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), Vitória da Conquista, 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, 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].

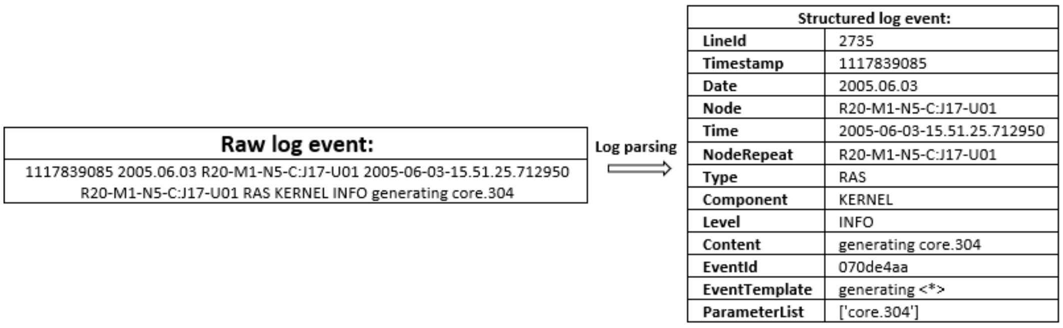
For anomaly detection, [19] identified 28 ML techniques, divided into six categories (classification, ensemble, optimization, rule system, clustering, and regression). From these, the following unsupervised approaches were selected for evaluation in this work due to their wide use and demonstrated effectiveness:

- **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 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.



```
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
```

Structured log event:	
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']

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.

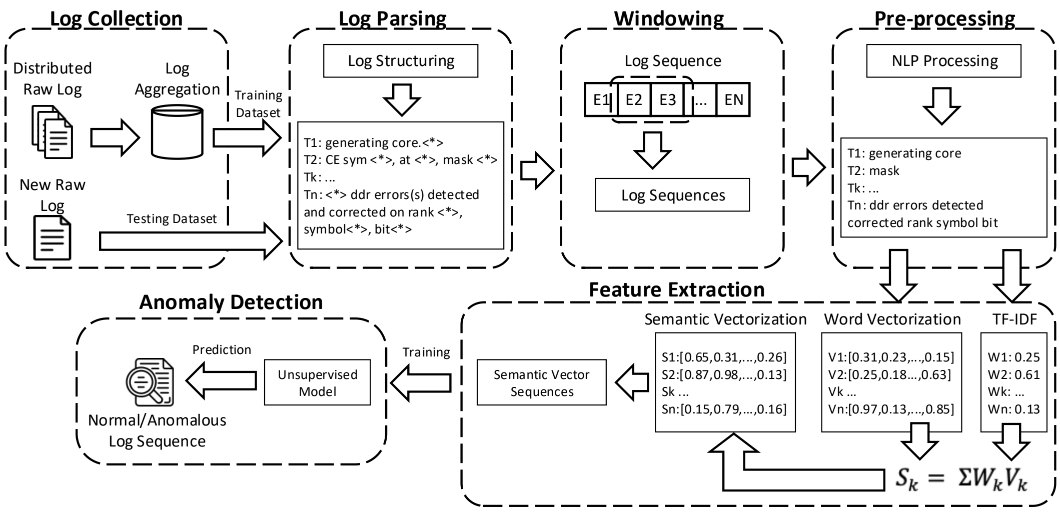


Figure 2 Proposed workflow for anomaly detection in log files.

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;
- Lematization 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 = $\frac{2 \cdot \mathrm{P} \cdot \mathrm{R}}{\mathrm{P}+\mathrm{R}}$;
- **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.

Table 1 Summarized BGL 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

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".

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.

Table 2 Summarized CCMEval Data.

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

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.

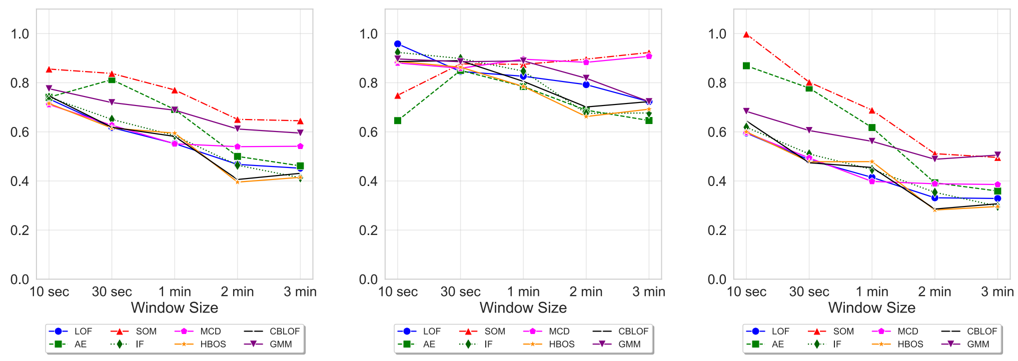
5.1 Scenario 1 – BGL

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

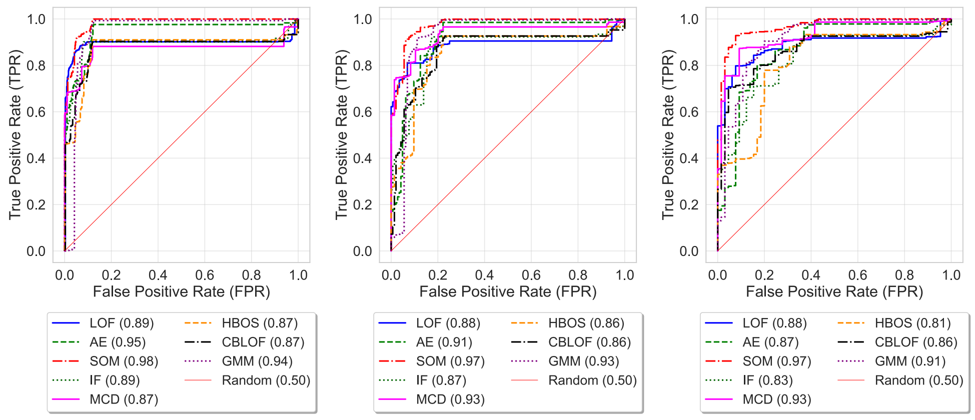
Figures 3a, 3b and 3c show F1-Score, Recall and Precision, respectively. The ROC curves (Figures 4a, 4b and 4c) illustrate the performance of the models using different window sizes (10 seconds, 1 minute and 3 minutes, respectively). In general, there is a drop in the F1-Score as the time window increases, 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. 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



(a) F1-Score. **(b)** Recall. **(c)** Precision.

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



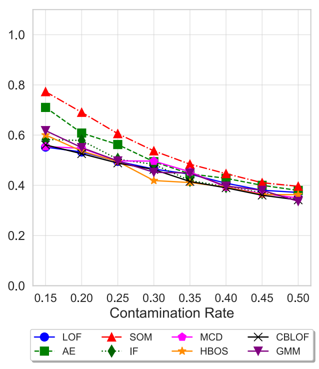
(a) 10 sec window. **(b)** 1 min window. **(c)** 3 min window.

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

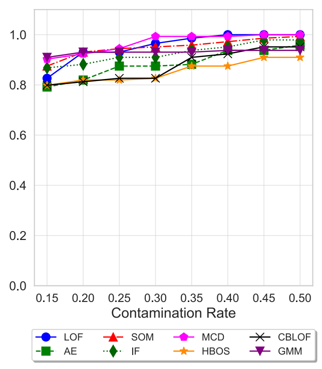
accuracy and generalization. However, it is important to note that the robustness and accuracy of these models depend not only on the nature of the task but also on the quality of the data. Therefore, careful evaluation is essential not only to determine their effectiveness in different contexts but 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

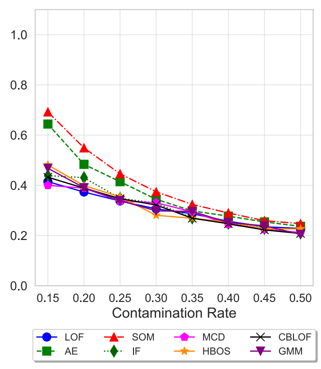
In this experiment, we analyze the sensitivity of the models to variations in the contamination rate. The results illustrated in Figure 5c show that, in general, Precision decreases as the contamination rate increases, due to the increase in false positives, as the algorithms incorrectly assume the presence 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.



(a) F1-Score.



(b) Recall.



(c) Precision.

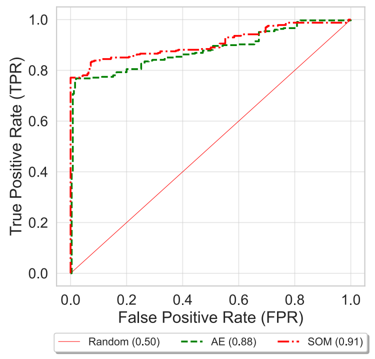
Figure 5 Performance with increasing contamination rate in the BGL log.

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, February 2016. doi:10.1016/J.FUTURE.2015.01.001.

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. doi:10.1145/1541880.1541882.

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. doi:10.1145/3133956.3134015.

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. doi:10.1109/DSN.2016.66.

7 Zengyou He, Xiaofei Xu, and Shengchun Deng. Discovering Cluster-based Local Outliers. *Pattern Recognition Letters*, 24(9-10):1641–1650, 2003. doi:10.1016/S0167-8655(03)00003-5.

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. doi:10.1109/IWQOS.2016.7590435.

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. doi:10.1109/TNSM.2020.3034647.

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, Hérve Jégou, and Tomas Mikolov. Fasttext.zip: Compressing Text Classification Models. *arXiv preprint*, 2016. arXiv:1612.03651.

12 Teuvo Kohonen. The Self-organizing Map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990. doi:10.1109/5.58325.

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. doi:10.1109/ICDM.2008.17.

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. doi:10.1109/DASC/PICOM/DATACOM/CYBERSCITEC.2018.00037.

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*, 2013. arXiv:1301.3781.

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. doi:10.1109/ACCESS.2021.3083060.

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. doi:10.1109/DSN.2007.103.

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. doi:10.3115/V1/D14-1162.

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. doi:10.1016/0306-4573(88)90021-0.

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. doi:10.1109/TDSC.2020.3037903.

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. doi:10.1109/ACCESS.2021.3126834.

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.