs	29
4.1.3	Cenários de Avaliação	30
4.2	Experimentos com o BGL	33
4.2.1	Experimento 1.1 - Comparação de <i>F1-Score</i> , <i>Recall</i> , <i>Precision</i> e curva ROC	33
4.2.2	Experimento 1.2 - Análise da Sensibilidade dos Modelos diante da Variação da Taxa de Contaminação	34
4.3	Experimentos com o CCMEval	34
4.3.1	Experimento 2.1 - Comparação de <i>F1-Score</i> , <i>Recall</i> , <i>Precision</i> e curva ROC	34
4.3.2	Experimento 2.2 - Interpretação das Anomalias no <i>log</i> CCMEval . .	35
4.4	Configuração Experimental	37
4.4.1	Configuração de Hiperparâmetros	37
4.4.2	Ferramentas e Tecnologias Utilizadas	37
5	Resultados e Discussão	39
5.1	Experimentos com o BGL	39
5.1.1	Experimento 1.1 - Comparação de <i>F1-Score</i> , <i>Recall</i> , <i>Precision</i> e curva ROC	39
5.1.2	Experimento 1.2 - Análise da Sensibilidade dos Modelos diante da Variação da Taxa de Contaminação	41
5.2	Experimentos com o CCMEval	42
5.2.1	Experimento 2.1 - Comparação de <i>F1-Score</i> , <i>Recall</i> , <i>Precision</i> e curva ROC	42
5.2.2	Experimento 2.2 - Interpretação das Anomalias no <i>log</i> CCMEval . .	44
6	Conclusões e Trabalhos Futuros	62
6.1	Perspectivas para a Continuidade do Trabalho	64
	Referências	66
	Apêndice	69
A	Artigo Apresentado no SLATE 2024	70

Lista de Figuras

2.1	Exemplo de <i>Log parsing</i>	7
2.2	Tipos de janelamento	8
3.1	Fluxo de Detecção e Interpretação de Anomalias em <i>Logs</i>	20
4.1	Exemplos de mensagens do <i>log</i> BGL. As mensagens foram selecionadas aleatoriamente, e a ordem não reflete a sequência original de registros.	31
5.1	Desempenho dos modelos com janelas de tempo fixas no BGL.	40
5.2	Curvas ROC dos modelos com janelas de tempo fixas no BGL.	40
5.3	Desempenho com o aumento da taxa de contaminação no <i>log</i> BGL.	42
5.4	Desempenho dos modelos com janela de tempo fixa no CCMEval.	43

Lista de Tabelas

2.1	Comparação dos trabalhos relacionados	17
4.1	Tamanho e Composição do Conjunto de Dados BGL	32
4.2	Dados Resumidos do BGL	32
4.3	Dados Resumidos do CCMEval	33
5.1	Métricas de Similaridade entre Modelos - Sequência Anômala 1	49
5.2	Métricas de Similaridade entre Modelos - Sequência Anômala 2	54
5.3	Métricas de Similaridade entre Modelos - Sequência Anômala 3	60

Capítulo 1

Introdução

Neste capítulo, apresenta-se o problema de pesquisa, começando pela contextualização do cenário atual da análise de *logs* e sua relevância em ambientes críticos, como o setor financeiro. Em seguida, detalha-se a definição do problema, delimita-se o escopo da pesquisa e formula-se a hipótese. Por fim, são apresentados a justificativa para o desenvolvimento deste estudo, os objetivos gerais e específicos e as contribuições esperadas.

1.1 Contextualização

A identificação de anomalias em arquivos de registros de eventos de sistemas – conhecidos como *logs* – é um processo fundamental na manutenção da estabilidade e segurança de sistemas computacionais. Em setores críticos, como o financeiro, as falhas ou vulnerabilidades podem ter consequências severas, como prejuízos financeiros, interrupção de serviços e comprometimento da experiência do usuário [1]. Nesse contexto, sistemas devem garantir alta disponibilidade, confiabilidade e segurança em todos os momentos.

Os *logs* são registros estruturados ou não estruturados que documentam o comportamento de sistemas ao longo do tempo, servindo como uma fonte essencial de dados para a identificação e a solução de falhas [2]. Eles armazenam eventos importantes, como mensagens de erro, alterações de configuração e acessos ao sistema, permitindo análises de causa raiz para problemas operacionais. No entanto, o crescimento exponencial do volume de *logs* gerados dificulta a análise manual, tornando a identificação de anomalias lenta e ineficiente.

Anomalias em registros de *logs* são definidas como padrões ou características que não seguem o comportamento esperado ou normal durante a operação do sistema [3]. Detectar e interpretar essas anomalias é crucial para prevenir falhas futuras e mitigar impactos operacionais. Contudo, os métodos tradicionais, como análise manual ou regras estáticas,

apresentam limitações significativas ao lidar com grandes volumes de dados e ambientes complexos.

O Aprendizado de Máquina (*Machine Learning* - ML) surge como uma alternativa eficiente, facilitando a automação da detecção de padrões anômalos em registros de eventos [4, 5]. Técnicas de ML não supervisionado, como *clustering* e detecção de *outliers*, permitem identificar anomalias sem a necessidade de rótulos prévios, acelerando o processo de diagnóstico e correção. No entanto, apenas detectar anomalias não é suficiente; é necessário interpretá-las e correlacioná-las com causas concretas.

Recentemente, a utilização de modelos avançados de Processamento de Linguagem Natural (*Natural Language Processing* - NLP) tem sido complementada por uma nova classe de algoritmos conhecida como Modelos de Linguagem de Grande Escala (*Large Language Model* - LLM). Modelos como o GPT (*Generative Pre-trained Transformer*), Claude e Gemini são capazes de analisar grandes volumes de texto, identificar relações semânticas e fornecer explicações detalhadas sobre eventos anômalos. Aplicados à análise de *logs*, esses modelos permitem:

- Interpretar contextos complexos de anomalias detectadas;
- Identificar as causas raiz de falhas ou comportamentos inesperados;
- Recomendar ações corretivas para evitar a recorrência do problema.

Essa abordagem integrada, combinando ML não supervisionado e interpretação utilizando LLMs, representa um avanço significativo na análise automatizada de *logs* e tem potencial para transformar processos críticos em sistemas de TI.

1.2 Definição do Problema

A análise de *logs* enfrenta desafios complexos devido às seguintes características:

- **Grande volume de dados:** Sistemas modernos produzem grandes volumes de registros diariamente, tornando inviável a análise manual em larga escala [6].
- **Falta de estruturação:** Os formatos variados e não padronizados dos *logs* dificultam a unificação e a interpretação.
- **Complexidade dos ambientes:** Sistemas distribuídos e interdependentes aumentam a dificuldade de correlação de eventos.
- **Dependência de análise humana:** A detecção de anomalias requer profissionais especializados, tornando o processo custoso e lento.

Portanto, o problema de pesquisa deste estudo pode ser resumido da seguinte forma:

Como integrar técnicas de ML não supervisionado e LLMs para automatizar a detecção e interpretação de anomalias em *logs* de sistemas de TI em ambientes críticos, como o setor financeiro?

1.3 Hipótese de Pesquisa

A hipótese central deste estudo é que a aplicação de técnicas de ML não supervisionado, combinadas com a interpretação de anomalias por meio de modelos de NLP, possibilita mapear e interpretar comportamentos anômalos como potenciais ameaças ou falhas em sistemas de TI de uma instituição financeira, automatizando e aprimorando a análise e mitigação desses eventos.

1.4 Justificativa e Delimitação do Problema

O tema deste estudo justifica-se, principalmente, pela relevância dos sistemas de TI no setor financeiro, onde a disponibilidade ininterrupta, a confiabilidade e a satisfação dos usuários são fatores críticos. A seguir, destacam-se os principais aspectos que fundamentam este trabalho:

- **Aumento da disponibilidade dos sistemas e redução de perdas:** A disponibilidade contínua dos sistemas é essencial para garantir operações estáveis. Qualquer interrupção pode acarretar prejuízos financeiros significativos e afetar diretamente a eficiência e a lucratividade das empresas.
- **Melhoria na confiabilidade:** A confiabilidade dos sistemas é uma exigência fundamental no setor financeiro, onde falhas podem comprometer a imagem da instituição. A aplicação de modelos de ML e NLP automatiza a identificação e a interpretação de anomalias, permitindo a prevenção ou mitigação de problemas recorrentes e aumentando a confiança nos sistemas.
- **Aumento da satisfação dos clientes:** A identificação eficiente da causa raiz de falhas contribui diretamente para a melhoria dos serviços oferecidos. A automação desses processos permite uma resposta mais rápida e precisa, beneficiando tanto os clientes internos quanto os externos.

Com base nessas perspectivas, este estudo concentra-se na aplicação de técnicas de ML não supervisionado e LLMs para detecção e interpretação de eventos anômalos em *logs* de sistemas de TI. A análise é conduzida em dois contextos distintos:

- **Log do Blue Gene/L supercomputer:** Conjunto de dados amplamente utilizado em estudos acadêmicos para validação de modelos [7]. Ele oferece um cenário robusto e bem documentado para testar as abordagens de detecção e interpretação de anomalias.
- **Log CCMEval do Microsoft Configuration Manager (MCM):** Utilizado em uma instituição financeira, este arquivo é fundamental para o monitoramento e avaliação do estado do agente do MCM [8]. A análise deste *log* representa um cenário prático, voltado para a mitigação de problemas reais em sistemas de TI críticos.

Ao delimitar o estudo nesses dois conjuntos de dados, busca-se avaliar a eficácia das técnicas propostas tanto em ambientes amplamente documentados (acadêmicos) quanto em contextos do mundo real, garantindo a robustez e a aplicabilidade prática dos resultados.

1.5 Objetivo Geral

Desenvolver uma abordagem automatizada para detecção e interpretação de anomalias em *logs* de sistemas de TI, utilizando técnicas de ML não supervisionado e LLMs.

1.5.1 Objetivos Específicos

- **Avaliar modelos de ML não supervisionado:** Comparar diferentes técnicas de detecção de anomalias no *log* BGL.
- **Aplicar a abordagem em cenários reais:** Analisar o *CCMEval.log* coletado de uma instituição financeira.
- **Interpretar anomalias com LLMs:** Fornecer explicações contextuais detalhadas sobre as anomalias detectadas.

1.6 Contribuição para a Pesquisa

Este estudo oferece contribuições relevantes:

- Proposta de uma metodologia integrada que combina ML não supervisionado e interpretação contextual utilizando LLMs.
- Validação da abordagem em cenários reais, utilizando *logs* de uma instituição financeira.

- Desenvolvimento de um fluxo automatizado para detecção e interpretação de anomalias, reduzindo o tempo e esforço na análise de *logs*.

1.7 Estruturação dos Capítulos

O restante deste trabalho está estruturado da seguinte forma. O Capítulo 2 contextualiza o trabalho e explora conceitos essenciais, enquanto o Capítulo 3 detalha a solução proposta. No Capítulo 4 são descritos os experimentos realizados, enquanto no Capítulo 5 são apresentados os resultados obtidos e conduzidas discussões sobre os mesmos. Por fim, o Capítulo 6 sintetiza as principais ideias e contribuições, aponta limitações e sugere direções para o futuro deste trabalho.

Capítulo 2

Fundamentação Teórica

Neste capítulo, são apresentadas a contextualização do trabalho e trabalhos de pesquisas relacionados ao tema. Na Seção 2.1, são explorados os conceitos fundamentais que servem de base para o desenvolvimento do trabalho, fornecendo uma base sólida para a compreensão do problema de pesquisa. Posteriormente, a Seção 2.2 apresenta os principais estudos relevantes, destacando as suas contribuições mais significativas para a formulação da solução proposta.

2.1 Contextualização

Esta seção fornece uma visão do contexto geral do trabalho, apresentando a importância e o cenário em que a detecção de anomalias em registros de *log* se insere. São abordados conceitos fundamentais relacionados à estrutura dos *logs*, destacando a presença de informações temporais e a natureza não estruturada dos dados. Além disso, é discutida a relevância da análise de *logs* na identificação de eventos e comportamentos anômalos em sistemas diversos.

2.1.1 Registros de *Log* e *Log Parsing*

Os registros de *log* consistem em dados geralmente não estruturados e que possuem um formato sequencial no tempo. Embora possam assumir várias formas, esses registros normalmente contêm um carimbo de data/hora (*timestamp*) e um *log* bruto (*raw log*). O *log* bruto descreve um evento registrado e geralmente é composto por duas partes distintas: uma constante (*log template*) e uma variável [9].

O *log parsing* é a etapa inicial do pré-processamento, onde os *logs* brutos são estruturados e os *templates* são extraídos. A Figura 2.1 mostra o resultado de um *log parsing* bruto. Neste exemplo, para identificar anomalias, o atributo utilizado para extrair o

Evento de log bruto:
1117839085 2005.06.03 R20-M1-N5-C:J17-U01 2005-06-03-15.51.25.712950 R20-M1-N5-C:J17-U01 RAS KERNEL INFO generating core.304

↓ Log parsing

Evento de log estruturado:	
LineId	2735
Timestamp	1117839085
Date	2005.06.03
Node	R20-M1-N5-C:J17-U01
Time	2005-06-03-15.51.25.712950
NodeRepeat	R20-M1-N5-C:J17-U01
Type	RAS
Component	KERNEL
Level	INFO
Content	generating core.304
EventId	070de4aa
EventTemplate	generating <*>
ParameterList	['core.304']

Figura 2.1: Exemplo de *Log parsing*

template foi a coluna ‘*Content*’, que contém a mensagem do evento de *log*. A coluna ‘*Content*’ representa o corpo principal da entrada de *log*, detalhando o evento específico que ocorreu. Isso resultou no template ‘*EventTemplate*’, que foi armazenado na coluna ‘*EventTemplate*’. O token ‘<*>’ é o coringa utilizado para substituir os parâmetros que são extraídos e incluídos na coluna *ParameterList*’.

2.1.2 Extração de Características

Na detecção de anomalias, são utilizadas diversas técnicas para a extração de características dos dados. A extração de características é uma etapa que visa a preparar os dados para análises subsequentes. Esta seção apresenta técnicas como o agrupamento de eventos, que aplica o conceito de janelamento para identificar padrões em sequências de *logs*. Também são abordadas as técnicas de Incorporação de Palavras (*Word Embedding*), como Word2Vec, GloVe e FastText, usadas para representar semanticamente o texto. Além disso, a técnica TF-IDF é discutida em relação à atribuição de pesos a termos, destacando sua aplicação no contexto do conjunto de dados.

Agrupamento de Eventos ou Janelamento

Essa técnica tem como objetivo identificar padrões incomuns que envolvem múltiplos eventos, como variações na sequência dos eventos ou correlações temporais nos registros de *log*. Nestas situações, é essencial organizar logicamente os eventos em grupos, muitas

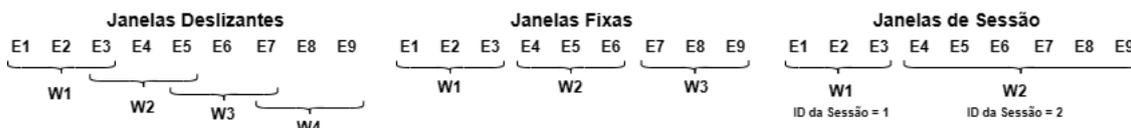


Figura 2.2: Tipos de janelamento

vezes chamados de sequências de *log* (*log sequences*), que serão analisados individualmente ou associados entre si [10].

Existem três tipos de janelamento na literatura: **Janela Deslizante**, em que janelas de uma duração específica se movem ao longo dos eventos de *log* com um tamanho de passo fixo; **Janela Fixa**, com o tamanho do passo igual ao tamanho da janela; e **Janela de Sessão**, que depende de um parâmetro de evento como um identificador de sessão.

A Figura 2.2 ilustra os três diferentes tipos de janelamento para os eventos de *log* [E1, E2, E3, E4, E5, E6, E7, E8, E9], resultando nas seguintes configurações de janelas: janelas deslizantes [W1, W2, W3, W4], janelas fixas [W1, W2, W3], e janelas de sessão [W1, W2] para as sessões com ID 1 e 2.

Incorporação de Palavras (*Word Embedding*)

No campo do NLP, a técnica de *Word Embedding* converte palavras em vetores de valores reais, permitindo que algoritmos interpretem o significado semântico e as relações entre palavras durante a análise de texto [11]. A obtenção de *word embeddings* geralmente envolve o treinamento de modelos de linguagem em extensos conjuntos de dados textuais. Essas representações simplificam a manipulação de texto por máquinas e melhoram o desempenho de modelos de ML em tarefas específicas.

TF-IDF

A técnica *Term Frequency-Inverse Document Frequency* (TF-IDF) é amplamente utilizada para atribuir pesos aos termos em várias aplicações, como mineração de texto e categorização de documentos [12]. Essa abordagem envolve atribuir pesos às palavras com base na frequência com que elas ocorrem em um documento específico (TF) e sua raridade em relação a todos os documentos analisados (IDF), destacando palavras que são frequentes em um documento específico, mas raras no conjunto total de documentos.

Vetorização Semântica

Técnicas tradicionais de detecção de anomalias em *logs* são limitadas, pois dependem de padrões conhecidos, o que as torna ineficazes para eventos novos. Zhang et al. [6] apontam dois desafios principais:

1. Evolução das mensagens de *log*: mudanças no código geram novos eventos, exigindo adaptação dos modelos..
2. Ruídos nos dados de *log*: problemas como mensagens ausentes ou duplicadas afetam a detecção de anomalias.

Para superar essas dificuldades, Zhang et al. propuseram a vetorização semântica, que representa eventos de *log* como vetores semânticos, permitindo uma detecção mais robusta. O fluxo envolve:

1. Pré-processamento dos Eventos de *Log*: Após o *Log Parsing*, os eventos passam por um pré-processamento para remover ruídos e elementos indesejados, usando técnicas de NLP, como a eliminação de *stop words*, caracteres especiais e números.
2. Ponderação dos Pesos das Palavras: Os pesos das palavras são calculados via TF-IDF, identificando a importância relativa de cada palavra nos eventos de *log* em relação ao conjunto de dados.
3. Vetorização das Palavras: As palavras são transformadas em vetores numéricos, permitindo representá-los em um espaço multidimensional.
4. Vetorização Semântica: Os pesos das palavras são combinados com seus vetores, formando um vetor semântico de dimensão fixa, que representa de forma significativa as informações dos eventos de *log*. A Equação 2.1 descreve o vetor semântico.

$$V = \frac{1}{N} \sum_{i=1}^N w_i \cdot v_i \quad (2.1)$$

Onde:

V : Vetor semântico

N : Número de *tokens* em uma sentença de evento de *log*

w : Peso da palavra calculado por TF \times IDF

v : Vetor da palavra

A vetorização semântica é realizada individualmente para cada janela de sequência de *log*, resultando em sequências de vetores semânticos que são posteriormente utilizadas para alimentar modelos de detecção de anomalias.

2.1.3 Detecção de Anomalias

A detecção de anomalias visa identificar eventos em conjuntos de dados que não seguem o padrão esperado de comportamento normal. Essa técnica abrange uma variedade de aplicações, incluindo a detecção de fraudes bancárias e a prevenção de intrusões em sistemas, entre outras. Eventos inesperados, chamados de anomalias, podem ser categorizados em três grupos principais [13]: **anomalia pontual**, que se refere a uma entrada de dado isolada que é significativamente diferente do restante; **anomalia contextual**, que ocorre em um contexto específico, tornando um registro anômalo dentro daquele contexto mais amplo; e **anomalia coletiva**, que envolve grupos de registros que, embora não sejam individualmente anômalos, se destacam em conjunto devido a características distintas quando comparados ao restante dos dados. A seguir, serão apresentados os principais conceitos, técnicas e soluções aplicadas para a detecção de anomalias, destacando a importância do ML nesse contexto.

Aprendizado de Máquina

Aprendizado de Máquina ou *Machine Learning* (ML) é um ramo da Inteligência Artificial (IA) que se concentra no desenvolvimento de algoritmos capazes de aprender padrões e realizar tarefas sem a necessidade de programação explícita, possuindo a habilidade de melhorar o desempenho na execução de certas tarefas por meio da experiência [4] [5]. As técnicas de ML são classificadas em quatro abordagens: **supervisionada** (treinamento com dados rotulados), **não supervisionada** (identificação de padrões sem rótulos), **semi-supervisionada** (combinação de dados rotulados e não rotulados) e **reforço** (decisões baseadas em recompensas) [14].

Para a detecção de anomalias, [15] identificou 28 técnicas de ML, divididas em seis categorias (classificação, *ensemble*, otimização, sistema de regras, *clustering* e regressão). Dentre essas, as seguintes abordagens não supervisionadas foram selecionadas para avaliação neste trabalho devido ao seu amplo uso e eficácia demonstrada:

- **Local Outlier Factor (LOF)**, proposto por [16], é uma técnica de análise de dados que identifica anomalias (*outliers*) comparando a densidade de um ponto de dados com a densidade de seus vizinhos próximos. Sua abordagem centrada em propriedades locais permite identificar anomalias em áreas específicas dos dados, embora sua interpretação exija atenção devido à sensibilidade a parâmetros, como o valor de k ;

- **Autoencoder (AE)**, inicialmente proposto por [17], é uma rede neural não supervisionada com um codificador (*encoder*) e um decodificador (*decoder*). O codificador comprime os dados de entrada em uma representação latente de menor dimensionalidade, e o decodificador os reconstrói. O treinamento minimiza o erro de reconstrução, ajudando

o modelo a aprender características relevantes. Na prática, o codificador gera a representação latente, e o decodificador reconstrói os dados, tornando os *autoencoders* eficazes para detecção de anomalias ao destacar desvios na reconstrução;

- **Self-Organizing Map (SOM)**, também conhecida como rede de Kohonen [18], é um tipo de rede neural que projeta dados de alta dimensionalidade em uma grade de baixa dimensionalidade, simplificando a visualização. É treinada usando aprendizado competitivo não supervisionado e é notável por sua eficiência no tratamento de grandes conjuntos de dados não lineares, preservando as propriedades topológicas do espaço de entrada;

- **Isolation Forest (IF)**, proposto por [19], é um algoritmo baseado em árvores de decisão que detecta anomalias isolando pontos de dados *outliers* através do particionamento aleatório dos dados. Sua escalabilidade o torna adequado para grandes conjuntos de dados;

- **Minimum Covariance Determinant (MCD)** [20] é um algoritmo robusto para estimar a matriz de covariância. Ele identifica subconjuntos de dados menos suscetíveis a *outliers*, minimizando a covariância entre as observações e removendo essas observações anômalas;

- **Histogram-based Outlier Score (HBOS)** [21] usa histogramas univariados para identificar *outliers*. É rápido, fácil de interpretar e versátil, adaptando-se bem a diferentes tipos de dados;

- **Cluster-based Local Outlier Factor (CBLOF)** [22] constrói sobre o LOF incorporando a estrutura de *clusters* dos dados. Isso permite que o CBLOF seja eficaz na identificação de anomalias dentro dos grupos formados, sendo particularmente útil para detecção de anomalias em dados agrupados;

- **Gaussian Mixture Model (GMM)** [23] é um modelo probabilístico que descreve a distribuição de probabilidade como uma mistura de múltiplas distribuições gaussianas (normais). Ele combina densidades de diferentes componentes e é útil para modelar dados complexos e identificar padrões.

2.1.4 Interpretação de Anomalias com NLP

A interpretação de anomalias desempenha um papel importante na operação de sistemas complexos, especialmente em ambientes sensíveis, como os de instituições financeiras. Além da detecção dessas anomalias, torna-se necessário compreendê-las para que suas causas possam ser identificadas e tratadas de maneira adequada. Modelos de NLP apresentam-se como uma abordagem promissora ao permitir a geração de explicações interpretáveis.

Os LLMs são treinados utilizando uma abordagem probabilística para gerar texto com base em grandes e diversificados conjuntos de dados de treinamento. Esses modelos focam na previsão da probabilidade da próxima palavra, dada uma sequência anterior de palavras, utilizando um processo auto-regressivo. Nesse processo, os *tokens* são previstos um a um até que um *token* de fim designado seja alcançado. LLMs são particularmente poderosos em sua capacidade de generalizar em diversos contextos, graças ao grande número de parâmetros e à variedade dos corpora de treinamento. Isso os torna eficazes em lidar com uma ampla gama de tarefas de linguagem natural, desde resposta a perguntas (*question answering*) e tradução até sumarização e outras tarefas mais especializadas [24]. A seguir, são apresentados três dos principais modelos disponíveis atualmente: GPT, Gemini e Claude.

GPT

O lançamento do modelo de linguagem GPT representou um marco na história da IA, tendo sua trajetória originada no artigo “*Attention is all you need*” [25]. O GPT-1 foi o primeiro passo importante, utilizando apenas o *decoder* da arquitetura *Transformer* para prever o próximo *token* com base no contexto anterior. Uma inovação importante no GPT-1 foi o uso da atenção mascarada no *decoder*, o que permitiu ao modelo gerar previsões sequenciais com mais eficiência. No entanto, apesar de seu potencial, o GPT-1 ainda apresentava limitações em sua capacidade de generalização para tarefas sem treinamento prévio.

O GPT-4, versão mais atual do GPT, trouxe inovações notáveis, especialmente por suas capacidades multimodais, que permitem processar entradas de texto e imagem simultaneamente. Isso ampliou consideravelmente suas aplicações, como na análise de documentos que combinam texto e gráficos. Além disso, o GPT-4 foi desenvolvido com uma arquitetura aprimorada e técnicas de alinhamento pós-treinamento, resultando em melhorias na factualidade e no comportamento do modelo. Seu desempenho em testes acadêmicos e profissionais também foi impressionante, superando modelos anteriores em diversas áreas. O desenvolvimento do GPT-4 também foi marcado pela criação de uma infraestrutura de DL com escalabilidade previsível. Essa infraestrutura permitiu que os engenheiros previssem seu desempenho com precisão, mesmo em versões menores durante o desenvolvimento. A OpenAI também fez avanços importantes em segurança, realizando testes adversariais para mitigar riscos como desinformação e viés, além de implementar um pipeline de segurança que reduz o potencial de respostas prejudiciais [26].

Gemini

O Gemini é uma família de modelos multimodais altamente avançados desenvolvidos pela Google, que apresenta notáveis capacidades em tarefas que envolvem a compreensão de imagem, áudio, vídeo e texto. Dividido em três variantes principais — Ultra, Pro e Nano — cada uma é projetada para atender diferentes requisitos de aplicações, desde tarefas de raciocínio complexo até o uso em dispositivos com restrições de memória. A avaliação do Gemini Ultra em uma ampla gama de *benchmarks* demonstra que este modelo alcança o estado da arte em 30 dos 32 *benchmarks* analisados, destacando-se especialmente por ser o primeiro modelo a atingir desempenho equivalente ao de especialistas humanos no MMLU, um *benchmark* focado em testes de conhecimento e raciocínio [27].

A arquitetura do Gemini é baseada em *Transformers* aprimorados, com otimizações voltadas para treinar e inferir eficientemente em grande escala. Os modelos suportam uma janela de contexto de 32k *tokens* e utilizam mecanismos de atenção eficientes, como a *multi-query attention*, para melhorar o desempenho. O Gemini também se destaca por ser nativamente multimodal, processando de forma integrada entradas textuais, visuais, de áudio e vídeo, com a capacidade de gerar saídas em texto e imagens. Essa arquitetura permite que o modelo realize raciocínio complexo em diferentes modalidades e utilize sinais temporais para entender sequências de vídeo e áudio [27].

Claude

Claude é um modelo de linguagem desenvolvido pela Anthropic, uma empresa de IA fundada por ex-funcionários da OpenAI. O modelo Claude é projetado para ser seguro e alinhado com os interesses humanos, incorporando pesquisas da Anthropic sobre a mitigação de vieses e a melhoria da segurança de grandes modelos de linguagem. A abordagem da Anthropic é baseada em princípios como a supervisão cuidadosa e a tentativa de criar sistemas que sigam normas de comportamento alinhadas com os valores humanos. Assim como outros modelos de linguagem, Claude pode ser usado para responder perguntas, gerar texto, auxiliar em tarefas de escrita, e muitos outros casos de uso. A Anthropic tem se focado bastante em desenvolver técnicas que ajudem a reduzir respostas indesejadas ou prejudiciais, enfatizando a construção de uma IA mais confiável e ética [28].

Com o lançamento do Claude 3.5 Sonnet, a Anthropic introduziu melhorias significativas em desempenho em tarefas de raciocínio, codificação e processamento visual. O Claude 3.5 Sonnet se destacou em benchmarks importantes como GPQA, MMLU e HumanEval, que testam habilidades avançadas de raciocínio e codificação, resolvendo 64% dos problemas em uma avaliação interna de codificação agentic. O Claude 3.5 Sonnet também passou por rigorosos testes de segurança, incluindo áreas como *Chemical*, *Bio-*

logical, Radiological, and Nuclear (CBRN) e cibersegurança, sendo classificado como de Nível de Segurança de IA 2 (ASL-2), o que garante que ele não apresenta riscos de danos catastróficos [29].

Engenharia de *Prompt* para LLMs

Um *prompt* é uma entrada textual que orienta um modelo de linguagem sobre uma tarefa a ser executada, fornecendo as instruções e o contexto necessários para gerar uma resposta. Embora possa assumir outros formatos, como áudio, é mais comumente transcrito em texto para processamento por LLMs. A função principal do *prompt* é guiar o modelo, garantindo que ele compreenda a tarefa e gere uma resposta precisa com base nas instruções fornecidas [30].

A engenharia de *prompts* refere-se ao desenvolvimento e otimização de *prompts* para maximizar o desempenho dos LLMs em diferentes tarefas. Isso envolve ajustar e testar várias formas de instruções para garantir que o modelo entenda corretamente a tarefa e produza a melhor saída possível. O sucesso de um LLM está diretamente ligado à qualidade da estruturação do *prompt*, tornando essa atividade crucial para explorar o pleno potencial desses modelos [31]. Na engenharia de *prompts*, diversas técnicas têm sido desenvolvidas para lidar com a complexidade crescente das tarefas atribuídas aos LLMs, das quais destacamos as seguintes:

- *Zero-shot prompting*: O modelo recebe apenas uma descrição da tarefa, sem exemplos prévios. Essa técnica explora o conhecimento geral do modelo com base em seus dados de treinamento. Embora simples, ela depende bastante da capacidade do modelo de compreender a tarefa a partir da descrição fornecida [30].
- *Few-shot prompting*: Aqui, o modelo recebe alguns exemplos junto com a descrição da tarefa, o que melhora seu desempenho em tarefas mais complexas. Essa técnica oferece ao LLM um contexto mais claro, mostrando exemplos de entradas e saídas desejadas [30].
- *Chain-of-thought (CoT) prompting*: Essa técnica orienta o modelo a seguir um raciocínio passo a passo antes de fornecer a resposta final. Ao estruturar o raciocínio lógico, o CoT é especialmente eficaz em tarefas que envolvem dedução e resolução de problemas, como questões matemáticas e de lógica [32].

Além dessas técnicas, novas abordagens complementares têm sido desenvolvidas para melhorar ainda mais o desempenho dos LLMs em tarefas específicas. Métodos como *Self-Consistency*, que busca otimizar a coerência das cadeias de raciocínio, e *Tree-of-Thoughts (ToT)*, que organiza pensamentos intermediários em uma estrutura de árvore

para lidar com raciocínios mais complexos, são exemplos de inovação. Outras técnicas, como *Graph-of-Thoughts (GoT)* e *Chain-of-Verification (CoVe)*, permitem que o modelo explore e verifique múltiplas cadeias de raciocínio para garantir robustez e precisão nas respostas. Essas abordagens têm mostrado melhorias significativas em *benchmarks* de raciocínio lógico e tomada de decisão [33].

Neste trabalho, a técnica de *Chain-of-Thought (CoT) prompting* foi escolhida por sua eficácia em orientar o modelo a realizar análises lógicas detalhadas e organizadas. Ao aplicar o CoT à análise de *logs* e à interpretação de anomalias, garante-se que o modelo siga um raciocínio estruturado, analisando cada parte do problema de forma sequencial. Isso é essencial para identificar falhas operacionais e ameaças potenciais com precisão, como demonstrado em estudos que destacam a superioridade do CoT em tarefas de raciocínio complexo [32].

2.2 Trabalhos Relacionados

A literatura apresenta diversos trabalhos referentes ao uso de modelos de detecção de anomalias em *logs*. Primeiramente, o trabalho de Du et al. [2] se destaca por desenvolver o DeepLog, um modelo de DL utilizando *Long Short-Term Memory (LSTM)* para modelar um sistema de *log* como uma sequência de linguagem natural. O modelo aprende e codifica a mensagem de *log* inteira, incluindo *timestamp*, *log key* e valores de parâmetros. Dessa forma, o DeepLog aprende automaticamente padrões de *logs* de execução normal e detecta anomalias quando esses padrões se afastam dos modelos treinados com os dados do *log* sob execução normal. Além disso, o modelo incorpora *feedback* do usuário, permitindo o treinamento/atualização incremental que se adapta a novos padrões de execução ao longo do tempo. No entanto, a ausência de interpretação das anomalias limita a aplicabilidade do modelo em cenários que demandam análises mais detalhadas.

Debnath et al. [34] desenvolveram o LogLens, um sistema de análise em tempo real que automatiza o processo de detecção de anomalias em *logs* com pouco ou nenhum conhecimento prévio das especificações do usuário e do sistema de destino. Ele utiliza técnicas de aprendizado de máquina não supervisionadas para descobrir padrões nos dados de aplicação, aproveitando esses padrões para realizar a análise em tempo real e projetar soluções avançadas de monitoramento. Apesar disso, o modelo não fornece explicações interpretáveis das anomalias detectadas, o que pode dificultar a identificação das causas subjacentes.

O trabalho de Zhang et al. [6] apresenta o LogRobust, um modelo projetado para extrair informações semânticas de eventos de *log*, representando-os como vetores semânticos. Utilizando um modelo Bi-LSTM baseado em mecanismos de atenção, o sistema

detecta anomalias ao capturar as informações contextuais das sequências de *log* e atribuir automaticamente a importância relativa a diferentes eventos. Com essa abordagem, o LogRobust demonstra robustez em dados instáveis e dinâmicos, proporcionando uma detecção de anomalias precisa e confiável. No entanto, o modelo é supervisionado e foca apenas na detecção, sem abordar a interpretação das anomalias, o que limita sua aplicabilidade prática.

Proposto por Huang et al. [35], o HitAnomaly é um modelo de detecção de anomalia baseado em *log* que utiliza a estrutura de transformador hierárquico para modelar sequências de *templates* e valores de parâmetros. Nele é projetado um codificador de sequência de registro e um codificador de valor de parâmetro para obtenção de suas representações correspondentes. Por fim, utilizam um mecanismo de atenção como modelo de classificação final. O modelo consegue capturar informações semânticas nas sequências de *log* e nos valores de parâmetros para lidar com vários tipos de anomalias. Contudo, o HitAnomaly não oferece uma interpretação detalhada das anomalias detectadas.

Studiawan et al. [36] propõem uma nova técnica de análise de sentimentos baseada em DL para verificar se há anomalias em *logs* de sistema operacional (SO). Nesse modelo, as mensagens de registro são modeladas como sentenças para identificação dos sentimentos usando as Redes de Unidades Recorrentes Fechadas (*Gated Recurrent Unit* - GRU). Os conjuntos de dados utilizados têm um grande desbalanceamento de classe, pois a quantidade de sentimentos negativos é bem menor que a de sentimentos positivos. Para resolver isso, eles utilizaram o método de *Tomek Link* para balancear as classes e então obter um modelo mais preciso. A técnica demonstra uma abordagem inovadora na análise de sentimentos, porém é restrita à análise de sentimentos e tem foco limitado em classes minoritárias.

O LAnoBERT proposto por Lee et al. [37] trata-se de um sistema de detecção de anomalias em *logs* que utiliza o modelo BERT, apresentando excelente desempenho em tarefas de NLP. O modelo aprende o contexto de dados de *log* normais por meio do modelo de Linguagem Mascarada (*Masked Language Model* - MLM), prosseguindo com a detecção baseada em aprendizagem não supervisionada. No entanto, o LAnoBERT também não oferece explicações interpretáveis para as anomalias identificadas.

No estudo recente de Nam et al. [38], o uso do BERT-CNN foi investigado para a análise de *logs* com o objetivo de prever falhas em Máquinas Virtuais (*Virtual Machines* - VM). O modelo proposto utiliza o BERT para extrair *embeddings* das sentenças de cada *log*, que são posteriormente inseridos em uma rede neural convolucional (*Convolutional Neural Network* - CNN) pré-treinada. Essa abordagem alcança alta precisão na predição de falhas, mesmo em situações não vistas durante o treinamento. Porém, o modelo é restrito a VMs e carece de explicações interpretáveis sobre as falhas detectadas.

Além dos trabalhos apresentados acima, considerado os mais importantes para a realização deste trabalho, também foram encontradas outros trabalhos relevantes: [39] *Tools and benchmarks for automated log parsing* que traz um comparativo de *log parsers*, [40] modelo Swisslog que se mostrou robusto para logs em processo de mudança e eficaz para diversas falhas, [41] modelo *Log Clustering* que foi utilizado na manutenção de muitos sistemas de serviço online da Microsoft, [42] DILAF que é um *framework* para análise distribuída de sistemas de larga escala para detecção de anomalias e [43] *Deep Learning for Anomaly Detection: Survey* que traz uma revisão do estado da arte sobre detecção de anomalias utilizando DL.

Um resumo comparativo das contribuições e limitações dos principais trabalhos mencionados é apresentado na Tabela 2.1.

Tabela 2.1: Comparação dos trabalhos relacionados

Trabalho	Abordagem	Contribuição	Limitação
DeepLog (Du et al., 2017)	LSTM	Detecção automatizada de anomalias com aprendizado incremental e adaptativo.	Ausência de interpretação das anomalias.
LogLens (Debnath et al., 2018)	Não supervisionado	Detecção em tempo real com pouco conhecimento prévio e baixo custo computacional.	Ausência de interpretação das anomalias.
LogRobust (Zhang et al., 2019)	Bi-LSTM + Atenção	Robustez em dados instáveis e dinâmicos; captura do contexto das sequências de <i>logs</i> .	Necessita de rótulos prévios; ausência de interpretação das anomalias.
HitAnomaly (Huang et al., 2020)	Transformador Hierárquico	Modelagem multigranular de <i>logs</i> com <i>templates</i> e parâmetros.	Ausência de interpretação das anomalias.
GRU-Sentimentos (Studiawan et al., 2020)	GRU + Balanceamento	Uso inovador de análise de sentimentos em mensagens de <i>logs</i> .	Restrito à análise de sentimentos; foco em classes minoritárias; ausência de interpretação das anomalias.
LAnoBERT (Lee et al., 2021)	BERT	Representação semântica de <i>logs</i> com excelente desempenho em NLP.	Ausência de interpretação das anomalias.
BERT-CNN (Nam et al., 2022)	BERT + CNN	Alta precisão na predição de falhas em Máquinas Virtuais (VMs).	Restrito a VMs; ausência de interpretação das anomalias.

Embora os avanços significativos na detecção de anomalias tenham sido demonstrados, observa-se que o foco principal tem sido a identificação dessas anomalias, com pouca ênfase na sua interpretação. A maioria dos estudos concentra-se em melhorar a precisão dos modelos, sem oferecer uma análise detalhada das causas das anomalias detectadas, o que limita sua utilidade prática em cenários operacionais complexos. Essa limitação torna-se especialmente crítica em ambientes reais, como o de instituições financeiras, onde *logs* de sistemas bancários críticos exigem não apenas uma detecção precisa, mas também interpretações detalhadas para que as equipes técnicas possam agir de forma rápida e eficaz na resolução de problemas. A ausência de explicações interpretáveis reduz o potencial de uso dos modelos em sistemas financeiros, que demandam alta confiabilidade e clareza na análise de falhas, situações em que qualquer incerteza pode ocasionar atrasos na resolução ou até mesmo interrupções de serviços críticos, resultando em prejuízos operacionais e financeiros significativos.

Este trabalho diferencia-se ao focar tanto na detecção quanto na interpretação das anomalias, utilizando modelos de aprendizado não supervisionado em cenários reais para detecção e LLMs para interpretação. A abordagem foi aplicada a diferentes tipos de *logs*, incluindo os do sistema *Blue Gene/L* e os do *Microsoft Configuration Manager*, estes últimos coletados diretamente de ambientes de produção de uma instituição financeira. Além de detectar anomalias, são oferecidas explicações interpretáveis, geradas a partir de modelos de interpretação baseados em LLMs, que facilitam a compreensão e a resolução dos problemas identificados. Ao unir detecção e interpretação, este estudo contribui para uma análise de *logs* mais completa e aplicável a sistemas críticos, como os do setor bancário, nos quais a confiabilidade e a clareza das anomalias são fundamentais para a tomada de decisão rápida e precisa. Assim, a pesquisa não apenas aprimora a capacidade de detecção e interpretação de anomalias, mas também abre novas perspectivas para o uso de aprendizado não supervisionado em ambientes críticos, oferecendo uma base sólida para futuras pesquisas e implementações em diversos setores.

Capítulo 3

Proposta

Neste capítulo, é apresentada a solução elaborada e desenvolvida ao longo da realização do mestrado, incluindo os detalhes do sistema de detecção e interpretação de anomalias em eventos de *log* em sistemas de TI.

3.1 Fluxo de Trabalho

O fluxo de trabalho proposto para a solução emprega técnicas consolidadas de pré-processamento, extração de características e interpretação de anomalias, com base em uma revisão abrangente da literatura relacionada à detecção de anomalias em eventos de *log*. A Figura 3.1 apresenta a estrutura global do sistema, que se subdivide essencialmente em sete etapas distintas. Cada uma dessas etapas será detalhada nas subseções subsequentes.

3.1.1 Coleta de *Logs*

A operação da solução inicia na fase de Coleta de *Logs*, na qual é realizada a coleta distribuída de *logs* dos sistemas para identificar anomalias. A coleta pode ser realizada de várias maneiras, mas a abordagem mais comum envolve a instalação de um agente coletor em cada máquina. Esses agentes são responsáveis por enviar os registros desejados para um servidor de armazenamento, onde ocorre a agregação dos *logs* (Agregação de *Logs*). Essa agregação serve para alimentar a base de dados dos modelos. É importante destacar que, em nossa arquitetura, os novos *logs* usados para testes não são armazenados, apenas os *logs* utilizados no treinamento dos modelos são retidos.

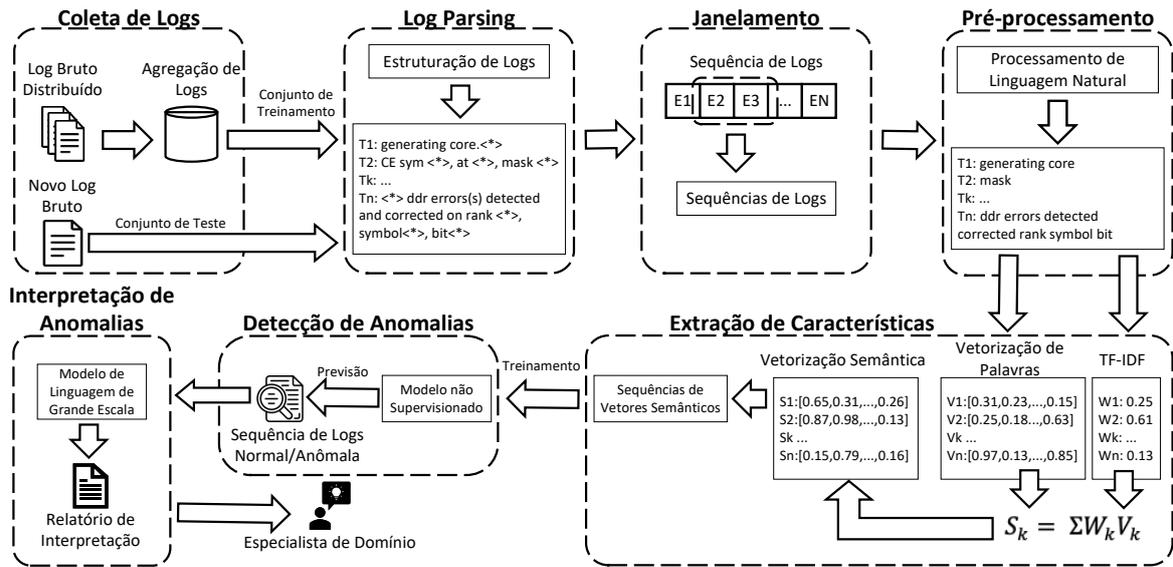


Figura 3.1: Fluxo de Detecção e Interpretação de Anomalias em Logs

3.1.2 Log Parsing

Após a coleta dos logs (Coleta de Logs), a próxima etapa é o *Log Parsing*, no qual os dados de logs são estruturados para facilitar sua análise e interpretação. Essa etapa é essencial, pois permite a extração de *templates* das mensagens de eventos, o que proporciona uma representação mais organizada e padronizada dos eventos registrados. Entre as várias técnicas de *Log Parsing* disponíveis, duas se destacam na literatura: Drain e Spell [39]. Neste trabalho, foi adotado o método Drain, amplamente reconhecido e utilizado em estudos acadêmicos.

O Drain é um analisador de registros de eventos em tempo real que se distingue por sua habilidade em converter uma grande quantidade de registros de maneira contínua e precisa, operando em tempo real. Esse método é baseado em uma árvore de análise com profundidade fixa e oferece vantagens substanciais. Quando uma mensagem de registro bruto chega, o Drain processa-a através da aplicação de expressões regulares simples, baseadas no conhecimento do domínio. A característica mais inovadora do Drain reside na sua capacidade de agrupar esses registros em categorias específicas, conhecidas como *clusters*, simplificando, assim, a interpretação e extração de informações relevantes dos registros [44].

Essa abordagem demonstra particular relevância em ambientes onde o volume de registros de eventos cresce rapidamente. Métodos tradicionais de análise *offline* podem se tornar ineficazes e morosos em tal contexto, tornando o Drain uma escolha estratégica para empresas e organizações que buscam realizar o *log parsing* de eventos em tempo real mais eficaz e precisa. A habilidade do Drain de processar registros de eventos em

fluxo contínuo e realizar uma categorização eficiente dos dados o torna uma ferramenta de inestimável valor em diversos domínios, que vão desde a segurança cibernética até o monitoramento de sistemas e a resolução de problemas.

Conforme destacado no estudo de Zhu et al. [39], o Drain se consolida como uma opção sólida e eficaz para o *log parsing* de eventos em tempo real, sendo uma escolha confiável para profissionais que buscam soluções de análise de registros de eventos mais eficazes e eficientes em ambientes dinâmicos e em constante expansão.

3.1.3 Janelamento

A etapa subsequente, conhecida como Janelamento, executa a extração de sequências de *logs* mediante o uso de janelas temporais. Isso permite a configuração do tamanho da janela (*window size*) e do tamanho do passo (*step size*). Quando o tamanho do passo é configurado para ser igual ao tamanho da janela, ocorre a extração de sequências utilizando janelas fixas, conforme anteriormente explicado no Capítulo 2.

A escolha de utilizar exclusivamente janelas fixas ou deslizantes em detrimento de janelas de sessão baseia-se na natureza homogênea dos dados de registros de *log*, que carecem de identificadores claros para definição de sessões. Além disso, a maior simplicidade computacional e a adaptabilidade a padrões temporais gerais oferecidas por janelas fixas ou deslizantes contribuem para uma abordagem eficiente e interpretável na detecção de anomalias.

3.1.4 Pré-Processamento

Após a conclusão da fase de Janelamento, inicia-se a etapa de Pré-Processamento. Nesse estágio, o pré-processamento das mensagens de *log* desempenha um papel fundamental na preparação de dados para análise e mineração de *logs*. A qualidade dos dados de *log* influencia diretamente a eficácia das análises subsequentes e a detecção de problemas em sistemas e aplicativos. Neste trabalho, o pré-processamento das mensagens segue as seguintes etapas:

1. **Remoção de Caracteres Indesejados:** A primeira etapa do pré-processamento consiste em eliminar caracteres que não são letras do alfabeto. Isso ajuda a limpar o texto e a concentrar-se nas palavras-chave relevantes contidas nas mensagens dos eventos de *log*.
2. **Remoção de *Stop Words*:** *Stop words* são palavras comuns que geralmente não contribuem significativamente para o significado do texto. Essas palavras são removidas para reduzir o ruído nos eventos de *log*.

3. **Remoção de Palavras Curtas:** Além da remoção de *stop words*, palavras muito curtas (com menos de 3 caracteres) também são eliminadas, uma vez que tendem a ter pouco valor semântico.
4. **Separação de *Tokens* Compostos:** Mensagens de *log* frequentemente incluem *tokens* compostos em *Camel Case*, como “*UserName*”. A separação desses *tokens* ajuda a tornar o texto mais legível e a capturar palavras individuais.
5. **Normalização com *Stemming*:** A normalização envolve reduzir as palavras às suas formas básicas (raízes). O *stemming* é uma técnica que realiza essa tarefa, ajudando a agrupar palavras relacionadas com base em sua raiz. A normalização com *stemming* foi implementada como opcional, mas não foi utilizada, pois não apresentou melhoria significativa nos resultados.
6. **Lematização:** A lematização é outra técnica de normalização que reduz as palavras a sua forma de dicionário, o “lema”. Diferente do *stemming*, ela preserva o significado original das palavras, o que garante maior precisão. Por esse motivo, foi adotada neste trabalho para assegurar maior consistência na análise dos dados.

Essas etapas de pré-processamento visam refinar os eventos de *log* para garantir que as análises subsequentes sejam mais precisas e relevantes. A flexibilidade do código permite escolher quais dessas etapas aplicar com base nas necessidades específicas de cada *log*. Por fim, o código garante que, se não houver evento de *log* após o pré-processamento, ele retorne ‘OOV’ (*Out-of-Vocabulary*) para palavras que não tem representação ou ainda para indicar a ausência de dados.

3.1.5 Extração de Características

Após a conclusão da etapa de Pré-Processamento, a próxima fase é a Extração de Características. Este processo visa capturar a semântica das mensagens de eventos de *log*, transformando-as em sequências de vetores semânticos. Essa abordagem torna o sistema mais resiliente a instabilidades ou ruídos nas mensagens. Essa etapa é essencialmente composta por três partes: vetorização das palavras, extração de pesos das palavras usando o algoritmo TF-IDF e, por fim, a combinação dos pesos e vetores para criar os vetores semânticos e em seguida as sequências de vetores semânticos. Mais informações detalhadas sobre a extração de características podem ser encontradas no Capítulo 2 deste trabalho.

As técnicas de Incorporação de Palavras ou *Word Embedding* têm se destacado na representação semântica de linguagem natural. Diversas abordagens estão disponíveis, sendo abordadas neste trabalho as mais conhecidas na literatura: Word2Vec [45], GloVe [46] e FastText [47].

O **Word2Vec**, uma técnica de incorporação de palavras popular desenvolvida por Mikolov et al. em 2013, baseia-se na arquitetura de redes neurais que mapeia palavras em vetores densos dentro de um espaço semântico. Esses vetores refletem a similaridade semântica entre as palavras, melhorando a capacidade dos modelos de linguagem em compreender as relações e o contexto das palavras.

O **GloVe**, ou Vetores Globais para Representação de Palavras, introduzido por Pennington et al. em 2014, é uma técnica de incorporação de palavras que combina aspectos de contagem global de palavras e aprendizado de máquina. Ele utiliza a coocorrência de palavras em um corpus de texto para criar representações vetoriais que capturam informações semânticas e sintáticas.

O **FastText**, introduzido pela pesquisa do Facebook em 2016, é uma extensão do Word2Vec que se concentra em n-gramas de caracteres. É especialmente útil para lidar com palavras raras e idiomas com morfologia complexa. O FastText gera representações vetoriais de subpalavras, tornando-se uma escolha robusta para tarefas de processamento de linguagem natural, como classificação de texto e tradução automática.

Para a vetorização das palavras neste trabalho, foram avaliados modelos pré-treinados de *word embedding* com diferentes dimensões (50d, 100d, 200d, 300d). Após testes preliminares, optou-se por utilizar o modelo GloVe com 50 dimensões (50d), já que os outros modelos não demonstraram ganhos significativos em relação à performance. A escolha pelo modelo GloVe de 50 dimensões se fundamenta não apenas na simplicidade computacional, que resultou em uma execução mais eficiente, mas também em sua reconhecida eficácia na literatura. Dessa forma, a extração de características se torna uma parte crucial do sistema, garantindo que as mensagens dos eventos de *log* sejam representadas de forma mais significativa e robusta para análises posteriores.

A utilização do algoritmo TF-IDF neste processo é motivada por evidências empíricas e pesquisas anteriores, como apresentado no trabalho de Zhang [6]. Em ambientes nos quais a estabilidade das mensagens de eventos de *log* pode variar significativamente, o TF-IDF oferece uma vantagem ao atribuir pesos a palavras com base em sua relevância no contexto de um documento específico. Essa abordagem compensa possíveis flutuações na distribuição de termos ao longo do tempo, melhorando assim a resiliência do sistema a variações e ruídos nos dados de *log*.

3.1.6 Detecção de Anomalias

A etapa de Detecção de Anomalias envolve a classificação de uma sequência de mensagens de *log* como “normal” ou “anômala”. Isso é realizado por meio do treinamento dos modelos com a base de dados coletada na fase de Coleta de *Logs* e da predição usando novos registros de *log*. No contexto deste trabalho, optou-se por empregar exclusivamente modelos

não supervisionados. Essa escolha se deve à complexidade de obter rótulos de classificação para eventos em ambientes do mundo real, onde a imprevisibilidade e a diversidade dos eventos são características comuns.

Além disso, este trabalho foi desenvolvido para a detecção de anomalias contextuais, uma escolha que se alinha à necessidade de considerar a interdependência temporal entre eventos em sequências de *logs*. A interpretação de um evento específico pode estar intrinsecamente relacionada aos eventos que o antecedem ou sucedem, e a detecção contextual busca capturar essa complexidade, avaliando o contexto mais amplo em que esses eventos ocorrem.

A abordagem supervisionada, que geralmente requer dados rotulados, mostra-se desafiadora devido à escassez e ao custo associado a esses dados, especialmente em ambientes em constante evolução. A opção por modelos não supervisionados visa superar tais desafios, permitindo uma detecção mais flexível de anomalias sem depender explicitamente de exemplos rotulados. Estes modelos se destacam ao se adaptar à complexidade e variabilidade presentes nos ambientes do mundo real, oferecendo assim uma solução eficaz diante das dificuldades em obter rótulos precisos.

Para a detecção de anomalias, foram escolhidas as técnicas: *Local Outlier Factor* (LOF) [16], *Autoencoder* (AE) [17], *Self-Organizing Map* (SOM) [18], *Isolation Forest* (IF) [19], *Minimum Covariance Determinant* (MCD) [20], *Histogram-based Outlier Score* (HBOS) [21], *Cluster-Based Local Outlier Factor* (CBLOF) [22] e *Gaussian Mixture Model* (GMM) [23].

Essa escolha baseou-se na ampla adoção dessas técnicas na literatura especializada. Além disso, a intenção é realizar uma comparação de seus desempenhos na detecção de anomalias. A diversidade dessas abordagens proporciona uma avaliação abrangente, permitindo a compreensão de como diferentes métodos se comportam na identificação de padrões incomuns. Essa fase serve como base para a etapa de interpretação das sequências anômalas, permitindo uma análise mais profunda e contextual das causas subjacentes aos comportamentos anômalos identificados.

3.1.7 Interpretação de Anomalias

Após a detecção das sequências anômalas, a etapa final é a interpretação dessas sequências. O principal objetivo desta fase é fornecer uma análise detalhada e contextual das anomalias, auxiliando na compreensão das causas subjacentes e propondo ações corretivas para prevenir sua reincidência. Historicamente, o mapeamento de anomalias para ameaças, problemas e falhas tem sido desafiador, exigindo a análise de profissionais especializados. Esses profissionais são responsáveis por interpretar as anomalias e avaliar suas consequências no sistema. No entanto, esse processo manual pode ser demorado e inefici-

ente em ambientes dinâmicos. O uso de LLMs visa automatizar esse processo, permitindo a geração de análises contextuais detalhadas e a sugestão de ações corretivas de maneira mais rápida e eficiente. A automação dessa análise, tradicionalmente feita por equipes especializadas, oferece um potencial significativo para melhorar a resposta às anomalias e aumentar a resiliência dos sistemas.

Foram utilizadas técnicas de NLP baseadas em LLMs para realizar essa interpretação, especificamente os modelos GPT, Gemini e Claude. Para a análise contextual detalhada das sequências anômalas, foi adotada a abordagem de *Chain-of-thought (CoT) prompting*, uma técnica que orienta o modelo a raciocinar de forma encadeada, gerando uma explicação passo a passo sobre as causas subjacentes às anomalias, suas potenciais ameaças e ações corretivas sugeridas. Essa abordagem melhora a capacidade dos LLMs de fornecer análises detalhadas e mais precisas, especialmente em cenários complexos, como a interpretação de anomalias em eventos de *log* de sistemas de TI. Cada um desses LLMs foi configurado para interpretar as anomalias detectadas, permitindo uma análise comparativa das respostas geradas por diferentes abordagens de NLP. Esses modelos foram acionados por meio de um *prompt* desenvolvido especificamente para o cenário de detecção de anomalias em *logs* de sistemas de TI. Esse *prompt* foi construído considerando as necessidades operacionais e de segurança do ambiente analisado, com foco em quatro pontos principais: descrição da anomalia, avaliação de ameaças, possíveis causas e soluções recomendadas. Embora tenha sido desenhado para esse cenário específico, o *prompt* é flexível o suficiente para ser adaptado a outros contextos de análise, permitindo ajustes conforme as particularidades de diferentes sistemas e tipos de anomalias. A interpretação das anomalias ocorre em três fases principais:

1. **Análise Contextual:** Nesta fase, os modelos de LLM examinam os registros de *log* que precedem e seguem os registros anômalos, avaliando o contexto em que elas ocorreram. O *prompt* utilizado instrui o modelo a explicar detalhadamente a anomalia, indicando claramente o que a sequência de *log* reporta e por que pode ser considerada anômala. Essa análise é essencial para determinar se o evento anômalo é isolado ou faz parte de uma sequência interrelacionada de eventos, o que é fundamental para uma interpretação mais precisa.
2. **Geração de Resultados:** Com base na análise contextual, os LLMs geram análises detalhadas que abordam a descrição da anomalia e avaliam seu potencial de ameaça. A partir do *prompt*, o modelo examina se a anomalia representa uma ameaça direta à segurança ou à operação do sistema. As análises são claras e objetivas, permitindo que as equipes técnicas e de gestão compreendam rapidamente o impacto da anomalia no sistema.

3. **Recomendações de Ação:** A fase final envolve a proposição de soluções práticas e diretas para mitigar os riscos, problemas ou ameaças identificados com base nas anomalias interpretadas. O *prompt* orienta os LLMs a listar causas prováveis e sugerir ações corretivas ou investigações adicionais. As soluções recomendadas podem variar desde ajustes de configuração até análises mais profundas do comportamento do sistema, sempre baseadas nas respostas geradas pela análise contextual e pela descrição detalhada fornecida pelo modelo.

Ao utilizar LLMs e um *prompt* cuidadosamente elaborado, o processo de interpretação das anomalias é parcialmente automatizado, auxiliando as equipes de TI na identificação de falhas e na tomada de ações proativas. Essa automação facilita a análise de grandes volumes de dados, permitindo que as anomalias detectadas sejam interpretadas de forma contextualizada. No entanto, é importante destacar que LLMs podem encontrar limitações na interpretação de anomalias não documentadas, como as ameaças de *zero-day*. Nesses casos, a solução também atua ao sinalizar essas anomalias, permitindo que sejam encaminhadas para análise por especialistas, complementando assim o processo automatizado

Capítulo 4

Experimentos

Este capítulo apresenta os cenários de avaliação da solução proposta, incluindo detalhes sobre os *logs* BGL e CCMEval utilizados, as configurações da solução e as métricas utilizadas para avaliação. Além disso, são descritas as ferramentas e tecnologias utilizadas, com o objetivo de facilitar a replicabilidade dos experimentos realizados.

4.1 Metodologia e Critérios de Avaliação dos Modelos

Nesta seção, são apresentadas as métricas utilizadas para avaliar a solução proposta, divididas em métricas para avaliação dos modelos de detecção de anomalias usados e métricas para comparação das interpretações realizadas pelas LLMs empregadas. Por fim, os cenários de teste definidos para os experimentos são apresentados,

4.1.1 Métricas de Desempenho para Detecção de Anomalias

A avaliação dos modelos de detecção de anomalias nos *logs* estudados é feita a partir de métricas que dependem de quatro categorias básicas de classificação de registros. São elas:

- **Verdadeiro Positivo (*True Positive* - TP):** Quantidade de registros anômalos que foram corretamente identificados como anomalias pelo modelo de detecção.
- **Verdadeiro Negativo (*True Negative* - TN):** Quantidade de registros normais que foram corretamente identificados como normais pelo modelo.
- **Falso Positivo (*False Positive* - FP):** Quantidade de registros normais incorretamente classificados como anomalias pelo modelo, indicando erros de classificação.

- **Falso Negativo (*False Negative* - FN):** Quantidade de registros anômalos que o modelo falhou em detectar, classificando-os incorretamente como normais.

Esses elementos são usados para calcular as principais métricas de desempenho, que são:

- **Precisão (*Precision*):** A porcentagem de sequências anômalas verdadeiras detectadas entre todas as sequências anômalas detectadas. A fórmula é dada por:

$$Precision = \frac{TP}{TP + FP}$$

- **Revocação (*Recall*):** A porcentagem de sequências anômalas verdadeiras detectadas entre todas as sequências anômalas existentes. A fórmula é dada por:

$$Recall = \frac{TP}{TP + FN}$$

- **F1-Score:** Representa a média harmônica entre precisão e revocação, sendo uma métrica útil quando é necessário um equilíbrio entre ambas. A fórmula é dada por:

$$F1-Score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

- **Curva Característica de Operação do Receptor (*Receiver Operating Characteristic Curve* - ROC):** A Curva ROC é uma ferramenta gráfica utilizada na avaliação do desempenho de modelos de classificação. Ela representa a relação entre a taxa de verdadeiros positivos (ou revocação) e a taxa de falsos positivos em diferentes limiares de classificação. Ao plotar essa curva, pode-se visualizar como o modelo equilibra a sensibilidade (verdadeiros positivos) e a taxa de falsos positivos, permitindo a análise de seu comportamento em várias configurações de limiar.
- **Área Sob a Curva Característica de Operação do Receptor (*Area Under the Receiver Operating Characteristic Curve* - AUC):** A AUC é uma métrica numérica que quantifica a capacidade do modelo em distinguir entre diferentes classes. Quanto maior a AUC, melhor o desempenho do modelo em termos de classificação. Se AUC for 1, o modelo faz uma distinção perfeita entre classes; se for 0,5, o desempenho é aleatório. A AUC é uma métrica para comparar e avaliar o desempenho de modelos de classificação.

4.1.2 Métricas para Comparação de Respostas dos LLMs

Além das métricas clássicas para detecção de anomalias, este estudo também utilizou métricas específicas para comparar as respostas geradas pelos modelos de LLMs. As métricas utilizadas foram as seguintes:

- **Similaridade do Cosseno (*Cosine Similarity*):** Utiliza TF-IDF para medir a similaridade entre as explicações geradas por diferentes LLMs. Isso é particularmente útil para medir quão semelhantes são as estruturas gerais das respostas dos modelos, mesmo quando utilizam vocabulário diferente. A fórmula para similaridade de cosseno é:

$$\text{Cosine Similarity} = \frac{A \cdot B}{\|A\| \|B\|}$$

onde A e B são vetores *TF-IDF* das explicações.

- ***BERT Score*:** Mede a similaridade semântica entre as explicações utilizando as representações de palavras de um modelo *Bidirectional Encoder Representations from Transformers* (BERT), proporcionando uma comparação mais robusta entre os textos.
- **Distância de Levenshtein (*Levenshtein Distance*):** Mede a diferença entre duas explicações contando o número mínimo de operações (inserções, deleções ou substituições) necessárias para transformar uma explicação na outra. Essa métrica é útil para identificar o grau de diferença na formulação textual das explicações, capturando variações sutis entre os modelos.
- ***METEOR Score*:** Mede a similaridade entre as explicações considerando correspondências entre palavras, sinônimos e reordenação de frases. Ele é frequentemente usado na avaliação de traduções automáticas.

As métricas foram selecionadas para capturar diferentes aspectos das respostas dos LLMs, abrangendo tanto a similaridade semântica (*BERT Score*, *Cosine Similarity*) quanto as divergências textuais (*Levenshtein Distance*, *METEOR Score*), permitindo identificar variações entre os modelos e destacar aqueles que produzem explicações mais consistentes ou divergentes. A comparação das respostas entre pares de modelos não visa diretamente identificar o melhor LLM para a interpretação de anomalias, mas sim avaliar a similaridade entre as explicações geradas por cada modelo. Isso é importante para entender como diferentes LLMs podem convergir ou complementar-se na interpretação das anomalias.

A motivação para essa comparação é selecionar combinações de modelos que ofereçam maior coerência ou complementaridade na análise dos *logs*. O resultado esperado

é identificar quais pares de modelos geram explicações semelhantes ou complementares, possibilitando uma escolha mais robusta para automatizar a interpretação e a detecção de anomalias em sistemas complexos. Esses resultados podem ser utilizados para otimizar a implementação de diagnósticos automatizados baseados em múltiplos LLMs, garantindo maior precisão e confiabilidade nas respostas.

4.1.3 Cenários de Avaliação

Os experimentos foram conduzidos em dois cenários distintos, utilizando os *logs* do BGL e do CCMEval. Esses *logs* possuem características específicas que permitiram avaliar a robustez dos modelos propostos. A estratégia de avaliação foi estruturada em duas etapas: inicialmente, todos os modelos de detecção de anomalias foram testados no cenário com *logs* BGL, um conjunto de dados amplamente estudado e com anomalias previamente identificadas, permitindo uma análise mais controlada e precisa do desempenho dos modelos. Com base nos resultados dessa etapa, os modelos com melhor desempenho foram selecionados para uma nova avaliação utilizando o *log* CCMEval, que foi coletado de um ambiente real e imprevisível de uma instituição financeira, onde as anomalias não são previamente conhecidas e o contexto é mais dinâmico e complexo.

Essa abordagem foi adotada para garantir que os modelos fossem primeiramente validados em um conjunto de dados controlado, como o BGL, antes de serem aplicados a um conjunto mais desafiador e não estruturado, como o CCMEval. Isso permitiu reduzir o risco de falhas na detecção de anomalias em dados reais e otimizar o uso de recursos computacionais, focando nos modelos mais promissores a cada etapa.

Essa estratégia de avaliação em duas etapas foi adotada para garantir que os modelos fossem validados inicialmente em um conjunto de dados controlado, como o BGL, antes de serem aplicados a um conjunto mais desafiador e não estruturado, como o CCMEval, coletado de um ambiente real e imprevisível. Dessa forma, foi possível reduzir o risco de falhas na detecção de anomalias em dados reais, além de otimizar o uso de recursos computacionais, concentrando os esforços nos modelos mais promissores, assegurando sua robustez e confiabilidade para a detecção de anomalias em contextos práticos.

Detecção de Anomalias no *log* BGL

BGL é um conjunto de dados aberto que foi inicialmente apresentado por Oliner e Stearley [7]. Basicamente ele é composto por *logs* coletados de um supercomputador *Blue-Gene/L* localizado nas instalações do *Lawrence Livermore National Labs* (LLNL) em Livermore, Califórnia. Este supercomputador é equipado com 131.072 processadores e 32.768GB de memória. Adicionalmente, o BGL possui as anomalias previamente identificadas, o que

```

- 1117957728 2005.06.05 R25-M1-N5-C:J05-U01 2005-06-05-00.48.48.452366
R25-M1-N5-C:J05-U01 RAS KERNEL INFO generating core.1395

- 1118081574 2005.06.06 R02-M1-N0-C:J12-U11 2005-06-06-11.12.54.909368
R02-M1-N0-C:J12-U11 RAS KERNEL INFO instruction cache parity error
corrected

KERNDTLB 1118539972 2005.06.11 R30-M0-N9-C:J16-U01
2005-06-11-18.32.52.640639 R30-M0-N9-C:J16-U01 RAS KERNEL FATAL data
TLB error interrupt

- 1119448856 2005.06.22 R03-M0-NB-C:J10-U11 2005-06-22-07.00.56.521555
R03-M0-NB-C:J10-U11 RAS KERNEL INFO CE sym 12, at 0x0b71f720, mask 0x40

```

Figura 4.1: Exemplos de mensagens do *log* BGL. As mensagens foram selecionadas aleatoriamente, e a ordem não reflete a sequência original de registros.

possibilita a análise das métricas de desempenho dos modelos de detecção de anomalias de forma mais precisa e controlada. Os *logs* contêm mensagens identificadas com diferentes categorias de alerta. Na primeira coluna dos *logs*, o caractere ‘-’ indica mensagens sem alerta, ou seja, registros que já foram classificados como normais. Todas as outras identificações representam registros considerados anômalos, permitindo uma clara distinção entre dados normais e anômalos para os estudos de detecção de anomalias. Portanto, as informações de identificação de alerta são úteis para pesquisas relacionadas à detecção e previsão de alertas. O conjunto de dados BGL tem sido amplamente utilizado em estudos que envolvem análise de *logs*, detecção de anomalias e previsão de falhas. A Figura 4.1 mostra exemplos de mensagens do BGL.

O conjunto de dados original do BGL abrange um total de 4.747.963 mensagens, compreendendo 4.399.503 registros normais e 348.460 registros anômalos, coletados ao longo de um período de 215 dias e ocupando um espaço de 708MB. No entanto, para os propósitos dos experimentos realizados neste trabalho, foi selecionada uma amostra das primeiras 2 milhões de linhas do conjunto de dados, compreendendo 1.771.267 registros normais e 228.733 registros anômalos.

A escolha de utilizar uma amostra das primeiras 2 milhões de linhas do conjunto de dados BGL foi baseada na necessidade de balancear representatividade e eficiência computacional, em consonância com os objetivos específicos deste trabalho. O conjunto original, composto por 4.747.963 mensagens, impõe desafios computacionais consideráveis. Ao optar por essa amostra, conseguimos preservar a diversidade temporal e a variabilidade entre registros, elementos fundamentais para os experimentos, ao mesmo tempo que simplifica a execução dos modelos e algoritmos, tornando o processo analítico mais eficiente. A Tabela 4.1 sintetiza as informações relativas aos dados originais e à amostra utilizada.

Tabela 4.1: Tamanho e Composição do Conjunto de Dados BGL

	# Msgs	# Normais	# Anômalos	% Anômalos
Conjunto Original	4.747.963	4.399.503	348.460	7,34%
Amostra Utilizada	2.000.000	1.771.267	228.733	11,44%

Entre as 2 milhões de linhas, sequências de registros de *log* foram selecionadas para cada tamanho de janela, de forma aleatória e sem reposição. Dessas sequências, 75% foram usadas para treinamento e 25% para teste. Além disso, os vetores nulos dos dados foram removidos. A Tabela 4.2 apresenta o resumo dos dados.

Tabela 4.2: Dados Resumidos do BGL

Tamanho da Janela	Total de Sequências	Treinamento (75%)	Teste (25%)
10 segundos	99.986	74.989	24.997
30 segundos	39.994	29.995	9.999
1 minuto	19.997	14.997	5.000
2 minutos	13.331	9.998	3.333
3 minutos	9.998	7.498	2.500

Detecção de Anomalias no *log* CCMEval

CCMEval é um *log* do *Microsoft Configuration Manager* (MCM), anteriormente conhecido como *System Center Configuration Manager* (SCCM), uma ferramenta de gerenciamento de sistemas desenvolvida pela Microsoft para gerenciar grandes grupos de computadores. Este *log* possibilita a identificação de anomalias, apontando possíveis falhas e vulnerabilidades nos agentes do MCM instalados nos *endpoints* da infraestrutura de TI. A análise detalhada do CCMEval fornece informações valiosas para a detecção e correção de problemas, garantindo que a integridade e a segurança dos sistemas da instituição financeira sejam mantidas. O *log* CCMEval foi coletado de uma instituição financeira como um cenário de uso real para validar os modelos mais eficazes identificados nos experimentos do *Blue Gene/L* e em seguida para ser utilizado na fase de interpretação das sequências anômalas identificadas.

Os dados utilizados para treinar os modelos foram obtidos de cinco servidores operando normalmente (sem falhas) por oito meses, resultando em um total de 31.696 mensagens após a agregação. Em parceria com especialistas da Microsoft, foi estabelecido o “Ground Truth” para identificar anomalias no *log* do CCMEval, garantindo a máxima confiabilidade na avaliação dos resultados. Para testar os modelos, foram coletados *logs* de quatro servidores ao longo de cinco meses, onde as falhas foram mapeadas em colaboração com

os especialistas. Isso resultou em 9.261 mensagens após a agregação, das quais 236 eram anômalas (2,55%).

Anomalias identificadas por especialistas da Microsoft para o “*Ground Truth*”:

1. Instance (*) of class (SMS_Client) does not exist (0x80041003)
2. Checking WMI repository for feature Provider failed
3. WMI check failed
4. Result: Failed, ResultCode: 504, ResultType: 202, ResultDetail: *#CCMEVALPARAMSEP#SMS_Client#CCMEVALPARAMSEP#0x80041003
5. Result: Remediation Failed, ResultCode: -2147023843
6. Updating MDM_ConfigSetting.ClientHealthStatus with value 3
7. Updating MDM_ConfigSetting.ClientHealthStatus with value 7
8. Failed to get MDM_ConfigSetting instance, 0x80041013
9. Client health check: FAILED
10. Failed to send client health status as a state message. Attempting to send status to the FSP.
11. Attempting to change service status for service ‘CmRcService’ to ‘Running’.

Das 31.696 mensagens obtidas por meio da agregação para o treinamento, 2.175 sequências de *log* foram geradas após a exclusão dos vetores nulos. O conjunto de dados de teste, composto por 9.261 mensagens, resultou em 578 sequências de *log* após a remoção dos vetores nulos. A Tabela 4.3 resume os dados.

Tabela 4.3: Dados Resumidos do CCMEval

Tamanho da Janela	Total de Sequências	Treinamento (79%)	Teste (21%)
10 segundos	2.753	2.175	578

4.2 Experimentos com o BGL

Esta seção apresenta os experimentos realizados utilizando o *log* BGL. O objetivo principal é comparar o desempenho de diferentes configurações de modelos e analisar a influência de diversos parâmetros na detecção de anomalias.

4.2.1 Experimento 1.1 - Comparação de *F1-Score*, *Recall*, *Precision* e curva ROC

O primeiro experimento foi conduzido com o objetivo de avaliar o desempenho dos modelos utilizando diferentes configurações de janelas temporais. Para isso, testamos janelas de 10

segundos, 30 segundos, 1 minuto, 2 minutos e 3 minutos, empregando a técnica de *word embedding* GloVe com 50 dimensões para a vetorização das sequências de *logs*. As métricas de avaliação incluíram o *F1-Score*, *Precision*, *Recall* e a curva ROC. Esses indicadores foram escolhidos para fornecer uma visão abrangente da capacidade do modelo em detectar anomalias com precisão, minimizando tanto falsos positivos quanto falsos negativos. O *F1-Score* é particularmente útil para medir o equilíbrio entre essas métricas, enquanto a curva ROC permite visualizar o desempenho do modelo em diferentes limiares de decisão. Cada janela temporal foi avaliada com essas métricas para investigar como a variação na granularidade temporal das sequências influencia a precisão e a capacidade de detecção de anomalias.

4.2.2 Experimento 1.2 - Análise da Sensibilidade dos Modelos diante da Variação da Taxa de Contaminação

O segundo experimento focou em analisar a sensibilidade dos modelos à variação da taxa de contaminação dos modelos. A taxa de contaminação representa a proporção de anomalias presentes no conjunto de dados e é um parâmetro fundamental para ajustar a detecção de anomalias, especialmente em cenários de dados não supervisionados. A taxa foi variada para investigar sua influência nas métricas *F1-Score*, *Precision* e *Recall*, permitindo uma análise detalhada de como os modelos reagem à presença de um maior ou menor número de anomalias. Essa abordagem visa explorar até que ponto a taxa de contaminação pode impactar os resultados gerais da detecção de anomalias e auxiliar na identificação de configurações ideais para diferentes tipos de conjuntos de dados. A análise fornece uma visão importante sobre o papel da taxa de contaminação na calibragem do modelo, sem a necessidade de rótulos explícitos para os dados anômalos.

4.3 Experimentos com o CCMEval

Nesta seção, são detalhados os cenários de experimentação com o *log* CCMEval. O objetivo é testar a eficácia dos modelos previamente validados no *log* BGL em um ambiente do mundo real para detecção de anomalias em agentes do MCM. Para esse fim, foram considerados *logs* extraídos de um cenário real de uma instituição financeira.

4.3.1 Experimento 2.1 - Comparação de *F1-Score*, *Recall*, *Precision* e curva ROC

Neste experimento, o objetivo foi avaliar a eficácia dos modelos não supervisionados que apresentaram os melhores resultados no *log* BGL, agora aplicados ao *log* real CCME-

val, em um cenário do mundo real. A experimentação buscou identificar anomalias que possam indicar falhas e vulnerabilidades nos agentes do MCM instalados nos *endpoints* de uma infraestrutura de TI. Para isso, foi realizada uma comparação das métricas de *F1-Score*, *Recall*, *Precision* e Curva ROC, com o intuito de compreender como esses modelos respondem a dados mais dinâmicos e menos estruturados, típicos de um ambiente de produção real.

4.3.2 Experimento 2.2 - Interpretação das Anomalias no *log* CC-MEval

A interpretação de anomalias em registros de *log* é fundamental para contextualizar os resultados e oferecer orientações sobre as possíveis causas de falhas e vulnerabilidades nos sistemas. Após identificar o modelo de melhor desempenho no *log* CCMEval, com base em métricas como *F1-Score*, *Recall*, *Precision* e Curva ROC, esse modelo foi selecionado para identificar as sequências anômalas. Em seguida, as anomalias detectadas foram analisadas pelos modelos de LLMs GPT, Gemini e Claude, com o objetivo de interpretar as causas das falhas. Foram selecionadas três sequências anômalas do *log* CCMEval, e cada uma foi analisada detalhadamente pelos LLMs com base em critérios predefinidos. As três sequências anômalas foram selecionadas devido à sua representatividade de diferentes tipos de falhas detectadas no *log* CCMEval: falha de envio de mensagens de status (Sequência 1), erro ao acessar uma instância de configuração crítica (Sequência 2), e falha na remediação de um serviço essencial (Sequência 3). Esses casos foram escolhidos para cobrir uma variedade de cenários de falhas, permitindo uma interpretação abrangente pelos modelos LLM. Para orientar a interpretação das anomalias, utilizando a abordagem de *Chain-of-thought (CoT) prompting*, o *prompt* fornecido às LLMs foi estruturado em quatro aspectos principais:

1. **Descrição da Anomalia:** O modelo forneceu uma descrição clara sobre o que a sequência de *log* indica e a razão pela qual foi classificada como anômala.
2. **Potencial Ameaça:** O modelo avaliou se a anomalia detectada poderia representar uma ameaça à segurança ou à operação do sistema, apresentando justificativas com base na análise dos registros.
3. **Causas Prováveis:** O modelo identificou possíveis causas ou eventos que poderiam ter levado à ocorrência da anomalia, correlacionando os padrões observados com cenários conhecidos de falhas.

4. **Soluções Sugeridas:** O modelo sugeriu ações corretivas ou medidas de mitigação, além de indicar estratégias para uma investigação mais aprofundada da origem da anomalia.

Os três LLMs foram configurados para utilizar o mesmo *prompt*, garantindo a comparabilidade dos resultados. O *prompt* foi estruturado para orientar os modelos a gerarem explicações detalhadas e contextuais, abordando todos os aspectos mencionados. A seguir, apresenta-se um exemplo do *prompt* utilizado:

```
A sequência de logs abaixo foi detectada como uma anomalia no log  
'ccmeval.log' do sistema 'Microsoft Configuration Manager':
```

```
<Sequência de log>
```

```
Por favor, forneça uma análise detalhada dessa anomalia, abordando os  
seguintes pontos:
```

1. **Descrição da Anomalia:** Explique claramente o que a sequência de logs indica e por que pode ser considerada anômala.
2. **Potencial Ameaça:** Avalie se essa anomalia pode representar uma ameaça à segurança ou à operação do sistema, justificando sua resposta.
3. **Causas Prováveis:** Liste as possíveis causas ou eventos que podem ter levado à ocorrência dessa anomalia.
4. **Soluções Sugeridas:** Sugira ações ou correções que podem ser tomadas para resolver ou investigar mais a fundo a anomalia identificada.

Foram analisadas três sequências de *log* anômalas, representativas de diferentes eventos críticos do *log* CCMEval. Cada uma foi submetida a análises individuais pelos três modelos (GPT, Gemini e Claude), utilizando o mesmo *prompt* e critérios de avaliação para garantir a comparabilidade dos resultados. A seleção dessas sequências buscou cobrir uma ampla variedade de anomalias, incluindo falhas em processos críticos e comportamentos inesperados que poderiam comprometer a operação do sistema gerenciado pelo MCM. Esse enfoque permitiu observar como os modelos lidam com diferentes tipos de anomalias e fornecem explicações relevantes.

Ao final do experimento, cada modelo produziu explicações detalhadas e contextualizadas, permitindo uma comparação em termos de clareza, profundidade e relevância. As explicações geradas não apenas forneceram uma visão abrangente das anomalias, mas também ajudaram a mapear suas causas e propor ações corretivas. Este experimento visa validar a eficácia dos LLMs em fornecer interpretações assertivas e aplicáveis em cenários reais de operação de TI, como a análise de *logs* do CCMEval.

4.4 Configuração Experimental

Nesta seção, são detalhadas as configurações dos hiperparâmetros dos modelos de detecção de anomalias, bem como as ferramentas e tecnologias utilizadas na condução dos experimentos, a fim de facilitar a sua replicabilidade.

4.4.1 Configuração de Hiperparâmetros

Os hiperparâmetros utilizados neste estudo seguem uma abordagem padronizada. Esses hiperparâmetros, que são parâmetros de configuração externos ao modelo, desempenham um papel fundamental na definição do comportamento dos algoritmos de detecção de anomalias não supervisionadas. A escolha por valores padrão visa garantir uma base comparativa consistente entre as diferentes metodologias, minimizando o impacto de ajustes específicos e permitindo uma análise objetiva do desempenho intrínseco de cada modelo. Essa padronização nos parâmetros experimentais facilita a análise imparcial, proporcionando uma avaliação focada nas características fundamentais de cada abordagem. Com isso, torna-se possível identificar padrões de comportamento e entender melhor as nuances de cada técnica de detecção de anomalias, sem a interferência de ajustes personalizados que poderiam enviesar os resultados.

A única variação intencional nos hiperparâmetros é observada na taxa de contaminação do conjunto de dados, que representa a proporção estimada de anomalias presentes. A escolha dessa variável como um parâmetro ajustável permite adaptar os modelos às características específicas de cada conjunto de dados. A taxa de contaminação é utilizada como um limiar (*threshold*) nas pontuações de anomalia (*scores*) das amostras, e sua modificação é crucial para lidar com diferentes níveis de complexidade nos dados. Essa flexibilidade permite que os modelos ajustem sua sensibilidade às anomalias, melhorando a precisão em cenários com diferentes proporções de irregularidades.

4.4.2 Ferramentas e Tecnologias Utilizadas

Os experimentos foram conduzidos em uma máquina com o sistema operacional Windows 11 (Versão 22H2), equipada com um processador Intel Core i7 de 4 núcleos a 2.40GHz e 20GB de memória RAM, utilizando o ambiente Python 3.11. O desenvolvimento do código fez uso das seguintes bibliotecas: Pandas 1.5.3 para manipulação de dados, Numpy 1.24.3 para operações numéricas, Scikit-Learn 1.3.0 para os algoritmos de aprendizado de máquina, Pyod 1.1.0 para detecção de anomalias, Gensim 4.3.1 e NLTK 3.8.1 para processamento de linguagem natural, Tensorflow 2.13.0 e Keras 2.13.1 para modelagem de redes neurais, e MiniSom 2.3.1 para implementação de mapas auto-organizáveis.

Para a análise dos *logs*, foi utilizado o *log parser* Drain, disponível no repositório do Logpai no GitHub [48], que permitiu a extração eficiente de padrões estruturais nos registros de *log*. A comunicação com os LLMs GPT (gpt-4-turbo), Gemini (gemini-1.5-pro) e Claude (claude-3.5-sonnet) foi facilitada por meio das APIs `openai`, `google.generativeai` e `anthropic`, respectivamente. Essas APIs garantiram a integração contínua com os modelos, permitindo a geração automatizada de explicações detalhadas para as anomalias detectadas nos *logs*, sem interrupções no fluxo de trabalho dos experimentos.

Todos os códigos desenvolvidos podem ser acessados através do link <https://github.com/aly2009/DissertacaoUNB>.

Capítulo 5

Resultados e Discussão

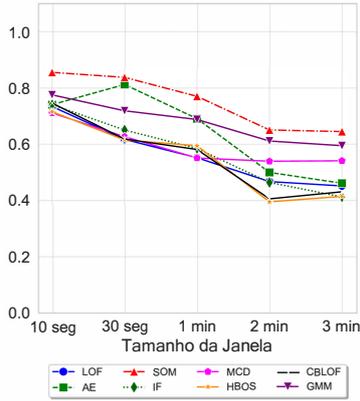
Este capítulo apresenta e discute os resultados dos experimentos realizados com os *logs* BGL e CCMEval, conforme os cenários descritos no capítulo anterior.

5.1 Experimentos com o BGL

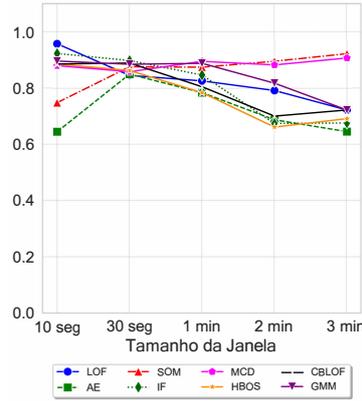
5.1.1 Experimento 1.1 - Comparação de *F1-Score*, *Recall*, *Precision* e curva ROC

As Figuras 5.1a, 5.1b e 5.1c mostram o *F1-Score*, *Recall* e *Precision* de cada um dos modelos avaliados. Já curvas ROC estão representadas nas Figuras 5.2a, 5.2b, 5.2c, 5.2d e 5.2e, as quais ilustram o desempenho dos modelos usando diferentes tamanhos de janela (10 segundos, 30 segundos, 1 minuto, 2 minutos e 3 minutos, respectivamente). De maneira geral, observa-se uma queda no *F1-Score* à medida que o tamanho da janela aumenta, isso ocorre principalmente devido a uma diminuição na *Precision*. A principal razão para essa queda é que, quanto maior a janela, mais eventos de *log* tendem a ser capturados. Como as anomalias são contextuais, um maior número de eventos capturados significa ter um contexto mais amplo. No entanto, essa ampliação do contexto pode levar à diluição das anomalias em meio a eventos normais, o que torna mais desafiador para o modelo identificar essas anomalias com precisão. A complexidade do problema também aumenta com janelas de tempo maiores, o que pode resultar em um maior número de falsos positivos e afetar negativamente a *Precision*. As curvas ROC e o respectivo AUC ilustrados na Figura 5.2 confirmam que a detecção de anomalias piora à medida que o tamanho da janela de tempo aumenta.

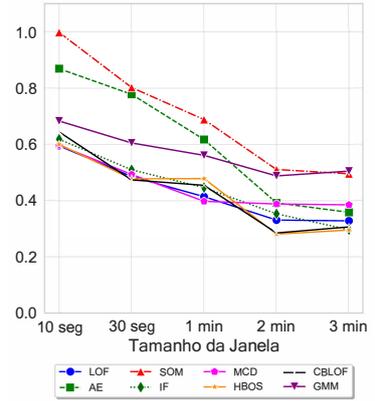
Examinando as métricas *F1-Score*, *Recall*, *Precision* e a curva ROC em conjunto, observa-se que os modelos SOM e AE apresentaram melhor desempenho à medida que o tamanho da janela aumentou, especialmente devido à sua maior robustez contra falsos



(a) *F1-Score*

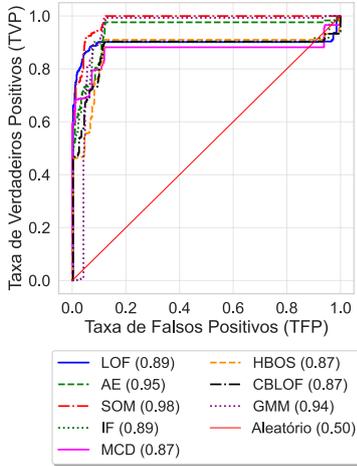


(b) *Recall*

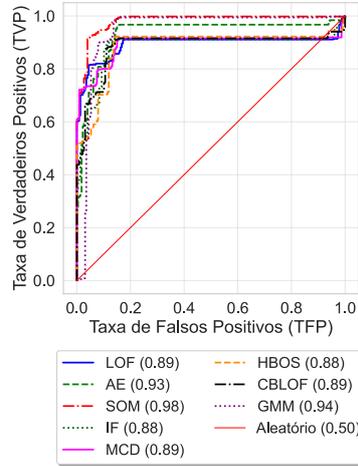


(c) *Precision*

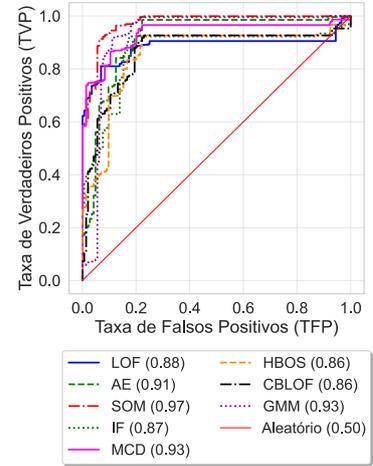
Figura 5.1: Desempenho dos modelos com janelas de tempo fixas no BGL.



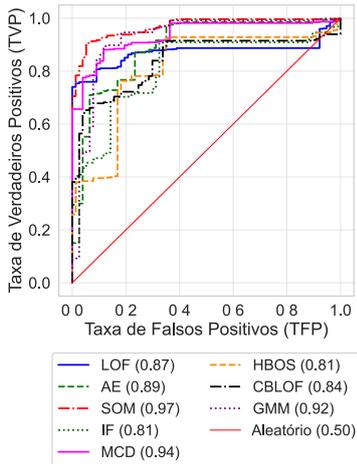
(a) Janela de 10 segundos



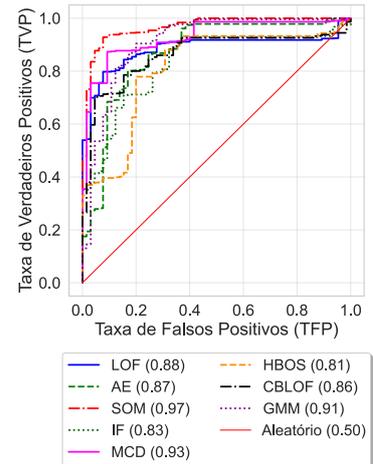
(b) Janela de 30 segundos



(c) Janela de 1 minuto



(d) Janela de 2 minutos



(e) Janela de 3 minutos

Figura 5.2: Curvas ROC dos modelos com janelas de tempo fixas no BGL.

positivos, como ilustrado principalmente na Figura 5.1c. Essa superioridade pode ser explicada pela natureza dos modelos de redes neurais, que possuem uma capacidade intrínseca de identificar e representar padrões complexos de forma mais eficaz, especialmente em espaços de alta dimensão. Além disso, esses modelos se destacam pela habilidade de reduzir a dimensionalidade, tolerar ruído e lidar com entradas incompletas, contribuindo para maior precisão e generalização. Por outro lado, o modelo MCD obteve o melhor desempenho em *Recall*, evidenciando sua capacidade de detectar anomalias globais com base em características estatísticas robustas. No entanto, esse modelo apresentou uma queda na *Precision* em janelas maiores, devido ao aumento de falsos positivos. Esse comportamento sugere que, embora o MCD seja eficaz na identificação de padrões gerais de anomalia, ele pode ser prejudicado por uma maior proporção de eventos não anômalos classificados como positivos. Em contraste, o modelo HBOS apresentou o pior desempenho geral, com baixa performance nas métricas de *F1-Score*, *Recall*, *Precision* e AUC. Essa deficiência foi particularmente evidente em janelas maiores, demonstrando sua susceptibilidade a falsos positivos e sua limitada capacidade de capturar anomalias contextuais. Esse resultado pode ser atribuído à natureza univariada do HBOS, baseada em histogramas, que não consegue lidar de forma eficaz com a complexidade multivariada dos eventos de *log*. Por fim, é importante destacar que o desempenho de todos os modelos é influenciado pela quantidade e distribuição das anomalias nos dados. Avaliar como esses modelos reagem a diferentes proporções de dados anômalos torna-se essencial, pois variações na taxa de contaminação podem impactar significativamente sua capacidade de detecção e precisão em diversos contextos.

5.1.2 Experimento 1.2 - Análise da Sensibilidade dos Modelos diante da Variação da Taxa de Contaminação

Neste experimento, foi analisada a sensibilidade dos modelos em relação às variações na taxa de contaminação, um hiperparâmetro que define a proporção estimada de dados anômalos no conjunto de dados. Os resultados ilustrados na Figura 5.3c mostram que, em geral, a *Precision* diminui à medida que a taxa de contaminação aumenta. Esse comportamento ocorre porque, com o aumento da proporção estimada de anomalias, os modelos começam a identificar mais instâncias como anômalas, mesmo quando muitas delas são, na verdade, normais. Como resultado, há um aumento no número de falsos positivos, o que impacta negativamente a precisão da classificação. Em contrapartida, o *Recall* (Figura 5.3b) apresenta um aumento inicial com a elevação da taxa de contaminação, pois mais instâncias anômalas são corretamente identificadas pelos modelos. No entanto, ele se estabiliza em valores altos à medida que a taxa de contaminação configurada no modelo

aumenta, sugerindo que os modelos conseguem capturar a maioria das anomalias, mas com uma diminuição significativa na qualidade geral das predições devido ao aumento de falsos positivos. O $F1$ -Score (Figura 5.3a), que reflete o equilíbrio entre $Precision$ e $Recall$, diminui à medida que a contaminação cresce. Essa queda é principalmente impulsionada pela redução acentuada na $Precision$, mesmo com o $Recall$ se mantendo em níveis elevados, indicando que a capacidade de discriminar corretamente entre instâncias normais e anômalas está sendo comprometida.

Analisando os resultados, observa-se que modelos SOM e AE novamente se destacam, apresentando um desempenho superior em comparação aos outros modelos, principalmente em termos de robustez do $F1$ -Score, corroborando as observações feitas no primeiro experimento. Essa robustez pode ser atribuída à maneira como esses modelos lidam com a representação dos dados, o que lhes confere maior resiliência às variações na taxa de contaminação. Dessa forma, decidiu-se avançar com SOM e AE para a avaliação seguinte, utilizando dados reais coletados através do CCMEval.

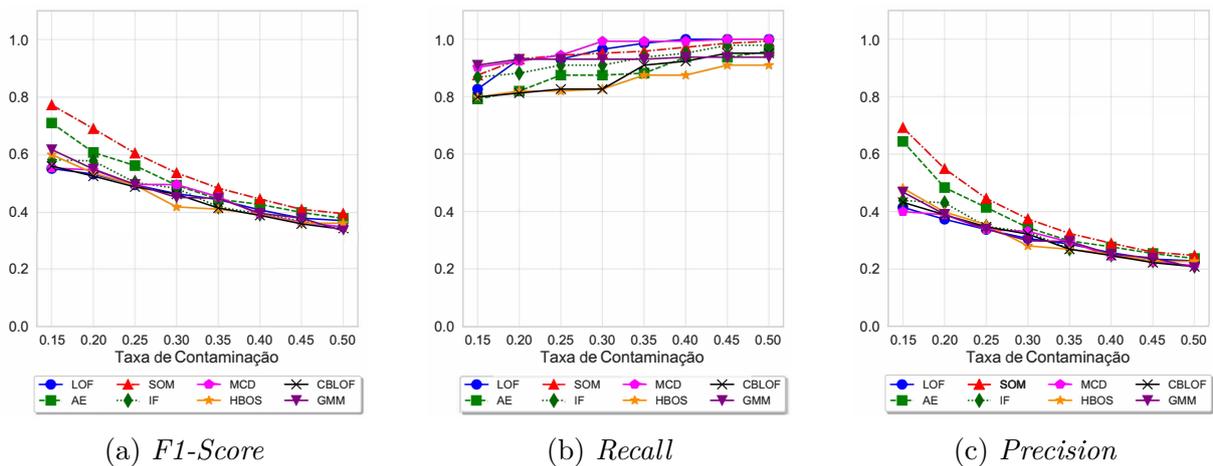


Figura 5.3: Desempenho com o aumento da taxa de contaminação no \log BGL.

5.2 Experimentos com o CCMEval

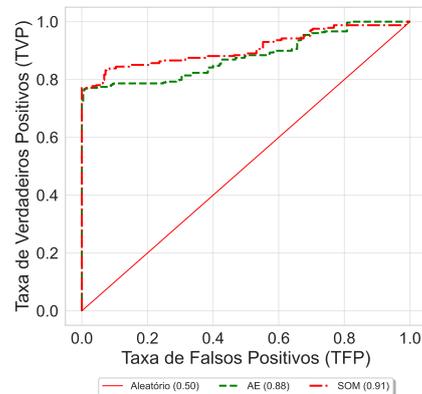
5.2.1 Experimento 2.1 - Comparação de $F1$ -Score, $Recall$, $Precision$ e curva ROC

Para avaliar o desempenho dos modelos de redes neurais em dados realistas, os resultados foram analisados utilizando o \log CCMEval. A análise focou nos modelos AE e SOM, que demonstraram os melhores resultados no cenário anterior (BGL). A Tabela 5.4a resume as métricas de desempenho, enquanto a Figura 5.4b apresenta as curvas ROC. Uma

abordagem de janela deslizante foi adotada, com um tamanho de janela de 10 segundos e um passo de 2 segundos.

Modelo	P	R	F1
AE	0.75	0.86	0.80
SOM	0.81	0.93	0.86

(a) Métricas de Desempenho



(b) Curva ROC

Figura 5.4: Desempenho dos modelos com janela de tempo fixa no CCMEval.

Os resultados indicam que tanto o modelo AE quanto o SOM são adequados para a detecção de anomalias no *log* CCMEval. O modelo AE apresentou *Precision* de 0,75, *Recall* de 0,86 e *F1-Score* de 0,80, refletindo um desempenho equilibrado na detecção de anomalias. O modelo SOM, por sua vez, destacou-se com métricas ligeiramente superiores, com *Precision* de 0,81, *Recall* de 0,93 e *F1-Score* de 0,86, evidenciando uma maior capacidade de identificar corretamente as anomalias. A análise das curvas ROC reforçam esses achados, com o modelo AE apresentando AUC de 0,88 e o SOM atingindo AUC de 0,91, sugerindo um desempenho robusto na separação entre instâncias normais e anômalas. Esses valores de AUC indicam que ambos os modelos de redes neurais possuem grande potencial para a detecção de anomalias em cenários práticos, oferecendo uma solução eficaz para a análise de *logs* do agente do MCM.

Pode-se afirmar que os resultados obtidos indicam que ambos os modelos apresentaram um desempenho sólido, com o SOM se destacando de maneira consistente em todas as métricas avaliadas. Enquanto o modelo AE demonstrou um bom equilíbrio entre *Precision* e *Recall*, o SOM mostrou uma vantagem geral tanto em termos de precisão quanto na capacidade de identificar corretamente as anomalias, refletido pelo seu *F1-Score* superior. Isso sugere que o SOM oferece uma abordagem mais robusta e eficaz para a detecção de anomalias, com desempenho notável na redução de falsos negativos e resultados consistentes em todas as métricas avaliadas.

5.2.2 Experimento 2.2 - Interpretação das Anomalias no *log* CC-MEval

Neste experimento, foram avaliados os resultados da interpretação de anomalias detectadas no *log* CCMEval, utilizando três modelos de LLM: GPT (*gpt-4-turbo*), Gemini (*gemini-1.5-pro*) e Claude (*claude-3.5-sonnet*). As sequências anômalas analisadas foram previamente identificadas utilizando o modelo SOM, que foi selecionado por ter apresentado o melhor desempenho entre todos os algoritmos testados na detecção de anomalias. Três dessas sequências foram analisadas detalhadamente, e cada LLM forneceu uma interpretação distinta. A seguir, são apresentadas as análises para cada sequência anômala, comparando as abordagens dos três modelos.

A comparação entre os modelos GPT (*gpt-4-turbo*), Gemini (*gemini-1.5-pro*) e Claude (*claude-3.5-sonnet*) revela diferenças significativas na interpretação das três sequências anômalas dos *logs* do MCM. A avaliação foi conduzida em pares, utilizando métricas como *Cosine Similarity*, *BERTScore*, *Levenshtein Distance* e *METEOR Score*, que proporcionam uma análise detalhada da similaridade e divergência nas saídas dos modelos. Essas métricas permitem quantificar as diferenças textuais e semânticas entre os pares de modelos, mostrando como cada um processou e apresentou as informações.

Sequência Anômala 1 - Falha no Envio do Status de Saúde do Agente

A primeira sequência anômala analisada foi a seguinte:

1. ‘Begin to send client health status report ’
2. ‘Failed to send client health status as a state message. Attempting to send status to the FSP.’
3. ‘Successfully sent client health status as an FSP message.’
4. ‘Evaluating Client Health status for MDM_ConfigSetting... ’
5. ‘Client health check: FAILED’
6. ‘Updating MDM_ConfigSetting.ClientHealthStatus with value 3’
7. ‘OS is not Win10RS3+, ENDOK.’
8. ‘===== [ccmeval finished normally in process 17040] =====’

A primeira sequência anômala envolve uma falha no envio do status de saúde do cliente (linha 2), seguida por uma tentativa bem-sucedida de enviar o status por meio do *Fallback Status Point* (linha 3). Posteriormente, a verificação de integridade do cliente falha (linha 5), e o sistema informa que o cliente não está rodando em uma versão compatível do sistema operacional (Win10RS3+, linha 7). Os três modelos forneceram análises consistentes, com ênfases diferentes sobre os eventos.

Resposta gerada pelo modelo GPT

1. **Descrição da Anomalia:**

A sequência de logs indica várias etapas e estados durante a avaliação da saúde do cliente no Microsoft Configuration Manager. A anomalia começa com uma tentativa falha de enviar o relatório de status de saúde do cliente como uma mensagem de estado, seguida por uma tentativa bem-sucedida de enviar essa informação como uma mensagem FSP (Fallback Status Point). Posteriormente, é relatado que a verificação de saúde do cliente falhou e o sistema atualiza o status de saúde do cliente para um valor específico (3). Finalmente, o log indica que o sistema operacional não é compatível com a versão mínima requerida (Win10RS3+), e o processo termina normalmente.

A natureza anômala desses logs reside na falha inicial em enviar o relatório de status de saúde e na falha subsequente na verificação de saúde do cliente, indicando potenciais problemas de configuração ou compatibilidade do sistema.

2. **Potencial Ameaça:**

Embora a anomalia não indique diretamente uma ameaça à segurança, ela sugere problemas operacionais que podem afetar a eficácia do gerenciamento e monitoramento do sistema. A incapacidade de verificar adequadamente a saúde do cliente pode levar a uma falta de detecção de outros problemas mais sérios, incluindo vulnerabilidades de segurança não identificadas. Portanto, enquanto a anomalia em si não é uma ameaça direta, ela pode expor o sistema a riscos não detectados.

3. **Causas Prováveis:**

- **Falha na Comunicação:** A falha inicial em enviar o relatório de status pode ser devida a problemas de rede ou configurações incorretas no ponto de comunicação.
- **Incompatibilidade de Versão do Sistema Operacional:** O log indica que o sistema operacional não atende à versão mínima requerida (Win10RS3+), o que pode ter contribuído para a falha na verificação de saúde.
- **Problemas de Configuração no Cliente:** Configurações incorretas ou corrompidas no cliente do Configuration Manager podem ter levado à falha na verificação de saúde.

4. **Soluções Sugeridas:

- **Revisão da Configuração de Rede:** Verificar e garantir que todas as configurações de rede e comunicação entre o cliente e o servidor estão corretas, incluindo as configurações do Fallback Status Point.
- **Atualização do Sistema Operacional:** Se possível, atualizar o sistema operacional para uma versão compatível que atenda aos requisitos mínimos (Win10RS3+).

- ****Diagnóstico do Cliente:**** Executar um diagnóstico completo no cliente do Configuration Manager para identificar e corrigir configurações incorretas ou corrompidas.
- ****Monitoramento Contínuo:**** Após as correções, monitorar continuamente os logs para garantir que não haja repetição das falhas e que o sistema esteja operando conforme esperado.

Essas ações ajudarão a mitigar os problemas identificados e melhorar a estabilidade e segurança do sistema.

Resposta gerada pelo modelo Gemini

Análise da Anomalia em ccmeval.log

****1. Descrição da Anomalia:****

A sequência de logs indica que o cliente do Microsoft Configuration Manager (SCCM) encontrou um problema ao enviar o relatório de status de integridade do cliente como uma mensagem de estado. Embora tenha conseguido enviar o relatório através do FSP (Fallback Status Point), a verificação de integridade do cliente falhou, resultando na atualização do status 'MDM_ConfigSetting.ClientHealthStatus' para '3' (que geralmente indica um estado de erro). A mensagem "OS is not Win10RS3+, ENDOK." sugere que o sistema operacional do cliente pode não atender aos requisitos mínimos para determinadas funcionalidades.

A anomalia reside na falha inicial em enviar o relatório de status e na subsequente falha na verificação de integridade do cliente.

****2. Potencial Ameaça:****

Embora não represente uma ameaça de segurança imediata, essa anomalia pode indicar problemas subjacentes:

- * ****Problemas de comunicação:**** A falha no envio da mensagem de estado pode indicar problemas de comunicação entre o cliente e o ponto de gerenciamento, impactando a aplicação de políticas e atualizações.
- * ****Funcionalidades limitadas:**** A mensagem sobre o sistema operacional sugere que o cliente pode não ser capaz de utilizar funcionalidades que exigem Win10RS3 ou superior, impactando a gestão e segurança do dispositivo.
- * ****Relatórios imprecisos:**** A falha na verificação de integridade pode levar a relatórios imprecisos no SCCM, dificultando a gestão e monitoramento da infraestrutura.

****3. Causas Prováveis:****

- * ****Problemas de rede:**** Latência , firewall ou problemas de DNS podem impedir a comunicação com o ponto de gerenciamento.
- * ****Configuração incorreta:**** Configurações incorretas no cliente SCCM ou no ponto de gerenciamento podem causar problemas de comunicação e verificação de integridade.
- * ****Sistema operacional desatualizado:**** A mensagem "OS is not Win10RS3+" indica que o sistema operacional do cliente pode estar desatualizado , causando incompatibilidades.
- * ****Componentes corrompidos do SCCM:**** Arquivos corrompidos ou configurações inválidas no cliente SCCM podem levar a falhas na verificação de integridade.

****4. Soluções Sugeridas:****

- * ****Verificar a conectividade de rede:**** Verificar a conectividade entre o cliente e o ponto de gerenciamento , incluindo resolução de nomes e portas.
- * ****Revisar as configurações do SCCM:**** Verificar as configurações do cliente e do ponto de gerenciamento para garantir que estejam corretas e compatíveis.
- * ****Atualizar o sistema operacional:**** Se possível , atualizar o sistema operacional do cliente para Win10RS3 ou superior para garantir a compatibilidade.
- * ****Executar ferramentas de reparo do SCCM:**** Utilizar ferramentas como o " Configuration Manager Client Health Evaluation Tool" para diagnosticar e corrigir problemas no cliente SCCM.
- * ****Consultar os logs do SCCM:**** Analisar os logs do servidor SCCM e do cliente para obter mais informações sobre a causa raiz da anomalia.

É crucial investigar mais a fundo para determinar a causa raiz da anomalia e aplicar as soluções adequadas para garantir a funcionalidade completa e a segurança do cliente SCCM.

Resposta gerada pelo modelo Claude

Entendido. Vou analisar a sequência de logs fornecida e responder aos pontos solicitados:

1. Descrição da Anomalia:

A sequência de logs indica uma falha na verificação de integridade do cliente do Microsoft Configuration Manager. Especificamente:

- O sistema tentou enviar um relatório de status de integridade do cliente , mas falhou inicialmente.

- Conseguiu enviar o status através de um método alternativo (FSP).
- A verificação de integridade do cliente falhou para o componente MDM_ConfigSetting.
- O sistema não está executando Windows 10 RS3 ou superior.

Esta sequência é considerada anômala porque indica uma falha na verificação de integridade do cliente, o que não é um comportamento esperado em condições normais de operação.

2. Potencial Ameaça:

Esta anomalia não representa necessariamente uma ameaça direta à segurança, mas indica um problema de configuração ou compatibilidade que pode afetar a operação adequada do sistema de gerenciamento. A falha na verificação de integridade pode impactar a capacidade do Configuration Manager de gerenciar efetivamente o dispositivo, potencialmente deixando-o sem atualizações críticas ou configurações de segurança.

3. Causas Prováveis:

- Incompatibilidade de versão do sistema operacional (OS não é Windows 10 RS3+).
- Problemas de conectividade de rede afetando a comunicação com o servidor do Configuration Manager.
- Configurações incorretas no cliente do Configuration Manager.
- Corrupção de arquivos ou componentes do cliente do Configuration Manager.
- Conflitos com software de segurança de terceiros.

4. Soluções Sugeridas:

- Verificar a versão do sistema operacional e considerar a atualização para uma versão compatível do Windows.
- Investigar e resolver quaisquer problemas de conectividade de rede.
- Reinstalar ou reparar o cliente do Configuration Manager.
- Verificar e corrigir as configurações do cliente do Configuration Manager.
- Analisar logs adicionais para identificar possíveis conflitos com outros softwares.
- Executar uma verificação de integridade do sistema para identificar possíveis problemas de corrupção de arquivos.

É importante realizar uma investigação mais aprofundada para determinar a causa raiz específica e garantir que não haja implicações de segurança mais amplas.

Analisando individualmente cada uma das respostas dos modelos, pode-se concluir que:

- **GPT** destacou a falha de comunicação inicial e a incompatibilidade de versão como

causas centrais da anomalia. As soluções sugeridas incluíram a verificação das configurações de rede, atualização do sistema operacional e a execução de diagnósticos no cliente MCM. O GPT focou bastante na importância de garantir que o sistema esteja operando em uma versão compatível e que as configurações do cliente estejam corretas.

- **Gemini** enfatizou potenciais problemas de comunicação e configurações incorretas, sugerindo que o sistema operacional desatualizado poderia estar impedindo o uso de certas funcionalidades. O modelo propôs verificar a conectividade de rede, revisar configurações do MCM e atualizar o sistema operacional. Também sugeriu a utilização de ferramentas de reparo específicas para o cliente MCM.
- **Claude** focou na incompatibilidade do sistema operacional como a causa principal, levantando também problemas de rede e possíveis conflitos com software de terceiros. As soluções sugeridas incluíram a atualização do sistema operacional, resolução de problemas de conectividade e reparo do cliente MCM. Claude foi o mais detalhado na sugestão de investigar conflitos com outros softwares e executar verificações de integridade do sistema.

A Tabela 5.1 apresenta as métricas de similaridade entre os modelos LLM para a Sequência Anômala 1.

Modelos	Cosine Similarity	BERT Score	Levenshtein Distance	METEOR Score
GPT vs Gemini	0.7610	0.9905	2042	0.4190
GPT vs Claude	0.7597	0.9846	1932	0.2818
Gemini vs Claude	0.7448	0.9857	2103	0.2480

Tabela 5.1: Métricas de Similaridade entre Modelos - Sequência Anômala 1

A *Cosine Similarity* entre GPT e Gemini foi de 0.7610, indicando que as explicações fornecidas por ambos os modelos são estruturalmente semelhantes, com algumas variações. Esse valor, próximo de 1, sugere que, apesar de algumas diferenças, os modelos compartilham uma estrutura textual geral similar. As variações podem estar relacionadas à forma como cada modelo prioriza diferentes aspectos das explicações, como o foco em certas causas da falha. Entre GPT e Claude, a similaridade foi de 0.7597, apontando para uma estrutura textual igualmente próxima, embora não idêntica. O resultado mais baixo entre Gemini e Claude (0.7448) revela uma maior divergência na organização das informações e no enfoque de detalhes técnicos.

O *BERT Score*, que mede a similaridade semântica, foi elevado em todos os pares de comparação, o que indica que, embora as construções frasais variem, o conteúdo e o significado centrais das explicações são praticamente idênticos entre os modelos. Essa

alta similaridade reflete o fato de que todos os modelos compreendem o problema técnico de maneira semelhante — a falha no envio do status de saúde do agente — e oferecem interpretações muito próximas das causas e consequências dessa falha.

Já o *Levenshtein Distance* revela que a maior distância de edição foi observada entre Gemini e Claude (2103), o que sugere que, embora semanticamente próximos, esses modelos apresentam diferenças significativas na formulação das frases, resultando em um maior número de operações de edição para transformar um texto no outro. Isso pode ser explicado pelas escolhas distintas de palavras ou pela organização das ideias. A menor distância foi entre GPT e Claude (1932), indicando que esses dois modelos têm menos variações em como expressam suas respostas, com menos mudanças necessárias para alinhar os textos.

Por fim, observa-se que O *METEOR* Score entre GPT e Gemini foi de 0.4190, superior ao score entre GPT e Claude (0.2818), o que indica que GPT e Gemini utilizaram mais sinônimos e construções frasais semelhantes. Claude, por outro lado, optou por uma formulação mais distinta. O menor score foi registrado entre Gemini e Claude (0.2480), o que sugere uma divergência maior na escolha de termos e na estrutura das frases, ainda que o conteúdo central permaneça semanticamente próximo.

Segunda Sequência Anômala - Erro ao Obter Instância de Configuração do Agente

A segunda sequência anômala analisada foi a seguinte:

1. ‘Updating MDM_ConfigSetting.ClientHealthStatus with value 7’
2. ‘===== [ccmeval finished normally in process 1668] =====’
3. ‘Failed to get MDM_ConfigSetting instance , 0x80041013 ’

A segunda sequência anômala apresenta uma tentativa de atualização do status de saúde do cliente (linha 1), seguida pela conclusão do processo (linha 2). No entanto, ocorre uma falha crítica após o término do processo (linha 3), ao tentar obter a instância de configuração *MDM_ConfigSetting*, resultando no erro 0x80041013. Todos os três modelos forneceram descrições detalhadas e consistentes dos eventos, destacando a tentativa de atualização inicial seguida pela falha crítica.

Resposta gerada pelo modelo GPT

1. Descrição da Anomalia:

A sequência de logs indica que houve uma tentativa de atualizar a configuração de saúde do cliente (*ClientHealthStatus*) no Microsoft Configuration Manager (SCCM) com o valor 7, seguido por uma mensagem indicando que o processo de avaliação (*ccmeval*) foi concluído normalmente. No entanto, a

anomalia surge com a mensagem final, que relata uma falha ao tentar obter a instância de MDM_ConfigSetting com o código de erro 0x80041013. Este erro sugere uma falha na consulta ao WMI (Windows Management Instrumentation), o que pode ser considerado anômalo, especialmente se a operação anterior indicava que o processo havia terminado normalmente.

2. Potencial Ameaça:

Embora a anomalia possa não representar uma ameaça direta à segurança, ela pode afetar a operação e a integridade do sistema de gerenciamento de configurações. A incapacidade de acessar ou modificar configurações de gerenciamento de dispositivos móveis (MDM) pode levar a problemas de compliance, gestão de políticas e saúde geral do sistema de TI. Não há indicação direta de uma ameaça de segurança, mas a falha operacional pode ter implicações indiretas, como a não aplicação de políticas de segurança devido a configurações incorretas ou não atualizadas.

3. Causas Prováveis:

- ****Corrupção do Repositório WMI:**** O repositório pode estar corrompido, impedindo o acesso às instâncias necessárias.
- ****Permissões Insuficientes:**** O usuário ou serviço que executou a operação pode não ter as permissões necessárias para acessar ou modificar as configurações no WMI.
- ****Problemas de Conectividade ou Configuração:**** Problemas na rede ou na configuração do servidor que hospeda o WMI podem ter impedido o acesso às instâncias.
- ****Falha de Software ou Bug:**** Um bug no SCCM ou em componentes relacionados pode ter causado a falha ao tentar acessar as configurações do MDM.

4. Soluções Sugeridas:

- ****Verificar e Reparar o WMI:**** Utilize ferramentas como o ‘WMIDiag‘ para diagnosticar e reparar problemas no WMI. Isso pode incluir a reconstrução do repositório WMI se estiver corrompido.
- ****Revisar Permissões de Acesso:**** Certifique-se de que as contas de usuário ou serviços tenham as permissões adequadas para operar com o WMI e o SCCM.
- ****Checar Conectividade e Configurações de Rede:**** Verifique se há problemas de rede que possam estar impedindo a comunicação com o servidor WMI.
- ****Atualizar ou Patchear o Software:**** Certifique-se de que o SCCM e os componentes relacionados estão atualizados com as últimas versões e patches, corrigindo possíveis bugs que possam causar essas falhas.

Essas ações devem ajudar a resolver a anomalia detectada e garantir a operação adequada do sistema de gerenciamento de configurações.

Resposta gerada pelo modelo Gemini

Análise da Anomalia em 'ccmeval.log'

1. Descrição da Anomalia:

A sequência de logs apresenta uma inconsistência. Primeiro, indica a atualização do status de saúde do cliente (ClientHealthStatus) para "7", o que sugere sucesso. No entanto, logo após, a última linha relata falha ao obter a instância "MDM_ConfigSetting" com o código de erro "0x80041013". Essa contradição configura uma anomalia.

2. Potencial Ameaça:

Embora não seja possível afirmar com certeza a existência de uma ameaça à segurança, a anomalia precisa ser investigada. A falha em obter a instância "MDM_ConfigSetting" pode indicar problemas na comunicação com o servidor MDM (Mobile Device Management), erros de configuração ou, em casos mais raros, tentativas de manipulação do sistema.

3. Causas Prováveis:

- * **Problemas de comunicação:** Perda de conectividade de rede entre o cliente e o servidor MDM durante a operação.
- * **Erros no servidor MDM:** Falhas no serviço MDM, banco de dados ou problemas de autenticação/autorização.
- * **Configurações incorretas:** Parâmetros de configuração incorretos no cliente ou no servidor MDM.
- * **Corrupção de dados:** Dados corrompidos relacionados ao "MDM_ConfigSetting" no cliente ou no servidor.
- * **Interferência de software:** Software de segurança ou outro aplicativo interferindo na comunicação ou no processo.

4. Soluções Sugeridas:

- * **Verificar a conectividade de rede:** Confirmar se o cliente possui acesso estável ao servidor MDM.
- * **Analisar os logs do servidor MDM:** Buscar por erros ou eventos relacionados ao cliente em questão.
- * **Revisar as configurações do MDM:** Verificar se as configurações do cliente e do servidor estão corretas e sincronizadas.
- * **Executar ferramentas de diagnóstico:** Utilizar ferramentas do Configuration Manager para diagnosticar problemas de comunicação e configuração do MDM.
- * **Consultar a documentação da Microsoft:** Buscar informações sobre o código de erro "0x80041013" e soluções relacionadas.

Resposta gerada pelo modelo Claude

Entendido. Vou analisar a sequência de logs fornecida e responder aos pontos solicitados:

1. ****Descrição da Anomalia:****

A sequência de logs indica uma inconsistência no processo de atualização do status de integridade do cliente no Microsoft Configuration Manager. O log mostra uma tentativa de atualizar o status (ClientHealthStatus) para o valor 7, seguida por uma conclusão aparentemente normal do processo cmeval. No entanto, logo após, há uma falha ao tentar obter uma instância de MDM_ConfigSetting, com o código de erro 0x80041013. Esta sequência é anômala porque indica uma operação bem-sucedida seguida imediatamente por uma falha relacionada.

2. ****Potencial Ameaça:****

Esta anomalia, por si só, não representa necessariamente uma ameaça direta à segurança do sistema. No entanto, pode indicar um problema de configuração ou integridade do sistema que, se não for resolvido, pode levar a falhas mais graves ou vulnerabilidades. É necessário investigar mais para determinar se há implicações de segurança mais amplas.

3. ****Causas Prováveis:****

- Corrupção de dados no repositório de configuração do Configuration Manager.
- Problemas de permissão ou acesso ao repositório WMI (Windows Management Instrumentation).
- Conflito de versão entre componentes do Configuration Manager.
- Falha temporária na comunicação entre os componentes do sistema.
- Problema de integridade do sistema operacional subjacente.

4. ****Soluções Sugeridas:****

- Verificar e reparar a integridade do repositório WMI usando ferramentas como winmgmt /verifyrepository e winmgmt /salvagerepository.
- Reiniciar o serviço Windows Management Instrumentation.
- Verificar logs adicionais do Configuration Manager para identificar problemas relacionados.
- Executar uma verificação de integridade do sistema usando o DISM (Deployment Image Servicing and Management) e o SFC (System File Checker).
- Atualizar o cliente do Configuration Manager para a versão mais recente compatível com o servidor.
- Consultar a base de conhecimento da Microsoft para problemas conhecidos relacionados ao código de erro 0x80041013.

É importante realizar uma investigação mais aprofundada para determinar a causa raiz e garantir que não haja implicações de segurança mais amplas.

As respostas dos modelos abordaram causas prováveis variadas e propuseram soluções específicas para resolução do problema identificado, com diferenças em seus enfoques:

- **GPT** focou na possibilidade de corrupção do repositório WMI, problemas de permissão e falhas de software, sugerindo soluções como a verificação e reparo do WMI, revisão de permissões e atualização do MCM. A análise foi bem estruturada e enfatizou a integridade do sistema como fator chave.
- **Gemini** enfatizou problemas de conectividade e interferências de software, sugerindo que a falha poderia estar ligada à perda de comunicação entre o cliente e o servidor MDM. As soluções propostas incluíram a verificação da conectividade de rede e uma análise mais profunda dos logs do servidor MDM, além de revisar as configurações de cliente e servidor.
- **Claude** abordou a anomalia de forma mais ampla, levantando a hipótese de corrupção de dados ou problemas de permissão no WMI, além de falhas temporárias na comunicação entre componentes. Sugeriu soluções robustas, incluindo a verificação da integridade do repositório WMI, reinicialização de serviços e uso de ferramentas como DISM e SFC para garantir a integridade do sistema operacional.

A Tabela 5.2 apresenta as métricas de similaridade entre os modelos LLM para a Sequência Anômala 2.

Modelos	Cosine Similarity	BERT Score	Levenshtein Distance	METEOR Score
GPT vs Gemini	0.6475	0.9813	1896	0.3001
GPT vs Claude	0.6199	0.9925	1882	0.3004
Gemini vs Claude	0.6044	0.9850	1568	0.3079

Tabela 5.2: Métricas de Similaridade entre Modelos - Sequência Anômala 2

A *Cosine Similarity* entre GPT e Gemini foi de 0.6475, indicando uma menor similaridade estrutural em comparação com a Sequência Anômala 1. Esse valor sugere que, embora as explicações sejam relacionadas à mesma anomalia, há variações consideráveis na forma como os dois modelos organizaram suas respostas. Entre GPT e Claude, a similaridade foi ainda menor, 0.6199, o que indica uma divergência mais acentuada na forma de apresentar as informações. O valor mais baixo foi registrado entre Gemini e Claude (0.6044), o que revela uma divergência ainda maior entre esses dois modelos na estrutura textual das respostas, sugerindo abordagens mais distintas.

Já o *BERT Score* se manteve elevado em todas as comparações, com destaque para GPT vs Claude (0.9925), o que indica que, apesar das diferenças estruturais observadas na

Cosine Similarity, a similaridade semântica é alta. Isso implica que, embora os modelos usem estruturas textuais diferentes, as interpretações das causas e consequências da falha no *MDM_ConfigSetting* são semanticamente muito próximas. Essa alta similaridade semântica entre todos os pares reflete que os três modelos compreendem o problema de forma bastante consistente.

A menor *Levenshtein Distance* foi observada entre Gemini e Claude (1568), sugerindo que, apesar das diferenças estruturais evidenciadas pela *Cosine Similarity*, os textos desses dois modelos requerem menos mudanças para serem alinhados. A maior distância foi entre GPT e Gemini (1896), o que sugere uma maior discrepância na escolha de palavras e na construção frasal entre esses dois modelos. Entre GPT e Claude, a distância foi semelhante (1882), indicando que, embora a semântica seja similar, há variações significativas na forma como esses modelos expressam suas respostas.

O *METEOR Score* mais elevado foi registrado entre Gemini e Claude (0.3079), o que sugere que esses dois modelos compartilharam mais sinônimos e construções frasais semelhantes em relação aos outros pares. Por outro lado, a pontuação entre GPT e Gemini (0.3001) e GPT e Claude (0.3004) foi quase idêntica, sugerindo que a similaridade lexical entre esses modelos foi consistente, ainda que inferior à similaridade observada entre Gemini e Claude.

Terceira Sequência Anômala - Falha na Remediação do Serviço de Proxy

Por fim, a terceira sequência anômala analisada foi a seguinte:

1. `'Result: Remediation Failed , ResultCode: -2147023843, ResultType: 200, ResultDetail: '`
2. `'Evaluating health check rule {934F12E3-295E-4BA0-AE0F-09859685720F} : Verify/Remediate Configuration Manager Proxy service startup type.'`

A terceira sequência anômala indica uma falha no processo de remediação (linha 1), com o código de erro -2147023843. Em seguida, ocorre uma verificação de regra de saúde para avaliar o tipo de inicialização do serviço *Configuration Manager Proxy* (linha 2). A falha na remediação pode estar relacionada a essa verificação, exigindo investigação mais detalhada. Os três modelos descreveram consistentemente essa falha, associando-a à verificação do tipo de inicialização do serviço Proxy.

Resposta gerada pelo modelo GPT

1. ****Descrição da Anomalia:****

A sequência de logs indica que houve uma falha na remediação durante a avaliação de uma regra de verificação de saúde no Microsoft Configuration Manager. A regra específica, identificada pelo GUID

{934F12E3–295E–4BA0–AE0F–09859685720F}, estava verificando ou tentando remediar o tipo de inicialização do serviço de proxy do Configuration Manager. O código de resultado ‘–2147023843’ sugere que a operação de remediação não foi bem–sucedida. A falta de detalhes adicionais no ‘ResultDetail’ pode indicar que o erro não produziu uma mensagem de erro específica ou que o sistema não conseguiu capturar detalhes adicionais sobre a falha.

2. ****Potencial Ameaça:****

Embora uma falha na remediação não seja necessariamente uma ameaça direta à segurança, ela pode representar um risco operacional. Se o serviço de proxy do Configuration Manager não estiver configurado corretamente, isso pode afetar a capacidade do sistema de gerenciar corretamente as configurações de segurança e operacionais dos dispositivos cliente. Isso pode deixar o sistema vulnerável a problemas de desempenho ou a falhas de segurança se não for corrigido prontamente.

3. ****Causas Prováveis:****

- ****Permissões Insuficientes:**** O processo que tentou realizar a remediação pode não ter tido as permissões necessárias para alterar o tipo de inicialização do serviço.
- ****Corrupção do Serviço ou Configuração:**** O serviço de proxy pode estar corrompido ou mal configurado, impedindo alterações no tipo de inicialização.
- ****Problemas de Comunicação ou Conectividade:**** Problemas de rede podem ter impedido a comunicação adequada com o serviço durante a tentativa de remediação.
- ****Erro de Sistema ou Software:**** Um bug ou falha no próprio Configuration Manager ou no sistema operacional pode ter causado a falha na remediação.

4. ****Soluções Sugeridas:****

- ****Verificar Permissões:**** Certifique–se de que o usuário ou o processo que executa a remediação tenha as permissões necessárias para modificar os serviços do sistema.
- ****Revisar Configurações do Serviço:**** Verifique manualmente as configurações do serviço de proxy do Configuration Manager para garantir que estão corretas e tente ajustá–las conforme necessário.
- ****Consultar Logs Adicionais:**** Verifique outros logs do sistema e do Configuration Manager para possíveis mensagens de erro relacionadas que possam fornecer mais contexto sobre a falha.
- ****Testar Conectividade de Rede:**** Certifique–se de que não há problemas de rede que possam estar afetando a comunicação com o serviço.

- ****Atualizar ou Reparar o Configuration Manager:**** Se suspeitar de um bug ou corrupção, considere atualizar ou reparar a instalação do Configuration Manager.

Essas ações devem ajudar a identificar a causa raiz da anomalia e a corrigir o problema para restaurar a funcionalidade normal do sistema.

Resposta gerada pelo modelo Gemini

Análise da Anomalia em 'ccmeval.log'

1. Descrição da Anomalia:

A sequência de logs indica uma falha na correção de um problema detectado pelo Microsoft Configuration Manager (SCCM) durante a verificação de integridade do sistema. O problema específico está relacionado ao tipo de inicialização do serviço "Configuration Manager Proxy".

Por que é uma anomalia:

- * ****"Remediation Failed"**: Indica que o SCCM tentou corrigir automaticamente um problema, mas não obteve sucesso.**
- * ****"ResultCode: -2147023843"**: Este código de erro específico pode indicar problemas de permissões, corrupção de arquivos ou configurações incorretas no sistema.**
- * ****"ResultType: 200"**: Embora o código 200 geralmente indique sucesso, neste contexto, combinado com "Remediation Failed", sugere uma inconsistência ou um problema na execução da correção.**

2. Potencial Ameaça:

A anomalia em si não representa uma ameaça direta à segurança, mas indica um problema na funcionalidade do SCCM, especificamente na capacidade de corrigir automaticamente problemas no sistema. Isso pode levar a:

- * ****Vulnerabilidades futuras:**** Se o tipo de inicialização do serviço "Configuration Manager Proxy" estiver incorreto, o sistema pode ficar vulnerável a explorações ou ter sua funcionalidade comprometida.
- * ****Gerenciamento ineficaz:**** A falha na correção pode impedir o SCCM de aplicar atualizações de segurança, monitorar o sistema adequadamente ou executar outras tarefas de gerenciamento.

3. Causas Prováveis:

- * **Permissões insuficientes**: A conta de serviço do SCCM pode não ter as permissões necessárias para modificar o tipo de inicialização do serviço .
- * **Configurações de diretiva de grupo**: Diretivas de grupo podem estar substituindo as configurações do SCCM.
- * **Corrupção de arquivos**: Arquivos de sistema corrompidos relacionados ao serviço "Configuration Manager Proxy" ou ao próprio SCCM podem estar causando o problema.
- * **Interferência de software de terceiros**: Softwares de segurança ou otimização podem estar interferindo na operação do SCCM.

****4. Soluções Sugeridas:****

- * **Verificar permissões**: Assegurar que a conta de serviço do SCCM tenha as permissões necessárias para modificar as configurações do serviço "Configuration Manager Proxy".
- * **Analisar diretivas de grupo**: Verificar se existem diretivas de grupo que podem estar substituindo as configurações do SCCM.
- * **Executar verificação de integridade**: Utilizar ferramentas como SFC (System File Checker) para verificar e reparar arquivos de sistema corrompidos.
- * **Reiniciar o serviço**: Reiniciar o serviço "Configuration Manager Proxy" e o serviço SCCM.
- * **Consultar logs detalhados**: Analisar os logs do SCCM para obter informações mais detalhadas sobre a falha na correção, incluindo mensagens de erro específicas.
- * **Consultar a documentação da Microsoft**: Buscar soluções específicas para o código de erro -2147023843 na documentação do SCCM.

Resposta gerada pelo modelo Claude

Entendido. Vou analisar a anomalia detectada no log 'ccmeval.log' do Microsoft Configuration Manager conforme solicitado:

1. Descrição da Anomalia:

A sequência de logs indica uma falha na remediação de uma regra de verificação de integridade do Configuration Manager. Especificamente:

- O resultado da operação foi "Remediation Failed" (Remediação Falhou)
- O código de resultado é -2147023843
- O tipo de resultado é 200
- A regra em questão é identificada pelo GUID {934F12E3-295E-4BA0-AE0F-09859685720F}
- A regra está relacionada à verificação/remediação do tipo de inicialização do serviço Proxy do Configuration Manager

Esta sequência é considerada anômala porque indica uma falha na execução de uma tarefa de manutenção automática do sistema.

2. Potencial Ameaça:

Esta anomalia, por si só, não representa necessariamente uma ameaça direta à segurança. No entanto, pode indicar um problema operacional que, se não o for resolvido, pode afetar o funcionamento adequado do Configuration Manager. É necessário investigar mais para determinar se há implicações de segurança mais amplas.

3. Causas Prováveis:

- Configurações incorretas do serviço Proxy do Configuration Manager
- Permissões insuficientes para modificar o tipo de inicialização do serviço
- Corrupção no registro do Windows relacionado ao serviço
- Conflito com software de segurança ou outro software de gerenciamento
- Problemas de integridade do sistema operacional
- Falha na comunicação entre componentes do Configuration Manager

4. Soluções Sugeridas:

- Verificar manualmente o tipo de inicialização do serviço Proxy do Configuration Manager
- Analisar logs adicionais do Configuration Manager para obter mais contexto
- Executar o Configuration Manager Evaluation Tool manualmente
- Verificar as permissões da conta de serviço do Configuration Manager
- Realizar uma verificação de integridade do sistema operacional
- Considerar a reinstalação ou reparo do cliente Configuration Manager
- Consultar a documentação da Microsoft para orientações específicas sobre essa regra de integridade

É importante investigar mais a fundo para determinar a causa raiz e garantir que não haja implicações mais amplas para a segurança ou operação do sistema.

Para essas respostas, observa-se que as causas prováveis e as soluções sugeridas pelos modelos variaram.

- **GPT** focou na falha de remediação e levantou a hipótese de problemas de permissão, corrupção do serviço ou problemas de conectividade. As soluções propostas incluíram verificar permissões, revisar configurações do serviço manualmente e verificar *logs* adicionais para entender a falha. O GPT também sugeriu reparar ou atualizar o MCM, uma solução robusta para lidar com falhas de configuração.

- **Gemini** enfatizou a falha de correção automática, destacando o código de erro como um indicativo de permissões insuficientes ou corrupção de arquivos. O modelo sugeriu soluções como verificar permissões, analisar políticas de grupo e reiniciar o serviço, além de executar verificações de integridade e consultar a documentação da Microsoft para mais detalhes sobre o erro.
- **Claude** apresentou uma análise detalhada da falha e sugeriu causas como configurações incorretas, permissões insuficientes e conflitos com outros softwares. Suas soluções incluíram verificar o tipo de inicialização do serviço manualmente, verificar permissões, e realizar verificações de integridade do sistema operacional. Claude também recomendou a reinstalação ou reparo do cliente MCM, abordando possíveis problemas de configuração ou corrupção.

A Tabela 5.3 apresenta as métricas de similaridade entre os modelos LLM para a Sequência Anômala 3.

Modelos	Cosine Similarity	BERT Score	Levenshtein Distance	METEOR Score
GPT vs Gemini	0.7177	0.9839	2128	0.4247
GPT vs Claude	0.7471	0.9876	1984	0.2886
Gemini vs Claude	0.6773	0.9834	2021	0.2318

Tabela 5.3: Métricas de Similaridade entre Modelos - Sequência Anômala 3

A *Cosine Similarity* para a Sequência Anômala 3 mostrou maior similaridade entre GPT e Claude (0.7471), sugerindo que esses dois modelos apresentaram respostas com estruturas textuais relativamente próximas. Entre GPT e Gemini, o valor foi de 0.7177, revelando um nível moderado de similaridade. O menor valor foi registrado entre Gemini e Claude (0.6773), o que indica que esses dois modelos, embora semanticamente próximos, tiveram uma organização das informações mais distinta.

O *BERT Score* continuou elevado, com pequenas variações entre os pares. O maior valor foi observado entre GPT e Claude (0.9876), indicando que as respostas desses dois modelos são semanticamente muito próximas, sugerindo que, apesar de pequenas diferenças estruturais, ambos os modelos compreenderam e interpretaram a anomalia de forma bastante semelhante. Gemini e Claude também apresentaram uma alta similaridade semântica (0.9834), reforçando que o conteúdo central das respostas é consistente entre os modelos.

A maior distância de edição foi registrada entre GPT e Gemini (2128), indicando que os textos desses dois modelos foram os que mais divergiram em termos de formulação frasal. A menor distância foi entre GPT e Claude (1984), sugerindo que, embora as explicações de ambos os modelos sejam semanticamente próximas, houve algumas variações significativas

nas escolhas de palavras e na estrutura textual. Gemini e Claude, com uma distância de 2021, também apresentaram diferenças consideráveis, embora um pouco menores.

Por fim, O *METEOR Score* mais alto foi registrado entre GPT e Gemini (0.4247), indicando que esses dois modelos utilizaram mais sinônimos e construções frasais semelhantes. Já a pontuação entre GPT e Claude (0.2886) e entre Gemini e Claude (0.2318) foi mais baixa, sugerindo que Claude utilizou uma formulação mais distinta, com menos sinônimos e variações lexicais em comparação com os outros modelos. A diferença mais acentuada entre Gemini e Claude sugere que, apesar de uma alta similaridade semântica, suas construções textuais foram substancialmente diferentes.

Discussão dos resultados

A análise comparativa entre os modelos *gpt-4-turbo*, *gemini-1.5-pro*, e *claude-3.5-sonnet* demonstrou que, embora as explicações fornecidas por todos os modelos sejam semanticamente próximas, há variações na organização e estruturação das respostas. As métricas indicam que essas diferenças, embora notáveis, não comprometem a precisão semântica das explicações. Todos os modelos apresentaram interpretações consistentes das anomalias nos *logs* do MCM, identificando as principais causas e sugerindo soluções adequadas para os problemas analisados.

Embora a similaridade semântica seja elevada em todos os casos, as variações estruturais entre os modelos refletem diferentes estilos de apresentação e ênfase. Essas variações não afetam a precisão das explicações, mas podem influenciar a escolha do modelo mais adequado em cenários onde a clareza e o enfoque em certos detalhes técnicos são determinantes. Dependendo da necessidade de maior detalhamento ou de uma explicação mais direta, um modelo pode se destacar em relação aos outros.

Em resumo, a análise confirma que os três modelos são capazes de fornecer explicações coerentes e eficazes para as anomalias analisadas. As diferenças entre eles residem principalmente na forma de organizar e expressar as informações, sem comprometer a consistência semântica, que se mantém alta em todos os casos.

Capítulo 6

Conclusões e Trabalhos Futuros

Este estudo investigou a aplicação de técnicas de aprendizado de máquina não supervisionado para a detecção de anomalias em *logs* de sistemas de TI. Com o aumento exponencial da quantidade de dados gerados por esses sistemas, a necessidade de métodos automatizados e eficazes para identificar falhas e vulnerabilidades tornou-se crítica nas áreas de monitoramento e segurança. A combinação entre aprendizado de máquina e modelos de NLP não apenas detectou anomalias, mas também gerou interpretações contextuais e automatizadas, uma vantagem significativa em comparação com abordagens tradicionais. A hipótese central foi comprovada, demonstrando que a aplicação de técnicas de aprendizado de máquina não supervisionado, aliadas ao NLP, é eficaz na identificação e interpretação de comportamentos anômalos como potenciais ameaças ou falhas. Esta dissertação oferece uma contribuição importante ao propor um fluxo de trabalho robusto que combina a vetorização semântica de sequências de eventos com algoritmos de aprendizado de máquina, como *Autoencoders* (AE) e *Self-Organizing Map* (SOM).

Os resultados experimentais confirmaram a eficácia da abordagem proposta, demonstrando que os modelos não apenas detectaram anomalias de forma precisa, mas também forneceram uma base sólida para a análise posterior dessas falhas. Com isso, foi alcançado o objetivo geral de desenvolver, comparar e aprimorar modelos de detecção de anomalias, verificando a aplicabilidade tanto na detecção quanto na interpretação das anomalias por meio de NLP e LLMs. Nos testes realizados com os *logs* do *Blue Gene/L* (BGL) e, posteriormente, com os *logs* do *Microsoft Configuration Manager* (CCMEval), o AE alcançou *F1-score* de 0,80, enquanto o SOM chegou a 0,86, sobressaindo-se como o modelo com o melhor desempenho nas fases iniciais de detecção de anomalias. Esses valores são particularmente significativos, considerando que os modelos são não supervisionados, indicando que eles conseguem equilibrar bem a detecção de anomalias (*recall*) e a minimização de falsos positivos (*precision*) sem a necessidade de rótulos explícitos nos dados. Esse desempenho robusto reafirma a aplicabilidade dos modelos em ambientes reais, especialmente no

setor financeiro, onde a detecção precisa de anomalias é essencial para garantir a operação segura e contínua.

Além disso, os objetivos específicos foram cumpridos: (1) comparar e testar diferentes modelos de detecção de anomalias, validando os modelos com melhor desempenho; (2) aplicar e analisar detalhadamente o *log* CCMEval coletados de uma instituição financeira para identificar comportamentos inesperados; e (3) interpretar as anomalias detectadas por meio de NLP e LLMs, fornecendo explicações contextualizadas de suas causas e recomendações corretivas. Este estudo avança o estado da arte na detecção e interpretação de anomalias, integrando técnicas de NLP para análise interpretativa. A aplicação de LLMs, como GPT, Gemini e Claude, proporcionou a automatização da análise de anomalias de forma contextualizada. Com a abordagem *Chain-of-Thought (CoT)* para criação de *prompts*, esses modelos foram capazes de fornecer explicações detalhadas sobre as anomalias, classificando-as em termos de severidade, indicando causas prováveis e recomendando ações corretivas. Após a fase de detecção, o modelo SOM foi escolhido para identificar as sequências anômalas, por ter apresentado o melhor desempenho geral, e foi utilizado na avaliação com as LLMs. Essa automação da interpretação foi essencial para aprimorar a compreensão dos eventos críticos e reduzir o tempo de resposta para mitigar falhas em sistemas de TI.

Por meio da análise detalhada das métricas de desempenho dos modelos, considerando não apenas a acurácia geral, mas também fatores como a taxa de contaminação e o tamanho das janelas temporais, foi possível identificar os modelos mais adequados para diferentes tipos de ambientes, como aqueles que geram *logs* ruidosos ou que apresentam variações frequentes no comportamento dos sistemas monitorados. A análise de sensibilidade demonstrou que a robustez dos modelos está fortemente ligada à qualidade e estabilidade dos dados, sendo que a vetorização semântica se mostrou essencial para lidar com essa variabilidade.

Adicionalmente, esta dissertação contribui de forma prática ao implementar a solução proposta em um cenário de grande escala e relevância, como o gerenciamento de *endpoints* em uma instituição financeira. A validação realizada com o *log* CCMEval confirmou a eficácia do fluxo de trabalho desenvolvido, provando que a combinação de aprendizado de máquina não supervisionado com NLP pode não apenas detectar anomalias de maneira eficiente, mas também fornecer informações valiosas para a correção imediata, reduzindo significativamente o impacto de eventos inesperados.

Em suma, os resultados experimentais confirmaram a hipótese de que a combinação de aprendizado de máquina não supervisionado com NLP oferece uma solução eficaz para detecção e interpretação de anomalias em *logs* de sistemas de TI. O SOM e o AE foram os modelos mais promissores na fase de detecção, sendo o SOM o escolhido para avaliação

com as LLMs devido ao seu desempenho superior. Assim, este estudo apresenta uma solução abrangente e eficaz, abordando uma lacuna relevante na literatura contemporânea sobre detecção de anomalias. A implementação bem-sucedida dos modelos em diferentes cenários de teste e sua validação em ambiente real reforçam a aplicabilidade prática da pesquisa. Ao integrar técnicas de aprendizado de máquina não supervisionado com ferramentas de NLP, este estudo abre caminho para futuras inovações na análise de *logs*, com grande potencial para aplicação em sistemas que lidam com grandes volumes de dados.

6.1 Perspectivas para a Continuidade do Trabalho

Diversas direções promissoras podem ser exploradas em estudos futuros para expandir e aprimorar este trabalho, como a exploração de novos algoritmos de aprendizado de máquina não supervisionado e técnicas híbridas que combinem aprendizado supervisionado e não supervisionado, as quais podem proporcionar melhorias significativas na precisão da detecção de anomalias e na redução de falsos positivos. Além disso, novos desenvolvimentos em aprendizado profundo, como redes neurais recorrentes e *transformers*, podem ser investigados para analisar séries temporais de eventos de *logs* com maior profundidade.

Outra área de interesse é a aplicação da metodologia proposta em diferentes cenários, abrangendo *logs* variados de diversos sistemas, como sistemas operacionais, redes, dispositivos IoT e aplicações corporativas. A diversidade nos formatos e comportamentos desses *logs* pode exigir adaptações nas técnicas empregadas, especialmente em relação à escalabilidade e à robustez da solução para lidar com diferentes volumes e complexidades de dados.

O aprimoramento das capacidades de interpretação das anomalias por meio de técnicas de NLP, utilizando modelos de linguagem de grande escala mais recentes, é uma direção estratégica para permitir uma análise mais rica e contextualizada das falhas. Além disso, a incorporação de *feedback* humano no ciclo de detecção, através de técnicas como aprendizado por reforço, poderia refinar as explicações e recomendações fornecidas pelos modelos. A realização de um estudo de campo com profissionais da área, disponibilizando a solução para testes práticos e coleta de opiniões, pode oferecer melhorias valiosas para ajustes e refinamentos, assegurando que a solução atenda plenamente às necessidades do setor.

Finalmente, uma possível linha de continuidade é a integração desta solução com sistemas de resposta automática a incidentes, permitindo uma mitigação rápida e automatizada de falhas e vulnerabilidades detectadas. Avaliações de impacto em ambientes de produção, com análises de eficiência operacional e redução de custos, também podem

fornecer um entendimento mais detalhado dos benefícios práticos da implementação das soluções propostas neste estudo.

Referências

- [1] Huang, Shaohan, Carol J. Fung, Kui Wang, Polo Pei, Zhongzhi Luan e Depei Qian: *Using Recurrent Neural Networks Toward Black-Box System Anomaly Prediction*. 2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS), páginas 1–10, 2016. 1
- [2] Du, Min, Feifei Li, Guineng Zheng e Vivek Srikumar: *Deeplog: Anomaly detection and diagnosis from system logs through deep learning*. ACM Special Interest Group on Security, Audit and Control, páginas 1285–1298, 2017. 1, 15
- [3] Ahmed, Mohiuddin, Abdun Naser Mahmood e Md Rafiqul Islam: *A Survey of Anomaly Detection Techniques in Financial Domain*. Future Generation Computer Systems, 55:278–288, fevereiro 2016. 1
- [4] Samuel, A L: *Some Studies in Machine Learning Using the Game of Checkers*. IBM Journal of Research and Development, 3(3):210–229, 1959. 2, 10
- [5] Mitchell, Tom M: *Machine learning*. McGraw-hill New York, 1997. 2, 10
- [6] Zhang, Xu, Yong Xu, Qingwei Lin, Bo Qiao, Hongyu Zhang, Yingnong Dang, Chunyu Xie, Xinsheng Yang, Qian Cheng, Ze Li *et al.*: *Robust Log-based Anomaly Detection on Unstable Log Data*. ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, páginas 807–817, 2019. 2, 8, 15, 23
- [7] Oliner, Adam e Jon Stearley: *What Supercomputers Say: A Study of Five System Logs*. IEEE/IFIP International Conference on Dependable Systems and Networks, páginas 575–584, 2007. 4, 30
- [8] Microsoft: *Microsoft System Center Configuration Manager*, 2024. <https://docs.microsoft.com/en-us/mem/configmgr/>, Accessed: 2023-07-21. 4
- [9] He, Pinjia, Jieming Zhu, Shilin He, Jian Li e Michael R Lyu: *An evaluation study on log parsing and its use in log mining*. IEEE/IFIP international conference on dependable systems and networks, páginas 654–661, 2016. 6
- [10] He, Shilin, Jieming Zhu, Pinjia He e Michael R. Lyu: *Experience Report: System Log Analysis for Anomaly Detection*. Em *Proceedings - International Symposium on Software Reliability Engineering, ISSRE*, páginas 207–218. IEEE Computer Society, dezembro 2016, ISBN 9781467390019. 8

- [11] Teller, Virginia: *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Computational Linguistics, 26:638–641, 2000. 8
- [12] Salton, Gerard e Christopher Buckley: *Term-weighting Approaches in Automatic Text Retrieval*. Information Processing & Management, 24(5):513–523, 1988. 8
- [13] Chandola, Varun, Arindam Banerjee e Vipin Kumar: *Anomaly Detection: A Survey*. ACM Computing Surveys, 41(3):1–58, 2009. 10
- [14] Wang, Song, Juan Fernando Balarezo, Sithamparanathan Kandeepan, Akram Al-Hourani, Karina Gomez Chavez e Benjamin Rubinstein: *Machine Learning in Network Anomaly Detection: A Survey*. IEEE Access, 9:152379–152396, 2021. 10
- [15] Nassif, Ali Bou, Manar Abu Talib, Qassim Nasir e Fatima Mohamad Dakalbab: *Machine Learning for Anomaly Detection: A Systematic Review*. IEEE Access, 9:78658–78700, 2021. 10
- [16] Breunig, Markus M, Hans Peter Kriegel, Raymond T Ng e Jörg Sander: *LOF: Identifying Density-Based Local Outliers*. ACM Special Interest Group on Management of Data, 29:93–104, 2000. 10, 24
- [17] Rumelhart, David E, Geoffrey E Hinton, Ronald J Williams *et al.*: *Learning Internal Representations by Error Propagation*. Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations, páginas 318–362, 1985. 10, 24
- [18] Kohonen, Teuvo: *The Self-organizing Map*. Proceedings of the IEEE, 78(9):1464–1480, 1990. 11, 24
- [19] Liu, Fei Tony, Kai Ming Ting e Zhi Hua Zhou: *Isolation Forest*. IEEE International Conference on Data Mining, páginas 413–422, 2008. 11, 24
- [20] Hubert, Mia e Michiel Debruyne: *Minimum covariance determinant*. Wiley interdisciplinary reviews: Computational statistics, 2(1):36–43, 2010. 11, 24
- [21] Goldstein, Markus e Andreas Dengel: *Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm*. German Conference on Artificial Intelligence: Poster and Demo Track, 1:59–63, 2012. 11, 24
- [22] He, Zengyou, Xiaofei Xu e Shengchun Deng: *Discovering Cluster-based Local Outliers*. Pattern Recognition Letters, 24(9-10):1641–1650, 2003. 11, 24
- [23] Reynolds, Douglas A *et al.*: *Gaussian mixture models*. Encyclopedia of biometrics, 741(659-663), 2009. 11, 24
- [24] Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever *et al.*: *Language models are unsupervised multitask learners*. OpenAI blog, 1(8):9, 2019. 12
- [25] Vaswani, Ashish, Google Brain, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser e Illia Polosukhin: *Attention Is All You Need*. Advances in Neural Information Processing Systems, 2017. 12

- [26] OpenAI, Team, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat *et al.*: *Gpt-4 technical report*. ArXiv, 2023. <http://arxiv.org/abs/2303.08774>. 12
- [27] Team, Gemini, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth *et al.*: *Gemini: A Family of Highly Capable Multimodal Models*. ArXiv, dezembro 2023. <http://arxiv.org/abs/2312.11805>. 13
- [28] Anthropic: *The Claude 3 Model Family: Opus, Sonnet, Haiku Anthropic*. Relatório Técnico, Anthropic, 2024. <https://docs.anthropic.com/>. 13
- [29] Anthropic: *Claude 3.5 Sonnet Model Card Addendum*. Relatório Técnico, Anthropic, 2024. <https://www.anthropic.com>. 14
- [30] Brown, Tom B: *Language models are few-shot learners*. ArXiv, 2020. <http://arxiv.org/abs/2005.14165>. 14
- [31] Liu, Pengfei, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi e Graham Neubig: *Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing*. ACM Computing Surveys, 55(9), janeiro 2023, ISSN 15577341. 14
- [32] Wei, Jason, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H Chi Quoc, V Le e Denny Zhou: *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models Chain-of-Thought Prompting*. Advances in neural information processing systems, 35:24824–24837, 2022. 14, 15
- [33] Sahoo, Pranab, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal e Aman Chadha: *A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications*. ArXiv, fevereiro 2024. <http://arxiv.org/abs/2402.07927>. 15
- [34] Debnath, Biplob, Mohiuddin Solaimani, Muhammad Ali Gulzar Gulzar, Nipun Arora, Cristian Lumezanu, Jianwu Xu, Bo Zong, Hui Zhang, Guofei Jiang e Latifur Khan: *LogLens: A Real-time Log Analysis System*. IEEE International Conference on Distributed Computing Systems, páginas 1052–1062, 2018. 15
- [35] Huang, Shaohan, Yi Liu, Carol Fung, Rong He, Yining Zhao, Hailong Yang e Zhongzhi Luan: *Hitanomaly: Hierarchical Transformers for Anomaly Detection in System Log*. IEEE Transactions on Network and Service Management, 17(4):2064–2076, 2020. 16
- [36] Studiawan, Hudan, Ferdous Sohel e Christian Payne: *Anomaly Detection in Operating System Logs with Deep Learning-based Sentiment Analysis*. IEEE Transactions on Dependable and Secure Computing, 18(5):2136–2148, 2020. 16
- [37] Lee, Yukyung, Jina Kim e Pilsung Kang: *Lanobert: System Log Anomaly Detection Based on BERT Masked Language Model*. Applied Soft Computing, 146:1–14, 2023. 16

- [38] Nam, Sukhyun, Jae Hyounng Yoo e James Won Ki Hong: *Vm failure prediction with log analysis using bert-cnn model*. Em *2022 18th International Conference on Network and Service Management (CNSM)*, páginas 331–337. IEEE, 2022. 16
- [39] Zhu, Jieming, Shilin He, Jinyang Liu, Pinjia He, Qi Xie, Zibin Zheng e Michael R. Lyu: *Tools and Benchmarks for Automated Log Parsing*. IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice, páginas 121–130, 2019. 17, 20, 21
- [40] Li, Xiaoyun, Pengfei Chen, Linxiao Jing, Zilong He e Guangba Yu: *Swisslog: Robust and unified deep learning based log anomaly detection for diverse faults*. Em *Proceedings - International Symposium on Software Reliability Engineering, IS-SRE*, volume 2020-October, páginas 92–103. IEEE Computer Society, outubro 2020, ISBN 9781728198705. 17
- [41] Lin, Qingwei, Hongyu Zhang, Jian Guang Lou, Yu Zhang e Xuewei Chen: *Log clustering based problem identification for online service systems*. Em *Proceedings - International Conference on Software Engineering*, páginas 102–111. IEEE Computer Society, maio 2016, ISBN 9781450341615. 17
- [42] Astekin, Merve, Harun Zengin e Hasan Sözer: *DILAF: A framework for distributed analysis of large-scale system logs for anomaly detection*. *Software: Practice and Experience*, 49:153–170, 2019. 17
- [43] Landauer, Max, Sebastian Onder, Florian Skopik e Markus Wurzenberger: *Deep Learning for Anomaly Detection in Log Data: A Survey*. ArXiv, julho 2022. <http://arxiv.org/abs/2207.03820>. 17
- [44] He, Pinjia, Jieming Zhu, Zibin Zheng e Michael R. Lyu: *Drain: An Online Log Parsing Approach with Fixed Depth Tree*. Em *Proceedings - 2017 IEEE 24th International Conference on Web Services, ICWS 2017*, páginas 33–40. Institute of Electrical and Electronics Engineers Inc., setembro 2017, ISBN 9781538607527. 20
- [45] Mikolov, Tomas, Kai Chen, Greg Corrado e Jeffrey Dean: *Efficient Estimation of Word Representations in Vector Space*. ArXiv, 2013. <http://arxiv.org/abs/1301.3781>. 22
- [46] Pennington, Jeffrey, Richard Socher e Christopher D Manning: *Glove: Global vectors for word representation*. *Conference on Empirical Methods in Natural Language Processing*, páginas 1532–1543, 2014. 22
- [47] Joulin, Armand, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Herve Jégou e Tomas Mikolov: *Fasttext.zip: Compressing Text Classification Models*. ArXiv, 2016. <http://arxiv.org/abs/1612.03651>. 22
- [48] Logpai: *Drain: Online log parsing*. <https://github.com/logpai/logparser/blob/master/logparser/Drain/README.md>, 2023. Accessed: 2023-02-13. 38

Apêndice A

Artigo Apresentado no SLATE 2024

Neste artigo, é apresentada uma análise comparativa de algoritmos não supervisionados para a detecção de anomalias em registros de *logs*, com o uso de técnicas de NLP para a identificação de falhas e vulnerabilidades em sistemas de TI. Destaca-se a eficácia de modelos de redes neurais aplicados em cenários reais de infraestrutura tecnológica.

- **Título:** *Anomaly Detection in Logs: A Comparative Analysis of Unsupervised Algorithms*
- **Conferência:** *13th Symposium on Languages, Applications and Technologies (SLATE 2024)*
- **Data:** 4 a 5 de julho de 2024
- **Local:** Universidade de Aveiro, ESTGA, Águeda, Portugal
- **Identificador de Objeto Digital (DOI):** [10.4230/OASlcs.SLATE.2024.12](https://doi.org/10.4230/OASlcs.SLATE.2024.12)

Anomaly Detection in Logs: A Comparative Analysis of Unsupervised Algorithms

Alysson C. E. de Moura  

Department of Computer Science, University of Brasília (UnB), Brazil

Geraldo P. Rocha Filho  

Department of Computer Science, State University of Southwest Bahia (UESB), Brazil

Marcos F. Caetano  

Department of Computer Science, University of Brasília (UnB), Brazil

João J. C. Gondim  

Department of Computer Science, University of Brasília (UnB), Brazil

Aleteia Araujo  

Department of Computer Science, University of Brasília (UnB), Brazil

Marcelo A. Marotta  

Department of Computer Science, University of Brasília (UnB), Brazil

Lucas Bondan  

Rede Nacional de Ensino e Pesquisa (RNP), Brasília, Brazil

Department of Computer Science, University of Brasília (UnB), Brazil

Abstract

This study explores anomaly detection through unsupervised Machine Learning applied to banking systems' log records. The diversity in formatting and types of logs poses significant challenges for automating anomaly detection. We propose a workflow using Natural Language Processing (NLP) techniques for anomaly identification, which in further analysis can lead to identifying root causes of failures and vulnerabilities. We evaluate the performance of eight different models using Blue Gene/L log records. The most effective models were selected and subsequently validated with Microsoft Configuration Manager (MCM) logs collected from a financial institution, demonstrating their practical applicability in real-world scenarios. Experimental results highlighted the effectiveness of neural network models, specifically Self-Organizing Maps (SOM) and Autoencoders (AE), with F1-Scores of 0.86 and 0.80, respectively, when applied to MCM logs collected from the financial institution.

2012 ACM Subject Classification Computing methodologies → Artificial intelligence; Computing methodologies → Natural language processing; Computing methodologies → Machine learning

Keywords and phrases Anomaly Detection, Log Analysis, Natural Language Processing, Unsupervised Learning, Word Embeddings

Digital Object Identifier 10.4230/OASICS.SLATE.2024.12

1 Introduction

The fast identification of anomalies in logs is crucial to help discover the root causes of failures in computer systems, especially in the financial sector, which relies heavily on Information Technology (IT) systems. This sector operates with critical systems that require high availability, reliability, and data security [8]. Failures or vulnerabilities in these systems can significantly affect services and the user experience, resulting in substantial losses for companies. Machine Learning (ML) has emerged as an essential tool for speeding up this process, mainly by applying anomaly detection techniques to system event records, known as logs. Automating these processes is vital for identifying and correcting problems quickly,



© Alysson C. E. de Moura, Geraldo P. Rocha Filho, Marcos F. Caetano, João J. C. Gondim, Aleteia Araujo, Marcelo A. Marotta, and Lucas Bondan;

licensed under Creative Commons License CC-BY 4.0

13th Symposium on Languages, Applications and Technologies (SLATE 2024).

Editors: Mário Rodrigues, José Paulo Leal, and Filipe Portela; Article No. 12; pp. 12:1–12:14

OpenAccess Series in Informatics



Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

especially considering the massive volume of data generated by these logs. The main purpose of system logs is to record critical states and events, facilitating debugging and root cause analysis. Therefore, it is essential to adopt intelligent methods for efficiently detecting anomalies in this data [1].

Log analysis faces significant challenges due to its unstructured nature and the large volume of data generated [28]. This paper focuses on detecting anomalies in log records, addressing the complexity arising from different formats and origins. The emphasis is on analyzing both the Blue Gene/L log records, which are widely used in the literature, and the `ccmeval.log` file from Microsoft Configuration Manager (MCM) [17], which are crucial for ensuring the proper functioning of the MCM agent installed on the financial institution's servers. The significant amount of data and the unstructured nature of the logs represent an obstacle in identifying problems in a variety of systems.

This study is motivated by the need to increase the availability of banking systems and minimize financial losses. Continuous, uninterrupted operation of these systems is essential to prevent significant losses. The implementation of ML models to automatically identify the root cause of failures allows for more agile solutions, preventing or mitigating the recurrence of problems.

The main contribution of this work is the comparative analysis of unsupervised ML techniques for detecting anomalies in log records. These techniques offer an automated approach to quickly identifying anomalies in computer systems by analyzing log records applied to a real financial scenario. In addition, the use of state-of-the-art Natural Language Processing (NLP) techniques is incorporated to extract semantic information from log records. Experimentation in real-world scenarios, using logs collected from a financial institution, demonstrates the practical applicability and relevance to critical environments.

The rest of this paper is organized as follows. Section 2 presents the background of the work and the main related works. Section 3 presents the proposal for detecting anomalies in logs. Next, the scenarios considered for experimentation are detailed in Section 4, and the results obtained are analyzed and discussed in Section 5. Finally, the conclusions of the work and prospects for future work are presented in Section 6.

2 Background and Related Work

This section explores several key concepts for anomaly detection in log data. We delve into methods for parsing raw logs, extracting relevant features, and converting them into suitable semantic vector representations. Finally, we discuss related research in the area.

2.1 Anomaly Detection

Anomaly detection aims to identify events in data sets that do not follow the expected pattern of normal behavior. This technique covers a variety of applications, including bank fraud detection, system intrusion prevention, among others. Unexpected events, called anomalies, can be categorized into three main groups [3]: **punctual anomaly**, which refers to an isolated data entry that is significantly different from the rest; **contextual anomaly**, occurring in a specific context and making a record abnormal within that broader context; and **collective anomaly**, which involves groups of records that, although not individually anomalous, stand out together due to distinct characteristics when compared to the rest of the data.

2.1.1 Machine Learning

Machine Learning (ML) is a branch of Artificial Intelligence that focuses on developing algorithms to learn patterns and perform tasks without explicit programming, having the ability to improve performance in the execution of some tasks through experience. ML techniques are classified into four approaches: **Supervised** (training with labeled data), **Unsupervised** (identifying patterns without labels), **Semi-Supervised** (combining labeled and unlabeled data) and **Reinforcement** (reward-based decisions) [27].

For anomaly detection, [19] identified 28 ML techniques, divided into six categories (classification, ensemble, optimization, rule system, clustering, and regression). From these, the following unsupervised approaches were selected for evaluation in this work due to their wide use and demonstrated effectiveness:

- **Local Outlier Factor (LOF)**, proposed by [2], is a data analysis technique that identifies outliers by comparing the density of a data point to the density of its surrounding neighbors. Its approach centered on local properties allows anomalies to be identified in specific areas of the data, although its interpretation requires attention due to its sensitivity to parameters, such as the value of k ;

- **Autoencoder (AE)**, initially proposed by [23], is an unsupervised neural network with an encoder and a decoder. The encoder compresses input data into a lower-dimensional latent representation, and the decoder reconstructs it. Training minimizes reconstruction error, helping the model learn relevant features. In use, the encoder generates the latent representation, and the decoder reconstructs the data, making Autoencoders effective for anomaly detection by highlighting reconstruction deviations;

- **Self-Organizing Map (SOM)**, also known as the Kohonen network [12], is a type of neural network that projects high-dimensional data onto a lower-dimensional grid, simplifying visualization. It is trained using unsupervised competitive learning and is notable for its efficiency in handling large sets of non-linear data while preserving the topological properties of the input space;

- **Isolation Forest (IF)**, proposed by [14], is an algorithm based on decision trees that detects anomalies by isolating outlier data points through random partitioning of the data. Its scalability makes it suitable for large data sets;

- **Minimum Covariance Determinant (MCD)** [10] is a robust algorithm for estimating the covariance matrix. It identifies subsets of data that are less susceptible to outliers, minimizing the covariance between observations and removing outliers;

- **Histogram-based Outlier Score (HBOS)** [5] uses univariate histograms to identify outliers. It is fast, easy to interpret, and versatile, adapting well to different types of data;

- **Cluster-based Local Outlier Factor (CBLOF)** [7] builds upon LOF by incorporating the data's clustering structure. This allows CBLOF to excel at identifying anomalies within clusters, making it particularly effective for anomaly detection in clustered data;

- **Gaussian Mixture Model (GMM)** [22] is a probabilistic model that describes the probability distribution as a mixture of multiple Gaussian (normal) distributions. It combines densities from different components and is useful for modeling complex data and identifying patterns.

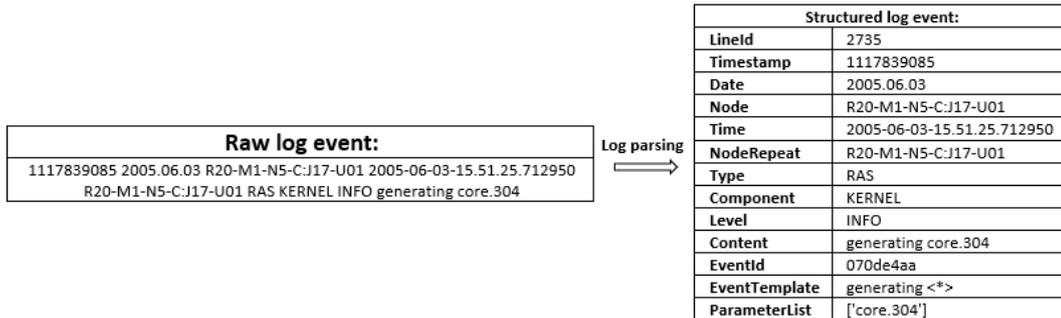
2.2 Log records and Log Parsing

Log records consist of data that is usually unstructured and has a time sequence format. Although they can take various forms, these records usually contain a timestamp and a raw log. The raw log describes a recorded event and is typically made up of two distinct parts: a

12:4 Anomaly Detection in Logs: A Comparative Analysis of Unsupervised Algorithms

constant (log template) and a variable [6].

Log parsing is the initial stage of pre-processing, where the raw logs are structured and templates are extracted. Figure 1 shows the result of a raw log parsing. In this example, to identify anomalies, the attribute used to extract the template was the ‘*Content*’ column, which contains the log event message. The ‘*Content*’ column represents the main body of the log entry, detailing the specific event that occurred. This resulted in the template ‘*generating <*>*’, which was stored in the ‘*EventTemplate*’ column. The token ‘*<*>*’ is the wildcard for replacing the parameters that are extracted and included in the ‘*ParameterList*’ column.



■ **Figure 1** Example of log parsing.

2.3 Feature Extraction

In anomaly detection, feature extraction is crucial for preparing data. In this work, we highlight the following techniques [28]:

- **Event Grouping or Windowing:** this technique aims to identify unusual patterns involving multiple events, such as variations in the sequence of events or temporal correlations in the log records. In these situations, it is essential to logically organize the events into groups, often called log sequences, which will be analyzed individually or associated with each other. There are three types of windowing in the literature: **sliding window**, in which windows of a specific duration move along the log events with a fixed step size; **fixed window**, with the step size set equal to the window size; and **session window**, which depends on an event parameter as a session identifier;

- **Word Embedding:** in the field of NLP, the Word Embedding technique converts words into vectors of real values, allowing algorithms to interpret the semantic meaning and relationships between words during text analysis [26]. Obtaining word embeddings usually involves training language models on extensive sets of textual data. These representations simplify text manipulation by machines and enhance the performance of ML models in specific tasks;

- **TF-IDF:** Term Frequency-Inverse Document Frequency is a technique widely used to assign weights to terms in various applications, such as text mining and document categorization [24]. This approach involves assigning weights to words based on the frequency with which they occur in a specific document (TF) and their rarity regarding all the documents analyzed (IDF), highlighting words that are frequent in a specific document but rare in the entire set of documents;

2.4 Semantic Vectorization

Many anomaly detection approaches in log events sequences are limited, requiring models based on known events. To improve robustness in detecting anomalies in log records, [28] applied Semantic Vectorization, in which the authors extracted semantic information from log events, representing them as semantic vectors. By converting log messages into vectors that capture the meaning and context of the events, this approach enhances the accuracy and effectiveness of anomaly detection models. Semantic vectorization is performed individually for each log sequence window, resulting in sequences of semantic vectors that are later used to feed anomaly detection models.

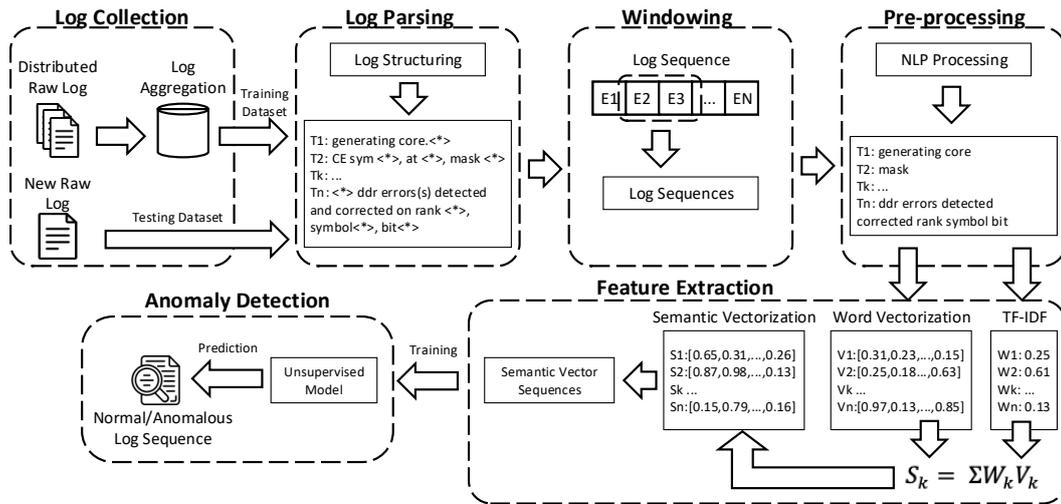
2.5 Related Work

Several studies address the use of anomaly detection models in logs. Du et al. [4] proposed DeepLog, a deep learning model using Long Short-Term Memory (LSTM) to model logs as natural language sequences. DeepLog automatically learns normal execution patterns and detects anomalies when these patterns are violated, incorporating user feedback. LogRobust [28] uses a supervised binary classifier on sequences of log events. In this way, it extracts semantic information from log events and represents them as semantic vectors. Using attention-based Bi-LSTM, LogRobust identifies and handles unstable log events and sequences, achieving more accurate and robust detection. HitAnomaly [9] uses a semantic-based approach that employs a hierarchical transform structure and uses an attention mechanism as the final classification model to deal with different types of anomalies in log records. Studiawan et al. [25] proposed a Deep Learning-based sentiment analysis technique to verify anomalous activity in operating system logs, using Gated Recurrent Unit (GRU) to identify sentiment in log messages. LAnoBERT [13] presents a log anomaly detection system using the BERT model without relying on the log parsing stage. This system uses unsupervised learning and masked language modeling. As described in [16], CNNs are applied to detect anomalies in logs by performing time series analysis. By using log messages grouped into time series, CNNs can learn the relationships between log messages and identify anomalies.

While previous research has shown promising unsupervised anomaly detection methods, their evaluations were often conducted on unique datasets. This makes directly comparing their performance challenging. This paper addresses this gap by evaluating several unsupervised models on the same datasets across various environments. This comprehensive approach provides a clearer understanding of how these models perform in real-world applications. Through this comparative analysis, we aim to identify the most effective unsupervised anomaly detection solutions for practical use.

3 Proposed Method: Workflow for Anomaly Detection in Log Records

For anomaly detection in log records, the use of a workflow based on the best works in the literature is proposed. This section covers the proposed workflow, including log collection, log parsing, windowing, pre-processing, feature extraction, and anomaly detection. To carry out the experiments, a workflow of operations to accommodate unsupervised ML models is proposed, testing them in different scenarios. As a result, eight of the main unsupervised learning techniques were implemented and evaluated in two different scenarios. Figure 2 depicts the six stages in detail.



■ **Figure 2** Proposed workflow for anomaly detection in log files.

3.1 Log Collection

Our workflow starts with a distributed Log Collection phase. Here, we gather system logs from various machines to identify potential anomalies. There are different approaches to achieve this, but the most common method involves deploying a lightweight log collection agent on each machine. These agents efficiently transmit relevant log records to a central log storage server, where the logs are aggregated. This aggregated data serves as the training dataset for our anomaly detection models.

3.2 Log Parsing

After the Log Collection phase, Log Parsing structures the logs and the event message templates are extracted. Two notable Log Parsing methods stand out in the literature: Drain and Spell [29]. The Drain method was chosen due to its widespread adoption in the literature and its proven effectiveness in accurately parsing log messages into structured templates. Drain, a real-time log parser, stands out for converting logs continuously and accurately. Based on an analysis tree with a fixed depth, Drain uses simple regular expressions, guided by domain knowledge, to process messages. Its ability to group records into clusters facilitates the interpretation and extraction of information.

3.3 Windowing

This phase extracts sequences of logs using configurable time windows. Two key parameters influence this process: Window Size and Step Size. When the step size equals the window size, fixed-length log sequences are extracted. We opt for fixed or sliding windows (as opposed to session windows) due to the inherent characteristic of log data. Logs often lack clear markers to define sessions definitively. Additionally, fixed and sliding windows offer advantages in computational efficiency and adaptability to general temporal patterns within the logs. This translates to a more efficient and interpretable approach for anomaly detection.

3.4 Pre-Processing

After Windowing has been completed, the pre-processing of log messages is essential to prepare data for analysis and mining. The quality of the log data directly affects the effectiveness of subsequent analysis, as noisy or incomplete data can lead to incorrect anomaly detection. Pre-processing includes:

- Removal of unwanted characters to clean up the text and focus on relevant keywords;
- Elimination of stopwords to reduce noise in log events;
- Exclusion of short words (with less than 3 characters) to preserve semantic relevance; 3 characters was chosen after preliminary tests showed minimal impact on results;
- Separation of compound tokens in Camel Case to improve text readability;
- Lemmatization as a normalization technique to reduce words to their lemma, preserving the original meaning.

If there is no log event after pre-processing, ‘OOV’ (Out-of-Vocabulary) is returned for words that have no representation or to indicate the absence of data. The pipeline was also tested without excluding short words to ensure important elements, such as negations, were not removed. However, excluding short words generally improve the quality of log data for anomaly detection.

3.5 Feature Extraction

Following Pre-Processing, the feature extraction stage transforms log messages into sequences of semantic vectors, capturing their meaning. This process involves three key steps:

- Word Vectorization: We convert individual words in the log messages into numerical vectors, allowing us to represent their meaning numerically;
- TF-IDF Weighting: We assign weights to each word based on Term Frequency-Inverse Document Frequency (TF-IDF). This emphasizes words that are important within a specific log message but less common across all messages;
- Semantic Vector Creation: We combine the word vectors with their corresponding TF-IDF weights, creating semantic vectors that encapsulate the meaning of each log message.

By combining these vectors sequentially, we obtain Semantic Vector Sequences. An overview of the feature extraction techniques utilized in this study is provided in Section 2.3.

This work utilizes word embedding techniques, a powerful approach for representing the semantic meaning of words in natural language. Several techniques exist, including Word2Vec [18], GloVe [21], and FastText [11]. Word2Vec efficiently learns dense vector representations of words, capturing semantic relationships and context. GloVe combines global word statistics with machine learning to create vectors that capture both semantic and syntactic information. FastText, developed by Facebook, focuses on character n-grams. This makes it particularly effective for handling rare words and languages with complex morphology. In this study, we leverage a pre-trained GloVe model, a well-established choice due to its documented performance in various applications.

Our decision to incorporate TF-IDF in the feature extraction process is driven by both empirical evidence and prior research, as demonstrated in [28]. TF-IDF proves particularly advantageous in environments where log message stability fluctuates significantly. It assigns weights to words based on their importance within a specific log message, but also considers their rarity across the entire dataset. This approach helps mitigate the impact of potential shifts in term distribution over time. Consequently, TF-IDF enhances the system’s robustness to variations and noise inherent in log data.

3.6 Anomaly Detection

In the final phase, we classify log sequences as “normal” or “anomalous” by training on the dataset collected during the Log Collection stage and making predictions on new log records. The choice of unsupervised models is motivated by the complexity of obtaining event labels in real-world environments. This work focuses on the detection of contextual anomalies, considering the temporal interdependence between events in log sequences. The primary motivation for selecting unsupervised models lies in their ability to operate without labeled data, which is often scarce or unavailable in dynamic environments. Unsupervised models overcome the challenges of scarce labeled data, allowing more flexible anomaly detection without explicitly relying on labeled examples. These models adapt to the complexity and variability of real-world environments, offering an effective solution to the difficulties in obtaining accurate labels.

4 Evaluation

This section presents the evaluation of the proposed method, including the metrics used, the scenarios for BGL and CCMEval logs, and the experimental setup.

4.1 Metrics

Anomaly detection can be seen as a binary classification problem. The following metrics were used to evaluate the performance of the models:

- **Precision (P)**: the percentage of true anomalous sequences detected among all anomalous sequences detected. $P = \frac{TP}{TP+FP}$;
- **Recall (R)**: the percentage of true anomalous sequences detected among all existing anomalous sequences. $R = \frac{TP}{TP+FN}$;
- **F1-Score (F1)**: harmonic mean of Precision and Recall $F1 = \frac{2 \cdot P \cdot R}{P+R}$;
- **Receiver Operating Characteristic (ROC) Curve**: graph showing the performance of a classification model, indicating the true positive rate as a function of the false positive rate at different cut-off points. The AUC (Area Under the Curve) is an associated metric, evaluating the overall capacity of the model, ranging from 1 (ideal) to 0.5 (random).

TP is the number of anomalous sequences correctly detected, FP is the number of normal sequences erroneously detected as anomalous and FN is the number of abnormal log sequences erroneously detected as normal.

4.2 Scenario 1 - BGL

BGL is an open dataset initially presented by [20], containing logs collected from a Blue-Gene/L supercomputer. The logs encompass messages categorized into various alert levels, including those without alerts. The BGL dataset was chosen due to its comprehensive and diverse log messages, which include well-documented anomalies. This diversity and the presence of detailed anomalies make it ideal for evaluating anomaly detection models in a realistic and challenging environment, thereby ensuring the robustness and applicability of the models.

The original BGL dataset has a total of 4,747,963 messages, of which 4,399,503 are normal and 348,460 anomalous (7.34%), collected over 215 days and occupying 708MB. For our experiments, a sample of the first 2 million lines was selected, containing 1,771,267 normal

records and 228,733 anomalous ones (11.44%). This choice balances representativeness and computational efficiency while maintaining temporal diversity and variability.

Among the 2 million lines, log sequences were selected for each window size, randomly and without replacement. Of these sequences, 75% were used for training and 25% for testing. In addition, the null vectors of the data were removed. Table 1 summarizes data.

Window Size	Total Sequences	Training (75%)/Testing (25%)
10 seconds	99,986	74,989/24,997
30 seconds	39,994	29,995/9,999
1 minute	19,997	14,997/5,000
2 minutes	13,331	9,998/3,333
3 minutes	9,998	7,498/2,500

■ **Table 1** Summarized BGL Data

4.3 Scenario 2 - CCMEval

CCMEval is a log from MCM, a system management tool developed by Microsoft to manage large groups of computers. This log makes it possible to identify anomalies, pointing out possible failures and vulnerabilities in the MCM agents installed in the endpoints of the IT infrastructure. Detailed analysis of CCMEval provides valuable information for detecting and correcting problems, ensuring that the integrity and security of the financial institution's systems are maintained. The second scenario uses CCMEval logs collected from a financial institution as a real-world usage scenario to validate the most effective models identified in the Blue Gene/L experiments.

The data used to train the models was obtained from five servers operating normally (without failures) for eight months, resulting in a total of 31,696 messages after aggregation. In partnership with Microsoft experts, "Ground Truth" was established to identify anomalies in the CCMEval log, ensuring maximum reliability in evaluating the results. To test the models, we collected logs from four servers over five months, where failures were mapped in collaboration with experts. This resulted in 9,261 messages after aggregation, of which 236 were anomalous (2.55%).

Anomalies identified by Microsoft experts for "Ground Truth":

1. Instance (*) of class (SMS_Client) does not exist (0x80041003)
2. Checking WMI repository for feature Provider failed
3. WMI check failed
4. Result: Failed, ResultCode: 504, ResultType: 202, ResultDetail: *#CCMEVALPARAMSEP#SMS_Client#CCMEVALPARAMSEP#0x80041003
5. Result: Remediation Failed, ResultCode: -2147023843
6. Updating MDM_ConfigSetting.ClientHealthStatus with value 3
7. Updating MDM_ConfigSetting.ClientHealthStatus with value 7
8. Failed to get MDM_ConfigSetting instance, 0x80041013
9. Client health check: FAILED
10. Failed to send client health status as a state message. Attempting to send status to the FSP.
11. Attempting to change service status for service 'CmRcService' to 'Running'.

12:10 Anomaly Detection in Logs: A Comparative Analysis of Unsupervised Algorithms

Of the 31,696 messages obtained through aggregation for training, 2,175 log sequences were generated after excluding the null vectors. The test data set, made up of 9,261 messages, resulted in 578 log sequences after removing the null vectors. Table 2 summarizes the data.

Window Size	Total Sequences	Training (79%)/Testing (21%)
10 seconds	2,753	2,175/578

■ **Table 2** Summarized CCMEval Data

4.4 Experimental Setup

Experiments were conducted on Windows 11, using Python 3.11 on a machine with an Intel Core i7 processor (2.40GHz, 4 cores) and 20GB of RAM. The Drain log parser from the Logpai GitHub repository [15] was used for analysis. The hyperparameters followed standard values, with a particular emphasis on the adjustable contamination rate for improved adaptability in detecting anomalies. The contamination rate, an important hyperparameter in unsupervised anomaly detection, represents the proportion of actual anomalies present in the dataset. It significantly influences the sensitivity of the models to detect anomalies.

5 Results

This section discusses the results obtained from the experiments, focusing on the performance of models in the BGL and CCMEval scenarios.

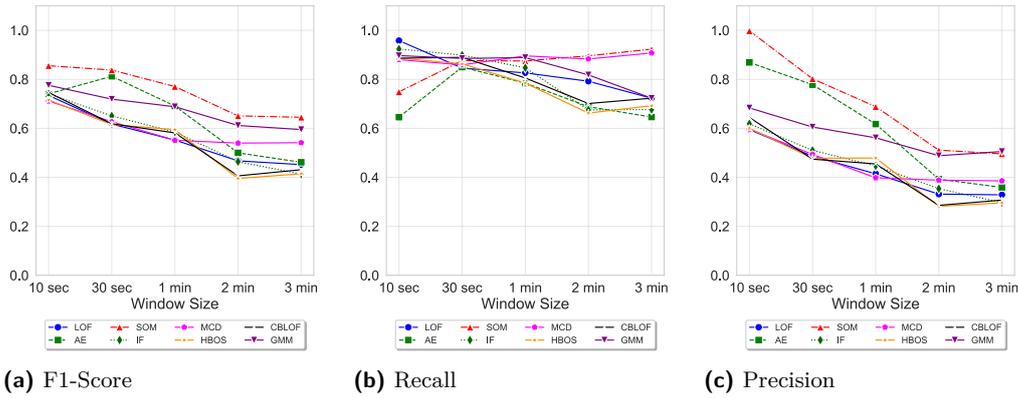
5.1 Scenario 1 - BGL

Two experiments were designed to evaluate the selected models in the BGL scenario. First, we analyze the performance of the models using time windows of fixed sizes: 10 seconds, 1 minute, and 3 minutes using the 50-dimensional Glove. The aim is to analyze how the size of the window affects the performance of the models. Then, in the second experiment, we analyze the sensitivity of the models to variations in the contamination rate.

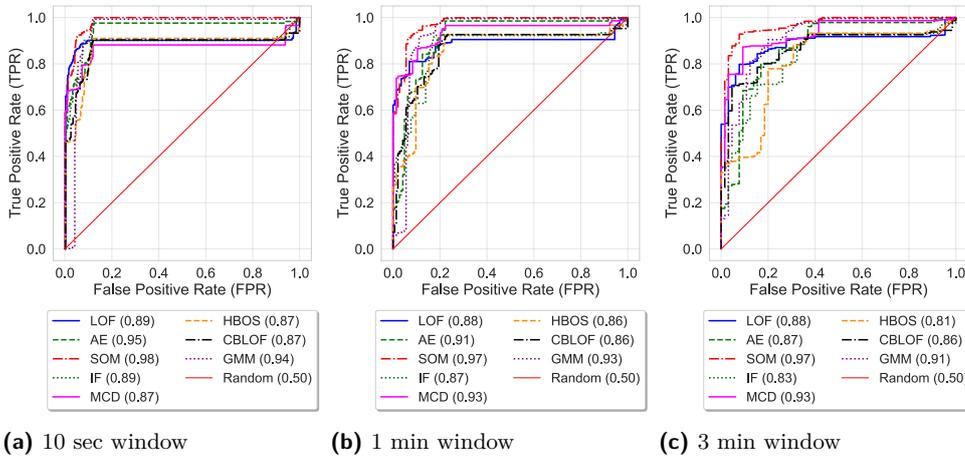
5.1.1 Comparison of F1-Score, Recall, Precision and ROC curve

Figures 3a, 3b and 3c show F1-Score, Recall and Precision, respectively. The ROC curves (Figures 4a, 4b and 4c) illustrate the performance of the models using different window sizes (10 seconds, 1 minute and 3 minutes, respectively). In general, there is a drop in the F1-Score as the time window increases, mainly due to a decrease in Precision. The main reason for this drop is that the larger the window, the more log events tend to be captured. Since anomalies are contextual, a greater number of captured events means having a wider context. This broadening of the context, however, can lead to a dilution of anomalies in the midst of normal events, which makes it more challenging for the model to accurately identify these anomalies. The complexity of the problem also increases with larger time windows, which can result in a greater number of false positives and negatively affect Precision. The ROC curves and respective AUC illustrated in Figure 4 confirm that anomaly detection worsens as the size of the time window increases.

Examining the F1-Score, Recall, Precision metrics, and the ROC curve together, we observe that the SOM and AE models perform better as the window size increases, especially due to their greater robustness against false positives, as illustrated mainly in Figure 3c.



■ **Figure 3** Performance of models with fixed time windows in BGL.



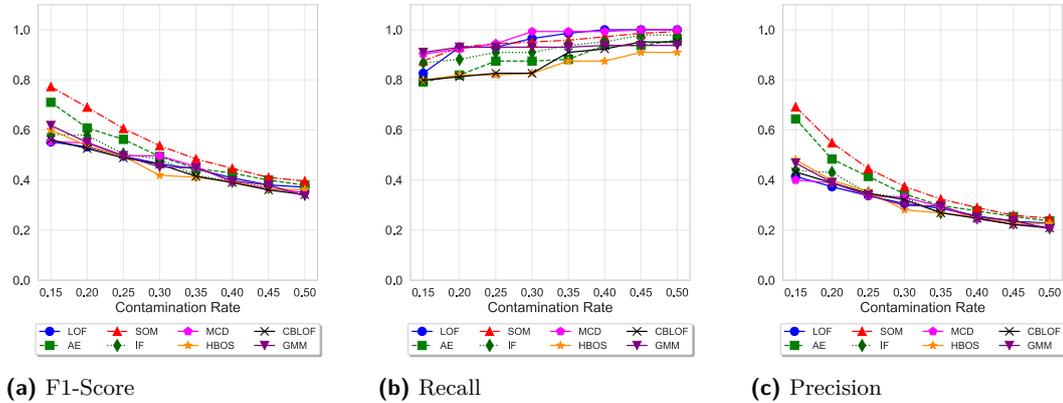
■ **Figure 4** ROC curve of the models with fixed time windows in the BGL.

This superiority can be attributed to the nature of neural network models, which have an intrinsic ability to identify and represent complex patterns in data more effectively, especially in high-dimensional spaces. In addition, SOMs and AEs stand out for their ability to reduce dimensionality, tolerate noise, and handle incomplete inputs, thus contributing to greater accuracy and generalization. However, it is important to note that the robustness and accuracy of these models depend not only on the nature of the task but also on the quality of the data. Therefore, careful evaluation is essential not only to determine their effectiveness in different contexts but also to understand how they might behave concerning the quality of the data specific to each situation.

5.1.2 Sensitivity of Models to Variation in Contamination Rate

In this experiment, we analyze the sensitivity of the models to variations in the contamination rate. The results illustrated in Figure 5c show that, in general, Precision decreases as the contamination rate increases, due to the increase in false positives, as the algorithms incorrectly assume the presence of more anomalous instances, leading to the misclassification of normal instances. The Recall (Figure 5b) increases slightly and stabilizes as the contamination rate increases, indicating an improvement in the identification of anomalous instances, but

with accuracy stabilizing at around 45%. The F1-Score (Figure 5a) decreases due to the decrease in Precision, reflecting the interaction between Precision and Recall. Thus, the SOM and AE models also stand out in this second experiment, corroborating the observations made in the first experiment.



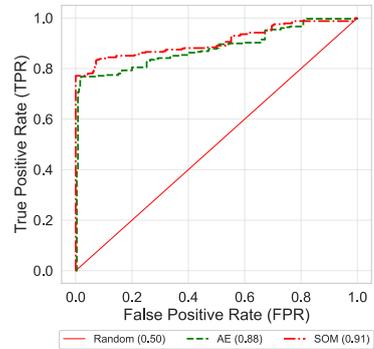
■ **Figure 5** Performance with increasing contamination rate in the BGL log.

5.2 Scenario 2 - CCMEval

To evaluate the performance of neural network models on realistic data, we analyzed results considering the CCMEval log. We focused on AE and SOM since they presented the best results in the prior BGL scenario. Table 6a summarizes the evaluation metrics, while Figure 6b presents the ROC curve. We employed a sliding window approach for analysis, using a 10-second window size and a 2-second step size.

Model	P	R	F1
AE	0.75	0.86	0.80
SOM	0.81	0.93	0.86

(a) Performance Metrics



(b) ROC Curve

■ **Figure 6** Model Performance - Scenario 2 (CCMEval).

The experiments carried out show that, among the models evaluated, neural network models, including AE and SOM, are the most suitable for anomaly detection on the servers of the financial institution presented as a scenario. Both AE and SOM models effectively detected anomalies in the CCMEval log, achieving metrics above 0.75 for all evaluated criteria. AE demonstrated a balanced performance with a Precision of 0.75, Recall of 0.86, and F1-Score of 0.80. SOM excelled at identifying true anomalies with a high Recall of 0.93, while maintaining a Precision of 0.81 and an F1-Score of 0.86. Further supporting these

findings, the ROC curves for AE and SOM yielded Area Under the Curve (AUC) values of 0.88 and 0.91, respectively. These results suggest that both neural network models hold promise for real-world log anomaly detection, offering a robust and effective approach in the CCMEval scenario.

6 Conclusions and Future Work

This section presents the conclusions of this study and discusses prospects for future research. This study investigated the potential of combining unsupervised ML and NLP techniques for anomaly detection in log data, aiming to identify root causes of system failures and vulnerabilities. We propose a workflow that leverages semantic vectorization of log message sequences, drawing inspiration from established research in the field. Eight widely adopted anomaly detection models were evaluated across two distinct scenarios: the BGL log dataset and real-world CCMEval logs collected from a financial institution. Notably, the neural network models, Autoencoder (AE) and Self-Organizing Map (SOM), demonstrated superior performance, suggesting their suitability for practical applications.

In future work, we plan to evaluate the models under various configurations and operating conditions to better understand their performance and generalizability, which is essential for assessing the feasibility of deploying an automated anomaly detection system based on our workflow. Additionally, we aim to explore threat identification and automated countermeasure generation by analyzing the context and risks associated with detected anomalies, leading to a comprehensive and proactive response system.

References

- 1 Mohiuddin Ahmed, Abdun Naser Mahmood, and Md Rafiqul Islam. A Survey of Anomaly Detection Techniques in Financial Domain. *Future Generation Computer Systems*, 55:278–288, 2 2016.
- 2 Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. LOF: Identifying Density-Based Local Outliers. *ACM Special Interest Group on Management of Data*, 29:93–104, 2000. doi:10.1145/335191.335388.
- 3 Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly Detection: A Survey. *ACM Computing Surveys*, 41(3):1–58, 2009.
- 4 Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. *ACM Special Interest Group on Security, Audit and Control*, pages 1285–1298, 2017.
- 5 Markus Goldstein and Andreas Dengel. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. *German Conference on Artificial Intelligence: Poster and Demo Track*, 1:59–63, 2012.
- 6 Pinjia He, Jieming Zhu, Shilin He, Jian Li, and Michael R Lyu. An evaluation study on log parsing and its use in log mining. *IEEE/IFIP international conference on dependable systems and networks*, pages 654–661, 2016.
- 7 Zengyou He, Xiaofei Xu, and Shengchun Deng. Discovering Cluster-based Local Outliers. *Pattern Recognition Letters*, 24(9-10):1641–1650, 2003.
- 8 Shaohan Huang, Carol Fung, Kui Wang, Polo Pei, Zhongzhi Luan, and Depei Qian. Using Recurrent Neural Networks Toward Black-Box System Anomaly Prediction. *IEEE/ACM 24th International Symposium on Quality of Service*, pages 1–10, 2016.
- 9 Shaohan Huang, Yi Liu, Carol Fung, Rong He, Yining Zhao, Hailong Yang, and Zhongzhi Luan. Hitanomaly: Hierarchical Transformers for Anomaly Detection in System Log. *IEEE Transactions on Network and Service Management*, 17(4):2064–2076, 2020.

- 10 Mia Hubert and Michiel Debruyne. Minimum covariance determinant. *Wiley interdisciplinary reviews: Computational statistics*, 2(1):36–43, 2010.
- 11 Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Herve Jégou, and Tomas Mikolov. Fasttext.zip: Compressing Text Classification Models. *arXiv preprint arXiv:1612.03651*, 2016.
- 12 Teuvo Kohonen. The Self-organizing Map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.
- 13 Yukyung Lee, Jina Kim, and Pilsung Kang. Lanobert: System Log Anomaly Detection Based on BERT Masked Language Model. *Applied Soft Computing*, 146:1–14, 2023.
- 14 Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation Forest. *IEEE International Conference on Data Mining*, pages 413–422, 2008.
- 15 Logpai. Drain: Online log parsing. <https://github.com/logpai/logparser/blob/master/logparser/Drain/README.md>, 2023. Accessed: 2023-02-13.
- 16 Siyang Lu, Xiang Wei, Yandong Li, and Liqiang Wang. Detecting anomaly in big data system logs using convolutional neural network. *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, pages 151–158, 2018.
- 17 Microsoft. *Microsoft System Center Configuration Manager*, 2024. Accessed: 2023-07-21. URL: <https://docs.microsoft.com/en-us/mem/configmgr/>.
- 18 Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*, 2013.
- 19 Ali Bou Nassif, Manar Abu Talib, Qassim Nasir, and Fatima Mohamad Dakalbab. Machine Learning for Anomaly Detection: A Systematic Review. *IEEE Access*, 9:78658–78700, 2021.
- 20 Adam Oliner and Jon Stearley. What Supercomputers Say: A Study of Five System Logs. *IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 575–584, 2007.
- 21 Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. *Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, 2014.
- 22 Douglas A Reynolds et al. Gaussian mixture models. *Encyclopedia of biometrics*, 741(659-663), 2009.
- 23 David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning Internal Representations by Error Propagation. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*, pages 318–362, 1985.
- 24 Gerard Salton and Christopher Buckley. Term-weighting Approaches in Automatic Text Retrieval. *Information Processing & Management*, 24(5):513–523, 1988.
- 25 Hudan Studiawan, Ferdous Sohel, and Christian Payne. Anomaly Detection in Operating System Logs with Deep Learning-based Sentiment Analysis. *IEEE Transactions on Dependable and Secure Computing*, 18(5):2136–2148, 2020.
- 26 Virginia Teller. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. *Computational Linguistics*, 26:638–641, 2000.
- 27 Song Wang, Juan Fernando Balarezo, Sithamparanathan Kandeepan, Akram Al-Hourani, Karina Gomez Chavez, and Benjamin Rubinstein. Machine Learning in Network Anomaly Detection: A Survey. *IEEE Access*, 9:152379–152396, 2021.
- 28 Xu Zhang, Yong Xu, Qingwei Lin, Bo Qiao, Hongyu Zhang, Yingnong Dang, Chunyu Xie, Xinsheng Yang, Qian Cheng, Ze Li, et al. Robust Log-based Anomaly Detection on Unstable Log Data. *ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 807–817, 2019.
- 29 Jieming Zhu, Shilin He, Jinyang Liu, Pinjia He, Qi Xie, Zibin Zheng, and Michael R. Lyu. Tools and Benchmarks for Automated Log Parsing. *IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice*, pages 121–130, 2019.