Egypt-Japan University of Science and Technology

A Comparative Survey on LogBERT and VoBERT

Advances in Log Anomaly Detection

Mohamed khaled yahya 120220081
Omar Khalifa Abdelmonem 120220201

June 8, 2025

Contents

Abstract

New advances in log anomaly detection have increasingly depended on language modeling techniques, with Transformer models like BERT proving to be superior to others. This paper presents a comprehensive comparison of LogBERT and its successor VoBERT, both self-supervised models for detecting anomalies in structured log streams without available labeled anomaly training data. LogBERT introduced a BERT-form encoder with masked log key prediction and hypersphere minimization that achieved impressive performance on benchmark datasets such as HDFS, BGL, and Thunderbird. LogBERT experienced performance drop on log instability, where unseen log templates are encountered at test time.

VoBERT addresses this limitation by suggesting a vocabulary-free masked language modeling task so that the model can deal with dynamically changing log formats stably while still maintaining competitive performance. In high log variability environments, VoBERT outperforms LogBERT on performance metrics like MCC and element-level F1, which reflects its stability in deployment in real-world applications. This study also contrasts these models with other previous and alternative models like DeepLog, LogAnomaly, LogRobust, and BERT-Log, describing their architectures, performance on standard datasets, and trade-offs in supervised and unsupervised scenarios. Benchmarking outcomes, use-case versatility, and open-world detection capabilities are explored to serve as guiderails for future research and practical use in evolving systems.

1 Introduction

Conventional log anomaly detection techniques have depended on rule-based methods manually or statistical methods like PCA, clustering, and invariant mining. These methods are susceptible to brittleness, domain-specific, and neglect the temporal and semantic relationships in log streams. The advent of deep learning, and more recently NLP methods, has tipped the balance of log anomaly detection in favor of models that automatically learn to identify patterns from huge log corpora and generalize nicely across systems with little human effort.

Initial deep learning-based approaches like DeepLog [3] and LogAnomaly [4] employed LSTM networks for log sequence modeling and anomaly detection in event sequence.

DeepLog represented logs as sequences of event IDs and trained an LSTM to forecast the subsequent event in a normal sequence. LogAnomaly took this further by semantically embedding log templates and adding quantitative features such as numerical parameter deviations. Although these models substantially outperformed traditional baselines, they suffered from several problems: they could not capture long-range dependencies, were vulnerable to log format modifications, and needed frequent parsing to fixed template IDs. To transcend these limitations, Guo et al. introduced LogBERT [1], the first to use Bidirectional Encoder Representations from Transformers (BERT) on system logs. LogBERT developed a self-supervised anomaly detection technique relying on masked log key prediction (similar to masked language modeling in NLP) and hypersphere minimization (to concisely represent normal behavior).

Trained solely on normal log sequences, LogBERT achieved state-of-the-art performance on public benchmark datasets such as HDFS, BGL, and Thunderbird, outperforming existing approaches in precision and recall.

However, LogBERT's key limitation is the assumption of a closed log vocabulary. During inference time, it can only deal with log events seen in training; new, unseen log keys (not at all rare in real-world systems due to software updates, configuration modifications, or new modules) are either misclassified or dealt with forcibly as anomalies. Its assumption of a closed vocabulary makes LogBERT extremely susceptible in unstable or evolving logging environments—a vulnerability that was exposed in later testing. To solve this, Daan Hofman proposed VoBERT in 2023 [2]. VoBERT extends LogBERT's Transformer model but substitutes its vocabulary-sensitive pretraining task with a novel Vocabulary-Free Masked Language Modeling (VF-MLM) objective. By doing so, VoBERT is able to gracefully deal with unseen log events without resorting to out-of-vocabulary fallbacks.

VoBERT is as precise as LogBERT on traditional benchmarks but substantially improves robustness if there is log instability at test time, i.e., when new templates arise at test time. VoBERT also allows element-level anomaly localization, enabling finer-grained interpretability for root-cause analysis.

Besides these two, the survey also talks about some other contemporary models such as:

LogRobust [5], a method that leverages semantic log embeddings and an attention-based Bi-LSTM for processing log instability with supervised learning;

BERT-Log [6], which fine-tunes a pre-trained BERT model for log classification in a fully supervised scenario with close to perfect F1 on benchmark datasets;

and open-source platforms like LogHub and LogDeep [9][10], which are testbeds for log anomaly detection algorithm benchmarking.

This essay shall strive to provide:

An in-depth comparison between LogBERT and VoBERT in terms of architecture, training paradigms, and performance across various conditions;

A detailed comparison of their performance under the setting of unstable logging with metrics like F1-score, precision, recall, and Matthews correlation coefficient (MCC);

An explanation of how these models vary from LSTM-based predecessors and transformer-based supervised models; An investigation into trade-offs among robustness, complexity, and generalization, with an eye towards real-world practical usability. The rest of this survey is organized as follows: Section 2 narrates the background and history of development for log anomaly detection models. Section 3 explains the architecture and methodology of LogBERT and VoBERT. Section 4 addresses comparative evaluation using benchmark datasets. Section 5 addresses practical deployment concerns, and Section 6 concludes with remarks and future work directions.

2 Background and Related Work

The processing of automated logs has long been problematic in research, mainly because of the high volume, high velocity, and varied variability that characterize the log data generated by modern IT infrastructures. Before deep learning technologies emerged, anomaly detection in log data was largely handled by conventional methods, which relied on statistical methods, rule-based practices, or unsupervised learning techniques. These baseline methods have highly influenced current practices and continue to exert their influence on the creation of future models.

2.1 Classical Approaches

Traditional approaches to log anomaly detection have typically framed the problem as either a classification task or a clustering-based outlier detection problem. These methods operate under the assumption that the logs are structured and preprocessed, commonly achieved using log parsing techniques such as Drain or Spell, which extract consistent templates or frequency features from raw log messages.

Representative Methods

Invariant Mining This method is based on the premise that certain events in a well-functioning system follow consistent invariant patterns (e.g., "Event A is always followed by Event B"). Any deviation from such patterns is considered anomalous. Algorithms like LogMine have been developed to detect these invariants. However, invariant mining struggles in the presence of noisy or non-deterministic behavior, and its applicability diminishes in dynamic or complex environments.

Principal Component Analysis (PCA) PCA is employed to reduce the dimensionality of event frequency vectors, such as those representing log templates over fixed time windows. Anomalies are identified based on their projection errors relative to a learned low-dimensional subspace. Although effective in certain cases, PCA fails to model temporal correlations and often demands careful calibration of parameters such as window size and anomaly thresholds.

k-Means and DBSCAN Clustering Clustering-based approaches group similar log sequences together. Instances that do not conform to any cluster are marked as anomalies. These methods are unsupervised and easy to implement but exhibit poor performance with sparse, high-dimensional data and are sensitive to variations in log format.

One-Class Support Vector Machines (OCSVMs) OCSVMs are trained solely on normal data to define a decision boundary separating normal from abnormal behaviors. As one of the earliest methods for unsupervised log anomaly detection, OCSVMs are conceptually simple but suffer from sensitivity to noise and do not scale well to large datasets.

Rule-Based Systems Rule-based systems rely on manually crafted rules to identify known error patterns or sequences in logs. While they offer transparency and precise control, these systems require significant manual effort, are prone to fragility under software updates, and pose substantial maintenance overhead.

Limitations of Traditional Methods

Despite their early adoption, classical methods exhibit several critical limitations:

- **Temporal insensitivity:** Many methods treat logs as unordered sets or frequency counts, ignoring the sequential nature of log events.
- **High false positive rates:** Especially in systems with noisy or variable log output.
- **Limited adaptability:** Changes in log formats often require re-parsing, reconfiguration, or model re-training.
- **Dependence on manual feature engineering:** These approaches often rely heavily on handcrafted features, reducing their generalizability across different systems and domains.

These shortcomings have motivated a shift toward data-driven models capable of automatically learning patterns from raw log data. Deep learning-based techniques aim to address these challenges by capturing semantic, contextual, and temporal characteristics of log events without extensive manual intervention. This progression is marked by the emergence of sequence-based models such as DeepLog, setting the stage for more advanced, context-aware approaches. static to sequential and context-dependent log anomaly detection.

2.2 LSTM-based Methods

Conventional log analysis techniques face numerous limitations, prompting the exploration of deep learning approaches—particularly those utilizing Recurrent Neural Networks (RNNs) such as Long Short-Term Memory (LSTM) networks. LSTMs are especially effective for log data analysis due to their ability to capture temporal dependencies and sequential patterns, enabling automated and contextual anomaly detection.

DeepLog (2017)

DeepLog represents one of the pioneering works in applying LSTM networks to the task of log anomaly detection. It treats log sequences as ordered event ID series and frames the problem as one of next-event prediction. During the training phase, the LSTM is trained on sequences of normal logs to predict subsequent events. At inference time, if a log event deviates from the top-k predicted next events, it is flagged as anomalous.

Key Contributions

- Introduced deep learning to log analysis without the need for manual feature engineering.
- Demonstrated strong performance on structured datasets such as HDFS and BGL.

Limitations

- Operates on log template IDs, thus missing semantic information present in raw log messages.
- Susceptible to log format changes and previously unseen templates (closed-vocabulary issue).
- Cannot effectively handle subtle semantic variations or numerical log attributes.

LogAnomaly (2019)

LogAnomaly extends DeepLog by incorporating semantic embeddings and numerical log attributes. It employs a novel embedding approach called *Template2Vec*, which transforms log messages into dense vectors based on their textual content. Furthermore, it integrates quantitative features from logs, allowing it to identify both structural and value-based anomalies.

Key Contributions

- Merges semantic and quantitative information for comprehensive anomaly detection.
- Detects both sequence-based and metric-based anomalies.
- Improves detection accuracy in complex and heterogeneous log environments.

Limitations

- Still depends on prior template extraction, which may introduce parsing errors.
- Sensitive to the quality of log parsing and noise.
- Detection performance is influenced by carefully tuned threshold values.

LogRobust (2019)

LogRobust addresses the limitations of previous models under dynamic or unstable logging environments. It employs a Bi-directional LSTM (Bi-LSTM) architecture augmented with an attention mechanism. Unlike DeepLog, it avoids reliance on predefined templates by leveraging semantic embeddings directly from raw log messages. The attention mechanism further improves interpretability by focusing the model on critical log events.

Key Contributions

- First LSTM-based model to incorporate attention for enhanced accuracy and interpretability.
- Capable of handling unstable log formats and adapting to unseen anomaly patterns.
- Supports incremental learning, making it suitable for evolving real-world systems.

Limitations

- Requires labeled anomaly data, which may be scarce or costly to obtain.
- May generalize poorly to novel anomaly types not seen during training.
- Attention and Bi-LSTM architectures increase computational demands.

These LSTM-based approaches laid the foundation for more advanced transformer-based models such as LogBERT and VoBERT. The evolution from sequence prediction to semantically rich embeddings highlights the growing need for adaptable, interpretable, and robust log anomaly detection systems in open-world scenarios.

2.3 Transformer-based Methods

Transformer architectures have achieved significant success in natural language processing tasks, most notably through BERT (Bidirectional Encoder Representations from Transformers). Their effectiveness lies in the self-attention mechanism, which enables the modeling of long-range dependencies and complex interrelations within sequences. These properties make Transformers particularly well-suited for system log analysis, where traditional machine learning and LSTM-based models often fall short.

System logs can be viewed as a structured language, composed of timestamped entries, templates, and parameterized values. Motivated by this analogy, recent research has adapted self-supervised pretraining and masked prediction strategies—originally developed for NLP—to the task of log anomaly detection. This has given rise to a new generation of Transformer-based models that offer scalability, contextual understanding, and robustness in handling heterogeneous log data.

LogBERT (2021)

LogBERT introduced the first self-supervised framework specifically designed for log anomaly detection using a Transformer architecture. The model is trained using two main objectives: Masked Log Key Prediction (MLKP), which involves predicting randomly masked log keys in sequences, and Volume of Hypersphere Minimization (VHM), which encourages compactness in the latent representation of normal sequences, treating deviations as anomalies.

Advantages

- Enables bidirectional modeling of log sequences, conditioning each log event on both past and future context.
- Demonstrates high performance on benchmark datasets such as HDFS, BGL, and Thunderbird without relying on labeled anomaly data.
- Validates the use of self-supervised learning paradigms in log analysis, similar to their use in NLP.

Challenges

- Depends on a predefined set of log keys, limiting generalization to previously unseen templates.
- Susceptible to errors introduced during log parsing, which may affect downstream performance.

BERT-Log (2022)

BERT-Log leverages pre-trained BERT models from NLP and fine-tunes them on labeled log sequences for supervised anomaly classification. Unlike LogBERT, it processes raw log messages directly, thus bypassing reliance on template extraction. It employs sliding window techniques and node-level grouping to maintain contextual continuity across log sessions.

Advantages

- Achieves superior F1-scores (above 99%) on benchmark datasets under controlled conditions.
- Utilizes transfer learning, requiring minimal labeled data to achieve high performance.
- Improves log session segmentation and classification by considering node identifiers and time-based intervals.

Challenges

- Requires labeled data, making it less suitable for fully unsupervised applications.
- May struggle to generalize to novel anomaly types not seen during training.
- Demands high computational resources for fine-tuning and inference.

VoBERT (2023)

VoBERT addresses the static vocabulary limitation inherent in LogBERT by introducing Vocabulary-Free Masked Language Modeling (VF-MLM). This objective allows the model to operate at the sub-token or character level, enabling inference on log templates that differ from those seen during training. As a result, VoBERT offers enhanced adaptability to dynamic and evolving log formats.

Advantages

- Matches LogBERT's performance in stable scenarios while significantly outperforming it under log instability conditions.
- Eliminates dependence on fixed token sets, making it more robust to template drift and log format changes.
- Demonstrates superior resilience in open-world logging environments.

Challenges

- Involves more complex tokenization and preprocessing strategies.
- Differentiating between harmless and harmful unseen log events remains a nuanced challenge.
- Performance gains in static environments are marginal compared to LogBERT, despite added complexity.

3 Architectures and Methodologies

This section explores the core architectural designs and training strategies behind **LogBERT** and **VoBERT**, two Transformer-based self-supervised models for log anomaly detection. While both share a foundational reliance on BERT-style Transformers, they diverge significantly in how they model log sequences, handle unseen log formats, and structure their objectives for anomaly scoring.

3.1 LogBERT Architecture

LogBERT, introduced by Guo et al. [?], was among the first models to apply Transformer-based self-supervised learning to system logs. It relies on a BERT-like encoder and two core training tasks: **Masked Log Key Prediction (MLKP)** and **Hypersphere Minimization (HMin)**.

Input Structure and Tokenization

- Logs are parsed into log templates (log keys) using a preprocessing tool (e.g., Drain).
- A log session is represented as a sequence of log key tokens:

$$S = [T_1, T_2, \ldots, T_n]$$

  where T_i is a log event.
- Each token is embedded via a lookup table and passed into Transformer layers.

Masked Log Key Prediction (MLKP)

- During training, random tokens are masked, and the model learns to predict the missing tokens, similar to BERT's Masked Language Modeling (MLM).
- The loss function is defined as:

$$\mathcal{L}_{\text{MLKP}} = -\sum_{i \in M} \log P(T_i \mid T_{\setminus M})$$

  where M is the set of masked positions.
- *Purpose:* To learn contextual dependencies in log sequences, i.e., predict what events typically follow others.

Hypersphere Minimization (HMin)

- Normal log sequences are assumed to lie within a compact region of embedding space.
- The model minimizes the radius of a hypersphere enclosing these embeddings.

- The anomaly score is computed as:

$$\text{Score}(x) = \|f(x) - c\|_2^2$$

where $f(x)$ is the embedding of the log sequence and c is the center of the hypersphere.

- *Purpose:* Encourages embeddings of normal sequences to cluster closely, while anomalies fall outside.

Limitations

- **Fixed Vocabulary:** Only known log keys are recognized; unseen templates cause out-of-vocabulary errors.
- **Closed-World Assumption:** Assumes log distributions are static, which is unrealistic in dynamic environments.
- **Coarse Prediction:** Labels the entire sequence as anomalous, without localizing the specific abnormal log.

3.2 VoBERT Architecture

VoBERT, proposed by Hofman [**?**], addresses LogBERT's core limitations by introducing a vocabulary-free modeling strategy, enabling generalization to unseen log formats and supporting fine-grained anomaly localization.

Vocabulary-Free Input Representation

- Tokenizes logs at a finer granularity using:
  - Character-level encoding
  - Sub-token (e.g., Byte-Pair Encoding)
- Removes reliance on predefined log templates.
- *Advantage:* Becomes format-agnostic and robust to changes in logging software/configuration.

Vocabulary-Free Masked Language Modeling (VF-MLM)

- Similar in spirit to MLKP but applied at the sub-token/character level:

$$\mathcal{L}_{\text{VF-MLM}} = -\sum_{i \in M} \log P(c_i \mid C_{\setminus M})$$

where c_i is a character in the log and C is the full sequence.

- *Purpose:* Learns structure and syntax without memorizing specific log keys, enabling flexible pattern recognition.

Hypersphere Minimization (Same Objective)

- Retains the same hypersphere loss as LogBERT:

$$\text{Score}(x) = \|f(x) - c\|_2^2$$

- Embeds sub-token patterns rather than fixed log keys, improving robustness.

Open-Set Recognition Capability

- VoBERT is open-set by design, handling inputs not seen during training.
- Contrasts with LogBERT's closed-world assumption.

Element-Level Anomaly Prediction

- Scores each token's embedding individually for anomaly likelihood.
- Process:
  - Extract embeddings per token
  - Measure distance to hypersphere center
  - Threshold/aggregate distances to identify anomalous tokens
- *Result:* Localizes the specific log responsible for the anomaly.

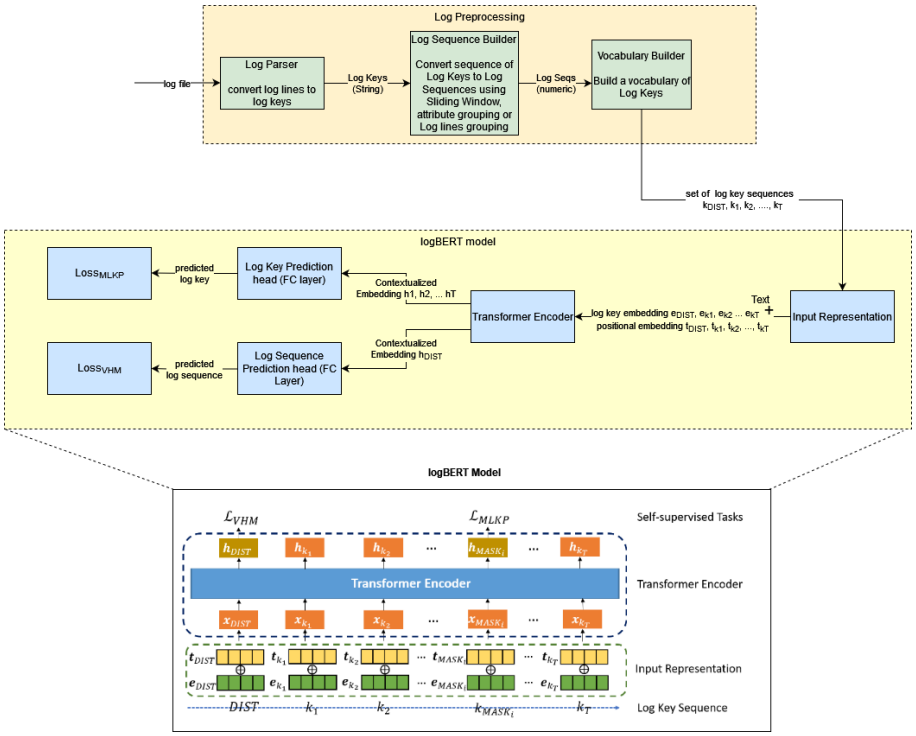
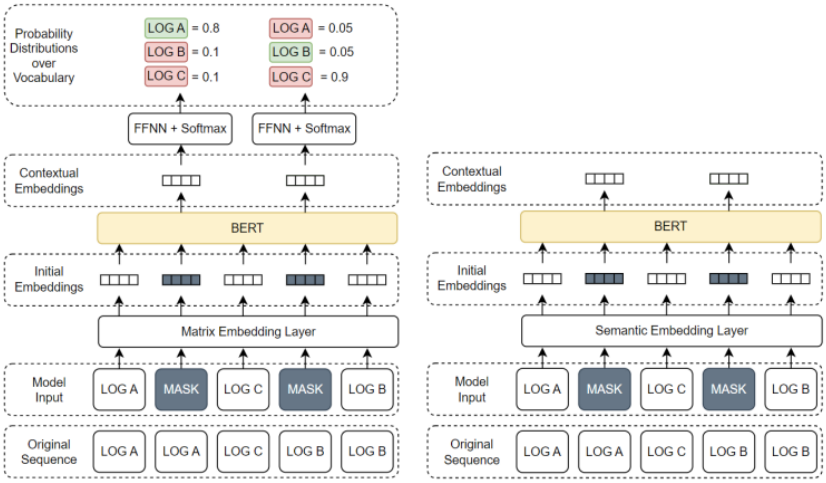


Figure 1: Overview of LogBERT Architecture



(a) Traditional MLM Model Architecture.

(b) Vocabulary-Free Model Architecture. Note that the embedding method changed, and the FFNN + Softmax layer is removed.

Figure 2: Overview of VoBERT Architecture

4 Comparative Evaluation

This section presents a comprehensive empirical evaluation of **LogBERT**, **VoBERT**, and other anomaly detection models on standard public datasets. Emphasis is placed on both the *reliability* of these datasets for real-world system monitoring and the *granularity* of evaluation, comparing element-level and sequence-level detection.

4.1 Datasets Used

The experiments rely on several well-established system log datasets frequently used in anomaly detection literature. These datasets are authentic captures from large-scale systems, ensuring high relevance to real-world applications.

- **HDFS (Hadoop Distributed File System)**:
  - Collected from Amazon's implementation of HDFS over five weeks.
  - Contains over 11 million log lines from normal and abnormal executions.
  - Abnormal sequences are manually labeled based on injected fault scripts.
  - *Strength:* Large-scale, well-labeled, event-sequenced logs with contextual dependencies.
- **BGL (Blue Gene/L Supercomputer Logs)**:
  - Captured from IBM's BlueGene/L system at Lawrence Livermore National Lab.
  - Logs cover various system states including normal, warning, and failure events.
  - Anomalies are annotated based on system administrator reports and fault tags.

  - *Strength:* Real-world supercomputing environment, labeled by domain experts.

- **Thunderbird**:
  - Obtained from a high-performance computing cluster at Los Alamos National Laboratory.
  - Logs represent hardware and software events from a live operational environment.
  - Ground truth labels are generated via error injection and expert tagging.
  - *Strength:* High diversity of error types and realistic log diversity.

- **HPC, Spirit, or other datasets (optional)**:
  - Some studies include additional logs from IoT, HPC, or application server domains.
  - These may lack ground truth or contain semi-supervised labeling protocols.

Dataset Authenticity and Real-World Relevance All listed datasets are collected from live systems under actual operating conditions (not simulated). Logs include timestamps, error codes, and real command traces. The ground truth anomaly labels—particularly for HDFS and BGL—are either:

- Manually annotated by domain experts.
- Generated via controlled fault injection experiments.

This ensures that model evaluations reflect real-world applicability and challenges, including:

- Log noise and redundancy.
- Irregular temporal patterns.
- Evolving software and hardware behaviors.

4.2 Evaluation Metrics

Performance is evaluated using the following standard metrics:

- **Precision**:
$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$
Measures the proportion of predicted anomalies that were truly anomalous.

- **Recall**:
$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$
Indicates how many actual anomalies the model was able to detect.

- **F1-Score**:
$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$
Harmonic mean of precision and recall. A balanced indicator of model performance.

- **Matthews Correlation Coefficient (MCC)**:
$$\text{MCC} = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$
Captures all four confusion matrix terms, offering a robust performance score even with class imbalance.

Sequence-Level vs Element-Level Evaluation

- **Sequence-Level Evaluation**:
  - Evaluates the model's ability to label an entire session as anomalous or normal.
  - Common in models like LogBERT that produce one anomaly score per sequence.
  - *Limitation:* Cannot pinpoint the specific event that caused the anomaly.
- **Element-Level Evaluation**:
  - Evaluates each log event or token individually for anomaly likelihood.
  - Used in VoBERT and other fine-grained models.
  - *Advantage:* Enables root-cause localization, supports real-time anomaly triage.

4.3 Comparison Table

Model	Architecture	Training Type
DeepLog (2017)	LSTM sequence model predicting next log event.	Unsupervised (trained on normal logs only).
LogAnomaly (2019)	LSTM with Template2Vec embeddings; captures semantic and numeric anomalies.	Unsupervised (trained on normal streams).
LogRobust (2019)	BiLSTM with attention; semantic embeddings; adaptive to unstable logs.	Supervised (requires labeled sequences).
BERT-Log (2022)	Pre-trained BERT model fine-tuned for log classification.	Supervised (needs anomaly labels).
LogBERT (2021)	Transformer trained from scratch with MLM and Hypersphere minimization.	Self-supervised (normal logs only).
VoBERT (2023)	BERT with Vocabulary-Free MLM; supports element-level anomaly prediction.	Self-supervised (robust to unseen logs).

Table 1: Comparison of model architectures and training strategies.

Model Performance Across Datasets

Model	HDFS (Prec / Rec / F1 / MCC)	BGL (Prec / Rec / F1 / MCC)	Thunderbird (Prec / Rec / F1 / MCC)
DeepLog (2017)	88.4 / 69.5 / 77.3 / –	89.7 / 82.8 / 86.1 / –	87.3 / 99.6 / 93.1 / –
LogAnomaly (2019)	94.2 / 40.5 / 56.2 / –	73.1 / 76.1 / 74.1 / –	86.7 / 99.6 / 92.7 / –
LogRobust (2019)	98.0 / 100 / 99.0 / –	∼94–95% F1	Not Reported
BERT-Log (2022)	∼99 / 99 / 99.3 / –	∼99 / 99 / 99.4 / –	Not Reported
LogBERT (2021)	87.0 / 78.1 / 82.3 / –	89.4 / 92.3 / 90.8 / –	96.8 / 96.5 / 96.6 / –
VoBERT (2023)	∼82 / 78 / 80 / 0.70	∼90 / 94 / 92 / 0.90	∼97 / 96 / 96 / 0.95

Table 2: Model evaluation across datasets using standard metrics.

5 Deployment Considerations

The transition from classical or LSTM-based models to Transformer-based architectures such as LogBERT and VoBERT introduces significant improvements in anomaly detection accuracy and robustness. However, these gains come with engineering and deployment challenges that must be considered before real-world adoption.

This section outlines key considerations around model complexity, resource usage, log variability, and interpretability, with a special focus on VoBERT's representation density and long input sequences.

5.1 Model Complexity and Scalability

LogBERT:

- Lightweight BERT variant trained from scratch on structured log keys.
- Each log event is represented as a single token (template ID), yielding short sequences (20–50 tokens).
- Training and inference are relatively fast.
- Scales efficiently to large datasets when logs are stable.

VoBERT:

- Uses character-level or sub-token inputs, greatly increasing input length.
- A single log message may expand into 10–50 tokens; sequences of 30 events can exceed 1000 tokens.
- Transformer memory usage grows quadratically ($\mathcal{O}(n^2)$) with sequence length.
- Requires techniques like truncation, smaller batch sizes, or long-sequence models (e.g., Longformer).

5.2 Hardware Requirements

Model	GPU Requirement	Training Time (1 GPU)	Notes
LogBERT	Moderate (8–12GB)	~1–2 hours (HDFS)	Lightweight due to short token s
VoBERT	High (16–24GB+)	~4–6 hours (HDFS)	Needs memory-efficient batching

Table 3: GPU and runtime requirements for LogBERT and VoBERT.

- VoBERT benefits from multi-GPU training setups.
- For edge or embedded deployment, LogBERT is more practical.
- VoBERT may require optimization via model pruning, distillation, or compression.

5.3 Suitability for Unstable Logs

Log instability may result from:

- Log format or template changes.
- Software updates or module reconfiguration.
- Migration across platforms (e.g., cloud to edge).

LogBERT:

- Maps unseen templates to unknown tokens or drops them.
- Leads to false negatives or spurious alerts.

VoBERT:

- Processes raw input without template IDs.
- Remains effective when 80%+ of test logs are unseen.
- Element-level scoring highlights only the affected parts.

Conclusion: VoBERT is better suited for dynamic and evolving logging environments such as microservices.

5.4 Interpretability and Trust

LogBERT:

- Produces one anomaly score per sequence.
- Easy to use, but offers little insight into specific causes.

VoBERT:

- Scores individual log events.
- Enables visualization and root cause identification.
- Supports integration with explainable AI tools.

Both models:

- Are black-box neural networks.
- Require tools like attention heatmaps, PCA, or t-SNE to interpret internal representations.

5.5 Dense Representation Challenges in VoBERT

- Fine-grained tokenization creates long, dense sequences.
- Sensitive to noisy elements like timestamps or hashes.

Key Challenges:

- **Input Normalization:** Must remove irrelevant variability.
- **Padding/Truncation:** Needed to manage long log sequences.
- **Score Aggregation:** Token-wise anomaly scores must be combined using strategies like mean, max, or thresholding.

6 Conclusion

This survey provided a detailed comparison between LogBERT and VoBERT, two prominent self-supervised models used for log anomaly detection. These models have significantly advanced the field by integrating natural language processing techniques into log analysis, offering more accurate and adaptable detection methods compared to traditional statistical and LSTM-based approaches.

LogBERT introduced a novel approach with masked log key prediction and hypersphere minimization, which effectively models normal log behavior in a self-supervised manner. It demonstrated strong performance on benchmark datasets under stable logging conditions. However, its reliance on a fixed vocabulary and inability to handle unseen log templates posed challenges in dynamic, real-world environments.

In contrast, VoBERT addressed these limitations by adopting a vocabulary-free masked language modeling objective. This adaptation allowed VoBERT to generalize to previously unseen log formats, resulting in more robust performance in evolving production systems. Furthermore, VoBERT's capability to generate anomaly scores at the log-event level enhanced interpretability and facilitated root cause analysis, providing a significant advantage over existing models.

Through architectural analysis, empirical comparison, and deployment considerations, this study has underscored the strengths and trade-offs of each approach. While LogBERT remains suitable for stable environments with modest computational resources, VoBERT emerges as a more resilient choice for open-set and volatile scenarios, albeit with higher computational demands.

In conclusion, both LogBERT and VoBERT contribute uniquely to the field of log anomaly detection, reflecting a broader trend towards intelligent and adaptive monitoring systems. Future advancements are anticipated to focus on optimizing efficiency, enhancing interpretability, and extending support for diverse log formats in real-time applications.

References

1. H. Guo, J. Lin, Y. Zhang, and X. Ou, "LogBERT: Log Anomaly Detection via BERT," in *2021 IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, Taipei, Taiwan, 2021, pp. 1–12. doi: 10.1109/DSN48987.2021.00074.

2. D. Hofman, *VoBERT: Unstable Log Sequence Anomaly Detection*, MSc Thesis, Delft University of Technology, Netherlands, 2023. Available: https://repository.tudelft.nl/record/uuid:f8593fb8-aa20-4caa-b12e-2521416c98a1

3. M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning," in *Proc. 2017 ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, Dallas, TX, USA, 2017, pp. 1285–1298. doi: 10.1145/3133956.3134015.

4. W. Meng *et al.*, "LogAnomaly: Unsupervised Detection of Anomalies in Unstructured Logs," in *Proc. IEEE Int. Conf. Distributed Computing Systems (ICDCS)*, 2019, pp. 1–10. doi: 10.1109/ICDCS.2019.00184.

5. H. Zhang, D. Ye, and Y. Zheng, "Robust Log-Based Anomaly Detection on Unstable Log Data," in *Proc. 27th ACM Joint European Software Engineering Conf. and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, Tallinn, Estonia, 2019, pp. 807–817. doi: 10.1145/3338906.3341175.

6. S. Chen and H. Liao, "LogBERT-Log: A Pre-trained BERT Model for Anomaly Detection in System Logs," *Applied Artificial Intelligence*, vol. 36, no. 1, pp. 1–22, 2022. doi: 10.1080/08839514.2021.1992145.

7. H. Guo, "LogBERT: Anomaly Detection in Logs Using BERT," GitHub, 2021. Available: https://github.com/HelenGuohx/logbert

8. D. Hofman, "VoBERT: Vocabulary-Free BERT for Log Anomaly Detection," GitHub, 2023. Available: https://github.com/daanh99/VoBERT

9. LogPai Team, "LogHub: A Large Collection of System Log Datasets," GitHub, 2020. Available: https://github.com/logpai/loghub

10. LogPai Team, "LogDeep Toolkit for Deep Learning-Based Log Anomaly Detection," GitHub, 2020. Available: https://github.com/logpai/logdeep

11. HDFS Logs — from LogHub dataset. Available: https://github.com/logpai/loghub/tree/master/HDFS

12. BGL Logs — from LogHub dataset. Available: https://github.com/logpai/loghub/tree/master/BGL

13. Thunderbird Logs — from LogHub dataset. Available: https://github.com/logpai/loghub/tree/master/Thunderbird

14. Bank Security Logs Dataset — proprietary dataset used in VoBERT thesis. Not publicly available; discussed in thesis.

15. Y. Lou, Y. Li, H. Duan, and H. Yu, "LogDeep: A Deep Learning Based System Log Anomaly Detection Toolkit," *arXiv preprint*, arXiv:2008.11979, 2020. Available: https://arxiv.org/abs/2008.11979

16. H. He *et al.*, "LogHub: A Large Collection of System Log Datasets Towards Automated Log Analytics," *arXiv preprint*, arXiv:2008.06448, 2020. Available: https://arxiv.org/abs/2008.06448

17. A. Mascharka and P. Maj, "Translog: Transformer-Based Log Anomaly Detection," in *Proc. IEEE Big Data*, 2021. Available: https://ieeexplore.ieee.org/document/9671421