LogRD: A Robustness-Enhanced Framework for Log Anomaly Detection with Dynamic Masking

Xiaowu Liang
Jinan University
Guangzhou, Guangdong, China
studyly@foxmail.com

Xianxia Zou*
Jinan University
Guangzhou, Guangdong, China
zouxianxia@163.com

Jie Ren
Jinan University
Guangzhou, Guangdong, China
rbott@stu2024.jnu.edu.cn

Zijiong Su
Jinan University
Guangzhou, Guangdong, China
muhuos@stu2024.jnu.edu.cn

Cenyu Zheng
Jinan University
Guangzhou, Guangdong, China
mikeez@stu2022.jnu.edu.cn

Weiwu Xu
Jinan University
Guangzhou, Guangdong, China
xww666@stu2022.jnu.edu.cn

***Abstract*—Log anomaly detection is vital for system reliability, yet existing methods struggle with high labeling costs or limited semantic adaptability. This paper proposes LogRD, a novel semi-supervised framework that leverages only normal logs for training. To address the "mask position dependency" prevalent in current masked language models, LogRD introduces a dual-masking strategy: dynamic masking during training and a unique odd-even masking scheme at inference. By strengthening semantic understanding and ensuring comprehensive sequence evaluation, LogRD demonstrates superior robustness. Experimental results on three benchmark datasets show that LogRD consistently outperforms state-of-the-art baselines in detection performance.**

***Index Terms*—log anomaly detection , semantic information , robustness , mask position dependency , dynamic masking**

I. Introduction

Logs capture detailed runtime information during system operations. Automated anomaly detection within these logs is crucial for early threat identification, enabling the proactive mitigation of potential security risks [1]. Compared to individual logs that reflect only partial states, log sequences capture the dynamic execution process of program logic, better revealing potential anomaly patterns and enhancing detection accuracy [2]. Log sequences are typically segmented by identifier or time window [3]. As system scale and complexity continue to increase, manual log analysis struggles to meet practical demands, motivating the development of automated anomaly detection methods. Early approaches predominantly relied on traditional machine learning techniques such as SVM, PCA, and Isolation Forest [4]–[6], but they typically depend on manual feature design and fail to adequately capture the temporal characteristics of logs. In recent years, Deep learning approaches have emerged as the predominant solution in this field. By leveraging neural networks and natural language processing techniques, these methods can automatically learn complex features, making them better suited for handling large-scale, high-dimensional temporal log data.

* Corresponding author

Neural network-based methods for log anomaly detection are generally categorized into supervised and semi-supervised approaches. Supervised methods, such as DeepSyslog [7], NeuralLog [8], and HitAnomaly [9], face limited practical application due to the scarcity of abnormal samples and the high cost of manual labeling. Consequently, semi-supervised methods [3] trained exclusively on normal logs have gained traction for their superior feasibility in real-world deployment. Existing frameworks typically employ log parsers to transform unstructured raw logs into structured event templates. These semi-supervised approaches primarily adopt two distinct strategies: sequence-based modeling, exemplified by DeepLog [10] and LogAnomaly [11], which leverages LSTMs to predict the next expected log event based on historical sequences; and masked modeling based on Transformer encoders, such as LogBERT [3], LogAnoBERT [12], and LogFit [2], which leverage masking mechanisms to learn deep sequential representations and capture long-range dependencies. Despite these advancements, current neural network-based log anomaly detection methods still exhibit certain limitations.

Current methods, primarily leveraging LSTMs or Transformers, typically formalize log anomaly detection as category prediction, thereby restricting the prediction space to training templates. Given that 20%–45% of log statements evolve throughout their lifecycle [13], [14], Out-of-Vocabulary (OOV) templates frequently trigger false positives during . While LogAnomaly attempts to mitigate this by mapping all OOV templates to their nearest known neighbors, this strategy forcibly associates entries even when semantic discrepancies are significant. Furthermore, reliance on parsers like Drain [1] or Spell [15] causes crucial information loss and makes performance contingent on parser compatibility, often necessitating manual refinement by experts. Consequently, parser-free approaches (e.g., LAnoBERT [12], NeuralLog) leverage regex-based preprocessing to preserve raw semantics.While these methods better retain log context, the highly domain-specific nature of log data often results in strong textual similarity between normal and abnormal entries (e.g., an average similarity

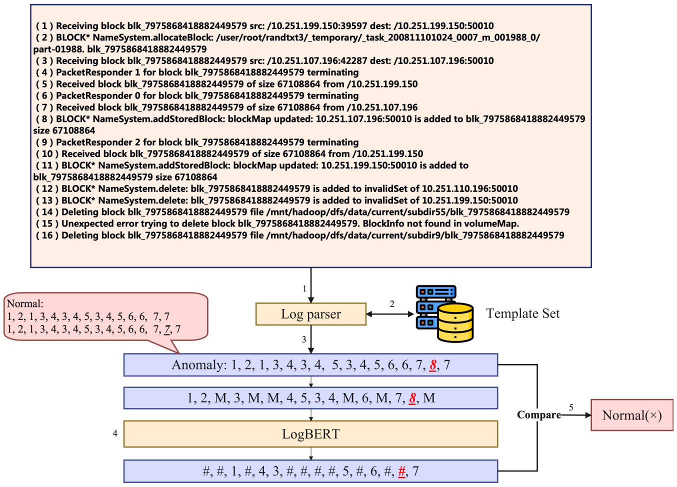

Fig. 1: An Example of Masked Position Dependency.

as high as 0.88 in the BGL dataset). Consequently, threshold-based similarity mapping struggles to establish an effective decision boundary, thereby compromising downstream detection performance.

In addition, mask-based models like LogBERT [3] and LogFit [2] often suffer from "mask position dependency", where the model overemphasizes masked tokens while overlooking anomalies in unmasked positions. Figure 1 illustrates this with an HDFS anomalous sequence: although log (15) is the root cause (an erroneous deletion entry), it may remain unmasked during random sampling. If the remaining unmasked logs align with normal patterns learned during training, the model will likely misclassify the entire sequence as normal. To mitigate this, LAnoBERT [12] adopts a per-position masking strategy during inference to evaluate every log independently. However, this approach incurs prohibitive computational and memory overhead on datasets with lengthy or variable sequences, limiting its practical deployment.

To address the limitations of existing methods, this paper proposes LogRD, a log anomaly detection framework within the semi-supervised mask prediction paradigm. LogRD does not rely on traditional log parsers. It preprocesses raw logs solely using regular expressions, transforming them into templates that preserve more semantic information. These log templates are then mapped to semantic embeddings. whitening post-processing enhances the discriminative power and similarity reliability of these embeddings. Based on semantic representations, LogRD further designs a retrieval-based mapping mechanism that replaces unseen logs only when semantic similarity exceeds a threshold, thereby boosting model adaptability in complex logging environments.

Inspired by RoBERTa's approach [16], LogRD addresses "mask position dependency" and dataset-specific masking ratio tuning by introducing a heuristic masking strategy specifically designed for log anomaly detection. During training, it employs dynamic masking with a fixed rate of 0.5 to expose more masking patterns. At detection, an even-odd masking strategy ensures that all logs in a sequence are masked and predicted, eliminating bias from specific mask positions.

The main contributions of this study are as follows.

1) We introduce a retrieval-Based Mapping Module to enhance robustness. This module allows the model to adapt to variations in log data and ensures its continued usability and practical relevance.
2) We propose a heuristic strategy combining dynamic and odd-even masking to effectively address challenges such as excessive dependence on hyperparameter tuning for masking rates and "mask position dependency."
3) We show that LogRD achieves state-of-the-art performance on three public datasets, with experimental results validating its effectiveness. To facilitate reproducibility, our code are available online[1].

II. Related Work

Log anomaly detection has transitioned from traditional rule-based [17] and statistical methods [18] to advanced deep learning frameworks, with an increasing focus on semi-supervised and parser-free paradigms to enhance system reliability.

A. *Supervised vs. Unsupervised Method*

Supervised techniques, such as LogRobust [14], HitAnomaly [9], and LogSy [19], achieve high accuracy but hinge on extensive labeled datasets. Their scalability is often hampered by the high cost and impracticality of log annotation in real-world deployments.

Conversely, semi-supervised methods such as DeepLog [10], LogBERT [3], and LogFit [2] rely on normal data to detect anomalies by modeling deviations from learned patterns. Although self-supervised tasks (e.g., masked prediction in LogBERT) improve performance, they often suffer from mask position dependency and limited semantic modeling, reducing robustness to log variations.

B. *Parsing-based vs. Parsing-free Methods*

Parser-based methods (e.g., Drain, SemParser, Spell) extract log templates or keys to represent events, serving as the foundation for various supervised and semi-supervised models [10], [11], [20], [21]. While advanced variants [3], [14] integrate attention mechanisms or BERT [22] to enhance context awareness, they remain susceptible to parser limitations, struggling with unseen templates and system evolutions while incurring high operational overhead. In contrast, parser-free methods [2], [8], [12] analyze raw sequences directly to preserve complete semantic information and eliminate manual maintenance. By incorporating deep semantic representations, these approaches (e.g., LogFit [2]) offer superior adaptability and robustness against dynamic log formats.

[1] https://github.com/cakeliang-cc/RDLog

C. Semi-supervised Log Anomaly Detection Architectures

Semi-supervised log anomaly detection has evolved along four main directions. Early LSTM-based methods, such as DeepLog [10] and LogAnomaly [11], model log sequences to predict the next event but struggle with long-range dependencies. Clustering-based methods, e.g., PLELog [23] and OC4Seq [24], detect anomalies via distance to cluster centroids, yet high semantic diversity limits stability, as shown by LogFit [2]. Retrieval-augmented generation (RAG) approaches, including LogRAG [20], RAGLog [25], and XRAGLog [26], leverage external retrieval and LLMs for anomaly reasoning but face high computational costs and elevated false positives under high concurrency. Transformer-based masked modeling methods, such as LogBERT and LAnoBERT, capture long-range dependencies and reduce local bias but remain challenged by OOV tokens and mask-related decision issues.

D. Embedding Methods for Log Representation

Embedding methods map raw log templates into continuous vectors. While static methods such as Word2Vec [27] and GloVe [28] lack contextual awareness, models like BERT [22], RoBERTa [16], and Longformer [29] capture complex semantics. To address BERT's inherent anisotropy, BERT-Whitening [30] centralizes and decorrelates embeddings to ensure a uniform distribution. Together with contrastive learning like SimCSE [31], these techniques enable LogRD to leverage pre-trained models with Whitening for robust log sequence modeling.

III. Methodology

This section introduces the proposed framework, LogRD, as shown in Figure 2. In the figure, the blue line represents the training path, while the red line indicates the detection (testing) path. During training, log sequences pass through **Data Preprocessing**, **PCA-Based Whitening**, **Input Representation**, and **Masking Module (A)** to generate masked sequences for **model training**. During testing, logs are processed sequentially through **Data Preprocessing**, **Input Representation**, **Retrieval-Based Mapping**, and **Masking Module (B)** before being fed to the **Anomaly Detection Model**.

A. Pre-model Data Processing

1) Data Preprocessing: Unlike log parsing–based approaches such as DeepLog and LogBERT, our method avoids using complex log parsers and instead applies a lightweight tokenization and normalization process to raw log entries. Log messages are split using common delimiters (e.g., spaces, colons, and commas), converted to lowercase, and normalized by mapping variable components—such as timestamps, numerical values, and IP addresses—to semantic placeholders via regular expressions, thereby reducing instance-specific variability while preserving semantic information. For example, the raw log entry "1117838601 2005.06.03 R02-M1-N0-C:J12-U11 2005-06-03-15.43.21.523068 R02-M1-N0-C:J12-U11 RAS KERNEL INFO instruction cache parity error corrected." is transformed into the template "num date ras kernel info instruction cache parity error corrected." This representation is referred to as a template, but is obtained without complex rule engineering or dataset-specific heuristics.

2) Whitening with LLM-Augmented Samples: Given the highly templated nature of log data, their semantic representations often aggregate into local clusters within the embedding space, thereby compromising the discriminative efficacy of similarity-based mapping. To enhance discriminative resolution, we employ whitening for decorrelation; this process standardizes the feature distribution and eliminates redundant linear correlations between dimensions by transforming the covariance matrix into an identity matrix. However, despite the significant diversity of log sequences arising from concurrency and parallelism, the underlying template set $T = \{t_i\}_{i=1}^{n}$ remains numerically constrained. This small scale of T hinders the effectiveness of whitening , as stable parameter estimation ideally requires the sample size n to exceed the embedding dimensionality d [30]. To address this, we leverage large language models to generate synthetic samples for T, thereby satisfying the requirements for whitening even under limited log templates.

Specifically, to address the numerical instability occurring when $n < d$, we propose an adaptive strategy as summarized in Algorithm 1. When this rank-deficiency is detected, the framework triggers LLM-based semantic augmentation using the system prompt illustrated in Fig. 3. This process generates lexically diverse yet semantically consistent variants, ensuring the template set effectively covers the embedding space. Subsequently, the whitening transformation is applied to the projected SimCSE embeddings to alleviate anisotropy. The resulting whitening statistics are then reused during detection to maintain distributional consistency and facilitate stable mapping for unseen logs.

3) Input Representation: Given an unstructured log sequence $\text{Seq}^j = \{\text{Log}_1^j, \text{Log}_2^j, \ldots, \text{Log}_k^j\}$, we first preprocess each log entry to extract its corresponding log template t_i^j, as defined in Eq. (1). Each template is then mapped to a semantic vector x_i^j via the embedding model f, as defined in Eq. (2). To further normalize the semantic representations, we apply a whitening transformation using the precomputed matrix W and mean vector μ, as described in Eq. (3).

$$t_i^j = \text{Preprocessing}(\text{Log}_i^j) \tag{1}$$

$$x_i^j = f(t_i^j) \tag{2}$$

$$e_i^j = (x_i^j - \mu)W \tag{3}$$

Where W and μ denote the whitening matrix and the mean semantic vector estimated in Section III-A2. The transformed vector e_i^j represents the whitened semantic embedding of x_i^j, and is used as the input semantic representation of the corresponding log entry for subsequent model encoding.Let $E = \{e_1, e_2, \ldots, e_n\}$ be the set of whitened semantic vectors for training log templates, and $Y = \{y_1, y_2, \ldots, y_n\}$ be their corresponding template labels.

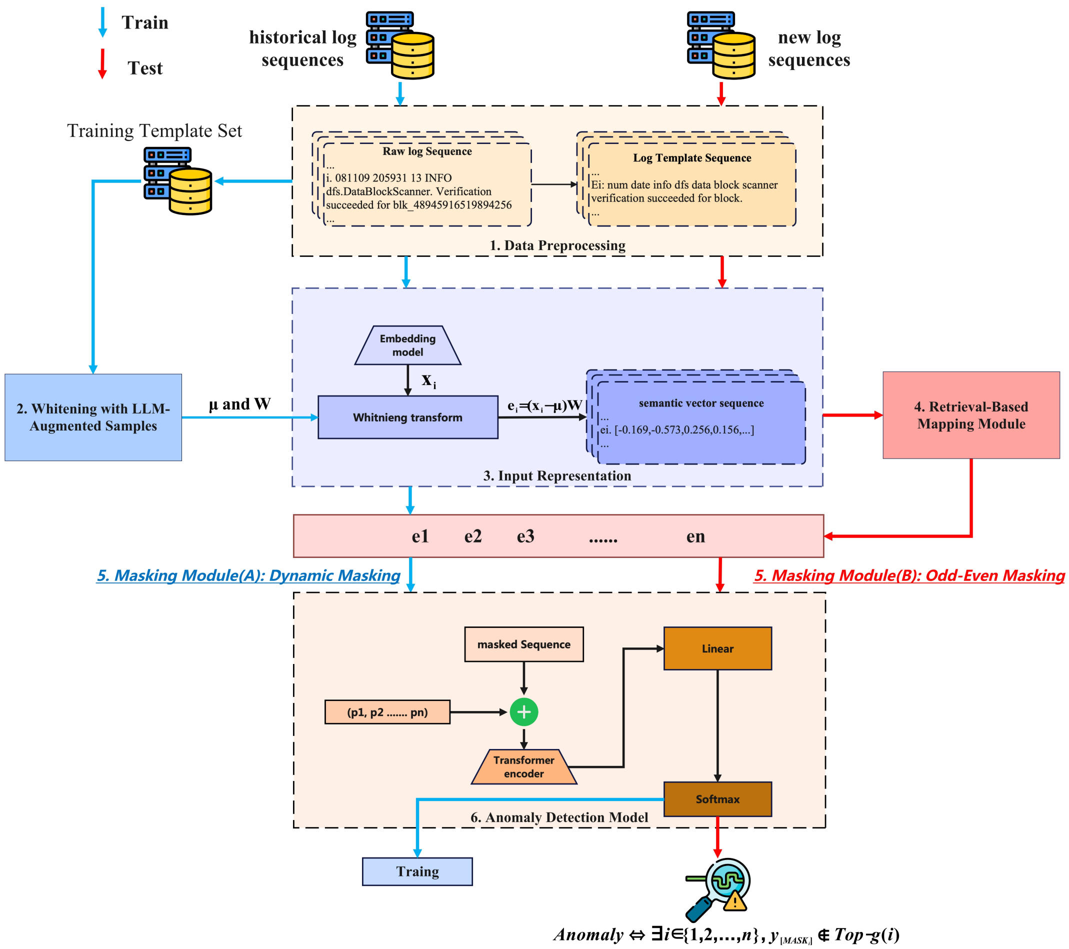

Fig. 2: The overview of LogRD

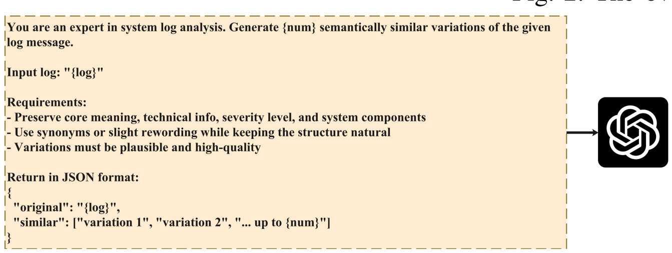

Fig. 3: System Prompt

B. Anomaly detection model building

LogRD employs a Transformer encoder to model log sequences and leverages a dynamic masking strategy to enhance the model's generalization across different masked log formats. The model is trained using a self-supervised objective to capture patterns of normal sequences, which are subsequently used for anomaly detection.

1) Masking Module (A) - Dynamic Masking: After encoding all logs in the sequence Seq^j, we obtain the sequence $S^j = \{e_1^j, e_2^j, \ldots, e_k^j\}$. LogRD standardizes the masking ratio to 0.50 and adopts the dynamic masking strategy, where different positions within the sequence are masked in each training iteration, as illustrated in Fig. 4(a). This strategy enhances the model's ability to focus on logs at varying positions and improves generalization. The sequence after dynamic masking is denoted as S^j_{mask}, which is then input into a Transformer Encoder for learning.

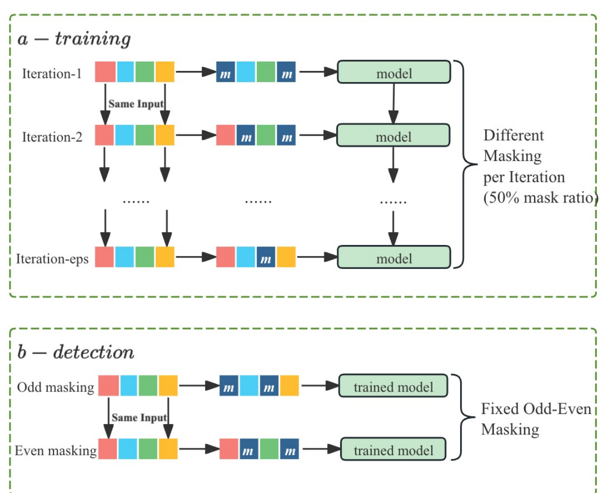

Fig. 4: Masking Module

Algorithm 1 Adaptive Template Augmentation and Whitening

Input:
Training log template set $\mathcal{T} = \{t_i\}_{i=1}^{n}$
Embedding dimension d
Text embedding model $f(\cdot)$
Output:
Whitening matrix W and mean vector μ
LLM Invocation Decision
if $n < d$ **then**
  Compute generation factor $\mathrm{num} = \lceil d/n \rceil$
  Call ChatGPT API with $\mathcal{T}$ and num as inputs
  For each template $t_i \in \mathcal{T}$, ChatGPT generates $\mathrm{num} - 1$ semantically similar templates
  Obtain generated templates: $\mathcal{T}_{\mathrm{gen}} = \{\hat{t}_{i,j}\}_{i=1,j=1}^{n,\mathrm{num}-1}$
else
  Set $\mathrm{num} = 1$
  Set $\mathcal{T}_{\mathrm{gen}} = \emptyset$
end if
Embedding Generation
Construct the augmented template set: $\mathcal{T}' = \mathcal{T} \cup \mathcal{T}_{\mathrm{gen}}$
Let $|\mathcal{T}'| = n \times \mathrm{num}$
for each template $t \in \mathcal{T}'$ **do**
  Compute embedding $x = f(t)$
end for
Form the embedding matrix: $\mathbf{A} = \{x_i\}_{i=1}^{n \times \mathrm{num}}$
Whitening Transformation
Compute mean vector μ, covariance matrix Σ of $\mathbf{A}$
Perform singular value decomposition:
$U, \Lambda, U^T = \mathrm{SVD}(\Sigma)$
Compute the whitening matrix:
$W = \left(U\sqrt{\Lambda^{-1}}\right)^T [:, : 256]$
return W, and μ

2) Objective Function: LogRD employs a Transformer Encoder to capture contextual dependencies within log sequences. For a given masked sequence S^j_{mask}, the encoder computes the hidden representation h^j_t for each position by integrating input embeddings e^j_t and positional embeddings p^j_t:

$$h^j_t = \mathrm{Transformer}(e^j_t + p^j_t) \tag{4}$$

To predict the masked templates, the representation of each masked position is projected onto the template label set Y via a fully connected layer and a softmax function:

$$\hat{y}^j_{[\mathrm{MASK}i]} = \mathrm{Softmax}(w_t \cdot h[\mathrm{MASK}_i]^j + b_t) \tag{5}$$

where w_t and b_t denote trainable parameters, and $\hat{y}^j_{[\mathrm{MASK}_i]}$ represents the predicted probability distribution for the i-th masked token in sequence S^j_{mask}. The model is optimized using a cross-entropy loss function:

$$\mathcal{L} = -\frac{1}{N}\sum_{j=1}^{N}\sum_{i=1}^{M} y^j_{[\mathrm{MASK}i]} \log \hat{y}[\mathrm{MASK}_i]^j \tag{6}$$

where $y^j_{[\mathrm{MASK}_i]}$ is the ground-truth template label, N is the batch size, and M is the number of masked positions. This objective encourages the model to reconstruct masked log templates, providing a robust foundation for anomaly detection.

C. Anomaly Detection

LogRD performs anomaly detection through data preprocessing and input representation, retrieval-based template mapping using semantic similarity, and a global decision module with odd-even masking and Top-g comparison for final anomaly identification.

1) Retrieval-Based Mapping Module: This module enhances model adaptability to evolving log patterns, with its architecture illustrated in Fig. 5. Through input representation, each template in a log sequence is mapped to an embedding e_j. If the template exists in the training label set Y, it proceeds directly to subsequent processing. Otherwise, the module identifies the template embedding $e_k \in E$ with the maximum cosine similarity to e_j. If this similarity exceeds a predefined threshold, e_j and its label y_j are replaced by the most similar counterpart (e_k, y_k). If the threshold is not met, e_j is retained but assigned a specialized label y_{n+1} to designate it as an unseen template. Such instances directly determine the sequence-level classification, labeling the sequence as anomalous to trigger manual inspection and model updates. This mechanism allows the model to approximate unseen patterns using proximal known templates, thereby mitigating the impact of log evolution and bolstering system robustness in dynamic real-world environments.

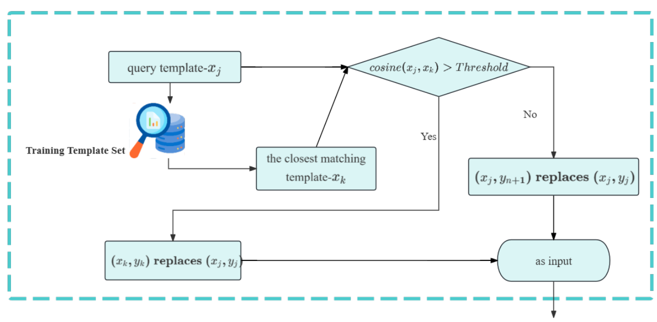

Fig. 5: Retrieval-Based Mapping

2) Masking Module (B) - Fixed Odd-Even Masking: Unlike models like LogBERT and LogFit, which use random masking for anomaly detection, our approach duplicates the log sequence and applies masks to odd and even positions, as illustrated in Fig. 4(b). Each masked sequence maintains a 50% mask ratio, matching the training mask rate. The odd-even masking is motivated by dynamic masking during training, which provides strong generalization ability and enables the model to learn diverse masking patterns. After generating the two masked sequences, we input them separately into the model—predicting odd-position logs from the odd-masked sequence and even-position logs from the even-masked one. This ensures all positions are evaluated.

3) Final Decision: For each masked position, the model computes a probability distribution via Equation (5), reflecting the likelihood of log templates at that position. Following DeepLog and LogBERT, we construct a candidate set Top-$g(i)$ from the top-g highest-probability normal log templates in $\hat{y}_{[\mathrm{MASK}_i]}$. A log entry is classified as normal if its ground-truth template $y_{[\mathrm{MASK}_i]}$ is within this set; otherwise, it is flagged as anomalous. Since LOGRD is trained on normal data, anomalous sequences have low prediction probabilities. If the observed log template is not in the Top-g set, it is classified as anomalous. A log sequence is anomalous if any entry is flagged as such, as formalized in Eq. (7).

$$\text{Anomaly} \Leftrightarrow \exists i \in \{1, 2, ..., n\}, y_{[\mathrm{MASK}_i]} \notin \text{Top-}g(i) \tag{7}$$

IV. EXPERIMENT

In this section, we evaluate the proposed LogRD framework using multiple sets of experiments on Three datasets to answer the following research questions:

RQ1: Does our proposed framework improve performance compared to existing anomaly detection models under the same experimental setup?

RQ2: Does LogRD's masking strategy effectively solve the problem of "mask position dependency"?

RQ3: Does our proposed framework demonstrate stability and adaptability to dynamic changes in log data?

A. Experimental Setup

1) Datasets: We evaluate the proposed LogRD on three log datasets from LogHub[2], including HDFS, BGL, and Spirit. The detailed statistics of three datasets are summarized in Table I.

TABLE I: Dataset Statistics

Dataset	Log Messages	Anomalies	average length
HDFS	11,172,157	284,818	19
BGL	4,747,963	348,460	562
Spirit	7,983,345	768,142	354

2) Baselines: To evaluate the performance of LogRD, we compare representative methods from four categories as baselines, including **LSTM-based approaches** (DeepLog and LogAnomaly), **clustering-based approaches** (PLELog), **LLM-based anomaly detection methods** (LogRAG, RAGLog, and XRAGLog), and **Transformer encoder-based methods** (LogBERT).

3) Implementation Details: Implemented with PyTorch and HuggingFace, LogRD utilizes SimCSE for log template embedding initialization. Sequences are structured by block IDs for HDFS and via a five-minute sliding window for BGL and Spirit. To prevent data leakage, sequences are split chronologically: normal data into training, validation, and test sets (6:1:3), while anomalous data are divided between validation and testing (1:9).

[2]https://github.com/logpai/loghub

4) Evaluation Metrics: We evaluate LOGRD using precision, recall, and F1-score. Precision is defined as $\frac{TP}{TP+FP}$, where TP is the number of correctly detected abnormal logs, and FP is the number of normal logs misclassified as abnormal. Recall is $\frac{TP}{TP+FN}$, where FN refers to abnormal logs misclassified as normal. The F1-score, the harmonic mean of precision and recall, is $2 \times \frac{Precision \times Recall}{Precision + Recall}$.

B. Experimental Results

1) Evaluation of Overall Performance (RQ1): Table II summarizes the performance of LogRD against four baseline categories. LogRD consistently achieves the best or competitive F1-scores across all datasets, demonstrating strong anomaly detection capability. On HDFS, LogBERT achieves perfect Precision but suffers from low Recall (52.38%) due to mask position dependency, resulting in many false negatives, whereas LogRD reaches an F1-score of 97.13%. On the more challenging BGL and Spirit datasets with significant OOV issues, traditional key-based and clustering-based methods show clear precision-recall imbalance or limited clustering quality, while RAG-based methods suffer from retrieval noise. In contrast, LogRD leverages Whitening and a Mapping module to generate more robust semantic representations, achieving the highest F1-scores of 91.76% and 99.37%, respectively.

2) Ablation Study of Masking Strategies (RQ2): This section presents ablation experiments to assess how LogRD's dynamic masking during training and the fixed odd-even masking during testing address the "Mask Position Dependency" issue.

First, we compared static masking (SM) and dynamic masking (DM) strategies during model training, as shown in Table III.Except for Spirit, the detection performance on the HDFS and BGL datasets declined to varying degrees. Specifically, on the HDFS dataset, the F1 score dropped to 91.53%, and the Precision decreased to 85.99%.These results indicate that static masking, due to its limited masking patterns, tends to cause the model to overfit specific masked positions, thereby restricting its generalization ability to other masked positions and making it difficult to apply to the fixed odd-even masking (FM) strategy used during our testing. In contrast, dynamic masking introduces diverse masking patterns, enabling the model to learn more robust contextual representations and significantly improving its generalization capability.

Next, we replaced the fixed odd-even masking (FM) strategy used during testing with a random masking (RM) strategy aligned with the training phase. Specifically, instead of duplicating the input sequence, we applied a 50% masking rate to a single sequence for anomaly detection, following designs such as LogBERT. This modification leads to consistent performance degradation across datasets, especially in recall. For example, on HDFS, recall drops from 96.85% to 60.75%. The main reason is that many anomalous sequences are misclassified as normal, indicating a mask position dependency issue, where the model fails to capture anomalies at non-masked positions.

To fairly evaluate the efficiency and performance of LogRD against LAnoBERT, we construct a controlled variant setting.

TABLE II: Performance comparison on HDFS, BGL, and Spirit datasets (%)

Algorithm	HDFS			BGL			Spirit		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
DeepLog	94.52	90.69	92.57	56.38	73.84	63.94	94.75	99.90	97.26
LogAnomaly	86.03	89.79	87.87	31.30	79.82	44.97	94.76	**99.91**	97.27
PLELog	94.62	96.37	95.49	70.20	79.15	74.41	93.14	76.68	84.11
RAGLog	74.36	76.32	75.32	64.10	92.59	75.76	98.75	96.82	97.69
LogRAG	83.17	80.81	81.97	80.29	82.69	81.47	96.16	96.62	96.36
XRAGLog	92.31	76.59	83.72	84.08	76.32	80.01	98.41	97.99	98.20
LogBERT	**100**	52.38	68.75	39.66	96.08	56.14	93.83	99.88	96.76
Ours	97.41	**96.85**	**97.13**	**87.04**	**97.03**	**91.76**	**99.06**	99.68	**99.37**

TABLE III: Ablation study of different masking strategies (%) and efficiency comparison on HDFS, BGL, and Spirit datasets.

Method	HDFS			BGL			Spirit		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
w/o DM	85.99	97.72	91.53	83.40	96.17	89.33	99.06	99.68	99.37
w/o FM	98.40	60.75	75.10	86.94	92.97	89.85	99.92	97.70	98.80
Both	97.41	96.85	97.13	87.04	97.03	91.76	99.06	99.68	99.37
Efficiency Comparison (Train / Test / F1)									
Method	Train(s)	Test(s)	F1	Train(s)	Test(s)	F1	Train(s)	Test(s)	F1
w/ LAno	2719	2970	96.75	10501	2368	91.03	3160	6297	99.37
LogRD	2884	377	97.13	9955	102	91.76	3603	231	99.37

This is necessary because LAnoBERT adopts a reconstruction-error-based detection paradigm, whereas LogRD relies on a candidate-set-based mechanism. To isolate the effect of masking strategies, we modify LogRD into a LAnoBERT-style variant (w/ LAno) by reducing the training mask rate from 0.5 to 0.15 and replacing the test-stage odd–even masking with LAnoBERT's token-wise exhaustive masking via sequence duplication. Results show that training cost remains comparable between the two methods, indicating minimal impact on efficiency. However, inference time differs significantly due to LAnoBERT's repeated sequence replication. On the Spirit dataset, inference time increases from 231s (LogRD) to 6297s (w/ LAno), a 27$\times$ slowdown. Despite this overhead, LAnoBERT does not outperform LogRD in F1-score on any dataset, demonstrating that LogRD achieves a better efficiency–performance trade-off under a consistent masking design.

3) Performance on Robust Detection (RQ3): The BGL and Spirit datasets exhibit dynamic log patterns, with unseen templates in normal test sequences reaching 51.24% and 17.32%, respectively. To evaluate the contribution of the Whitening and Mapping modules in addressing OOV challenges, we conducted ablation experiments (Table IV).

For the BGL dataset, removing the Whitening module leads to a 16.27% decrease in F1-score (from 91.76% to 75.49%). On the Spirit dataset, the F1-score also shows a clear decline, dropping from 99.17% to 96.72% after removing Whitening. This decline indicates that removing Whitening blurs the similarity distribution, leading to a less discriminative boundary between normal and abnormal templates, and thus degrades the model's ability to distinguish them effectively.

The effect of Whitening can be measured by the mean similarity gap ($diff$), which reflects the difference between the average Top-1 similarity of normal and abnormal OOV templates to the training set (Table V). Whitening substantially widens this gap: $diff$ increases from 0.054 to 0.320 on BGL and from -0.020 to 0.126 on Spirit, confirming that Whitening establishes a more discriminative decision boundary for precise template mapping. A similar trend is also observed with BERT-based encoders. This improvement is further explained by semantic similarity analysis: two fundamentally different templates in BGL—"ras app fatal ciod error creating node map from file cannot allocate memory" and "ras app fatal ciod error creating node map from file block device required"—have a high initial similarity of 0.94, which drops to 0.37 after Whitening, enforcing a sharper de-correlation. In contrast, for a normal OOV case, "ras kernel info ddr suppressing further ce interrupts" and its training template "ras kernel info suppressing further interrupts of same type" remain semantically related, with similarity decreasing more moderately from 0.92 to 0.51. These examples indicate that Whitening responds asymmetrically to semantic relationships: it removes redundant components that may distort similarity estimation, while preserving the dominant semantic components, thereby maintaining meaningful associations.

TABLE IV: LogRD Ablation-Whitening & Mapping

	BGL			Spirit		
	Precision	Recall	F1-score	Precision	Recall	F1-score
w/o Whitening	71.10	80.46	75.49	93.87	99.75	96.72
w/o Mapping	51.00	97.75	67.03	92.76	100.00	96.25
Both	87.04	97.03	91.76	99.06	99.68	99.37

Second, we completely removed the Mapping module. As shown in Table IV, the F1 score on the BGL dataset decreases

TABLE V: Mean Top-1 similarity between OOV test templates and training sets before and after Whitening ($diff$ indicates the margin).

	SimCSE		SimCSE-Whitening	
	score	diff	score	diff
BGL_{normal}	0.932	0.054	0.580	0.320
$\text{BGL}_{abnormal}$	0.878	–	0.260	–
Spirit_{normal}	0.953	-0.020	0.822	0.126
$\text{Spirit}_{abnormal}$	0.973	–	0.696	–
	BERT		**BERT-Whitening**	
	score	diff	score	diff
BGL_{normal}	0.963	0.018	0.399	0.259
$\text{BGL}_{abnormal}$	0.945	–	0.140	–
Spirit_{normal}	0.963	-0.009	0.799	0.117
$\text{Spirit}_{abnormal}$	0.972	–	0.682	–

to 67.03%, while on the Spirit dataset it drops to 96.25%. These results indicate that the Mapping module is crucial for maintaining the adaptability and robustness of the detection framework in dynamic log environments.

V. Conclusions

This paper addresses the challenges in log anomaly detection and proposes LogRD, which introduces a masking strategy to mitigate the *"mask position dependency"* issue common in masked models. By incorporating sentence whitening techniques, LogRD enables more effective measurement of SimCSE semantic vector similarity. Additionally, we propose the *"retrieval-Based Mapping Module"* to effectively handle the variability of log data. Extensive experiments on Three datasets demonstrate LogRD's effectiveness.

References

[1] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: An online log parsing approach with fixed depth tree," in *2017 IEEE international conference on web services (ICWS)*. IEEE, 2017, pp. 33–40.

[2] C. Almodovar, F. Sabrina, S. Karimi, and S. Azad, "Logfit: Log anomaly detection using fine-tuned language models," *IEEE Transactions on Network and Service Management*, vol. 21, no. 2, pp. 1715–1723, 2024.

[3] H. Guo, S. Yuan, and X. Wu, "Logbert: Log anomaly detection via bert," in *2021 international joint conference on neural networks (IJCNN)*. IEEE, 2021, pp. 1–8.

[4] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[5] H. Hotelling, "Analysis of a complex of statistical variables into principal components." *Journal of educational psychology*, vol. 24, no. 6, p. 417, 1933.

[6] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 eighth ieee international conference on data mining*. IEEE, 2008, pp. 413–422.

[7] J. Zhou, Y. Qian, Q. Zou, P. Liu, and J. Xiang, "Deepsyslog: Deep anomaly detection on syslog using sentence embedding and metadata," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3051–3061, 2022.

[8] V.-H. Le and H. Zhang, "Log-based anomaly detection without log parsing," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2021, pp. 492–504.

[9] S. Huang, Y. Liu, C. Fung, R. He, Y. Zhao, H. Yang, and Z. Luan, "Hitanomaly: Hierarchical transformers for anomaly detection in system log," *IEEE transactions on network and service management*, vol. 17, no. 4, pp. 2064–2076, 2020.

[10] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 1285–1298.

[11] W. Meng, Y. Liu, Y. Zhu, S. Zhang, D. Pei, Y. Liu, Y. Chen, R. Zhang, S. Tao, P. Sun *et al.*, "Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs." in *IJCAI*, vol. 19, no. 7, 2019, pp. 4739–4745.

[12] Y. Lee, J. Kim, and P. Kang, "Lanobert: System log anomaly detection based on bert masked language model," *Applied Soft Computing*, vol. 146, p. 110689, 2023.

[13] S. Kabinna, C.-P. Bezemer, W. Shang, M. D. Syer, and A. E. Hassan, "Examining the stability of logging statements," *Empirical Software Engineering*, vol. 23, pp. 290–333, 2018.

[14] X. Zhang, Y. Xu, Q. Lin, B. Qiao, H. Zhang, Y. Dang, C. Xie, X. Yang, Q. Cheng, Z. Li *et al.*, "Robust log-based anomaly detection on unstable log data," in *Proceedings of the 2019 27th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering*, 2019, pp. 807–817.

[15] M. Du and F. Li, "Spell: Online streaming parsing of large unstructured system logs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 11, pp. 2213–2227, 2018.

[16] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[17] M. Cinque, D. Cotroneo, and A. Pecchia, "Event logs for the analysis of software failures: A rule-based approach," *IEEE Transactions on Software Engineering*, vol. 39, no. 6, pp. 806–821, 2012.

[18] A. Oprea, Z. Li, T.-F. Yen, S. H. Chin, and S. Alrwais, "Detection of early-stage enterprise infection by mining large-scale log data," in *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 2015, pp. 45–56.

[19] S. Nedelkoski, J. Bogatinovski, A. Acker, J. Cardoso, and O. Kao, "Self-attentive classification-based anomaly detection in unstructured logs," in *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2020, pp. 1196–1201.

[20] W. Zhang, Q. Zhang, E. Yu, Y. Ren, Y. Meng, M. Qiu, and J. Wang, "Lograg: Semi-supervised log-based anomaly detection with retrieval-augmented generation," in *2024 IEEE International Conference on Web Services (ICWS)*. IEEE, 2024, pp. 1100–1102.

[21] W. Niu, X. Liao, S. Huang, Y. Li, X. Zhang, and B. Li, "A robust wide & deep learning framework for log-based anomaly detection," *Applied Soft Computing*, vol. 153, p. 111314, 2024.

[22] M. V. Koroteev, "Bert: a review of applications in natural language processing and understanding," *arXiv preprint arXiv:2103.11943*, 2021.

[23] L. Yang, J. Chen, Z. Wang, W. Wang, J. Jiang, X. Dong, and W. Zhang, "Semi-supervised log-based anomaly detection via probabilistic label estimation," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 1448–1460.

[24] Z. Wang, Z. Chen, J. Ni, H. Liu, H. Chen, and J. Tang, "Multi-scale one-class recurrent neural networks for discrete event sequence anomaly detection," in *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 2021, pp. 3726–3734.

[25] J. Pan, W. S. Liang, and Y. Yidi, "Raglog: Log anomaly detection using retrieval augmented generation," in *2024 IEEE World Forum on Public Safety Technology (WFPST)*. IEEE, 2024, pp. 169–174.

[26] L. Zhang, T. Jia, M. Jia, Y. Wu, H. Liu, and Y. Li, "Xraglog: A resource-efficient and context-aware log-based anomaly detection method using retrieval-augmented generation," in *AAAI 2025 Workshop on Preventing and Detecting LLM Misinformation (PDLM)*, 2025.

[27] K. W. Church, "Word2vec," *Natural Language Engineering*, vol. 23, no. 1, pp. 155–162, 2017.

[28] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[29] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," *arXiv preprint arXiv:2004.05150*, 2020.

[30] W. Yin and L. Shang, "Efficient nearest neighbor emotion classification with bert-whitening," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 2022, pp. 4738–4745.

[31] T. Gao, X. Yao, and D. Chen, "Simcse: Simple contrastive learning of sentence embeddings," *arXiv preprint arXiv:2104.08821*, 2021.