Archives available at journals.mriindia.com

International Journal of Advanced Scientific Research and Engineering Trends

ISSN: 2456-0774

Volume 10 Issue 03, 2026

AI OPS Log Anomaly Detector – A Lightweight Unsupervised Framework for Log Analysis, Detection, And Visualization

[1]Rohini Govind Niphade, [2]Dr. Ankita Karale, [3]Dr. Balkrishna K. Patil, [4]Dr. Naresh Thoutam

[1]Student, Computer Department, Sandip Institute of Technology and Research Centre, Nashik, India
[2,3,4] Dr. Computer Department, Sandip Institute of Technology and Research Centre, Nashik, India
Email: [1]rohini96niphade94@gmail.com, [2]ankita.karale@sitrc.org, [3]balkrishnapatileng@gmail.com, [4]naresh.thoutam@sitrc.org

Peer Review Information

Submission: 10 Feb 2026

Revision: 26 Feb 2026

Acceptance: 11 March 2026

Keywords

Log Anomaly Detection, AIOps, Unsupervised Learning, Isolation Forest, ECOD, Local Outlier Factor, Drain3, TF-IDF, Feature Extraction, Log Visualization, Lightweight Systems

Abstract

System logs are a primary source of information for monitoring and maintaining modern software systems. However, the growing volume and complexity of logs make manual inspection inefficient and error-prone. Traditional log analysis tools either depend on predefined rules or require heavy infrastructure, limiting their usability in small-scale or offline environments. This work presents a lightweight and interactive log anomaly detection system that integrates template mining, feature extraction, and unsupervised machine learning within a unified interface. The system allows users to upload log files, process them through structured pipelines, and detect anomalies using models such as Isolation Forest, ECOD, and Local Outlier Factor. Feature extraction techniques include TF-IDF vectorization, optional semantic embeddings, and statistical feature generation, enabling comprehensive representation of log behavior. A key aspect of the system is its user-centric design, where all processing stages—from ingestion to visualization—are accessible through an intuitive interface. Users can perform operations such as parsing logs, extracting features, running anomaly detection, and visualizing results using timeline charts and distribution plots. The system also supports exporting results in multiple formats, ensuring usability for further analysis. Overall, the framework provides a practical and explainable solution for log anomaly detection, emphasizing simplicity, transparency, and offline capability. It is particularly suitable for academic, research, and small enterprise environments where lightweight deployment and interactive analysis are essential.

Introduction

System logs are an essential component of modern computing environments, capturing every significant event occurring within applications, servers, and networks. These logs provide valuable insights for monitoring system health, diagnosing failures, and ensuring reliable operation. However, with the growth of distributed systems and cloud-based applications, the volume and complexity of logs have increased significantly, making manual analysis both time-consuming and ineffective. [2]

Traditional log analysis methods rely on manual inspection or rule-based filtering techniques. While these approaches are simple to

implement, they are limited to identifying known issues and fail to detect new or unexpected anomalies. As systems evolve, maintaining such rules becomes increasingly difficult. More advanced solutions, including commercial AIOps platforms, offer automated detection capabilities but often require expensive infrastructure and continuous internet connectivity. Additionally, many of these tools provide limited transparency, making it difficult for users to understand how anomalies are detected. [9]

This creates a clear need for a system that is both lightweight and capable of automated anomaly detection without relying on predefined rules or heavy computational resources. Such a system should also be easy to use, allowing users to interact with different stages of the analysis process and understand the results clearly. [5]

To address this need, the proposed system introduces an interactive log anomaly detection framework that combines template mining, feature extraction, and unsupervised learning within a single platform. The system enables users to upload log files, process them step by step, and detect anomalies using multiple models. It also provides visualization tools such as timelines and distribution charts, helping users interpret results effectively. [16]

The primary contribution of this work lies in integrating system design, implementation, and visualization into a cohesive solution. The framework emphasizes usability, modularity, and explainability, making it suitable for academic research, educational purposes, and small-scale deployments where traditional AIOps solutions are not practical.

Objectives Of the System

The objectives of the proposed system are defined based on the need for a lightweight, explainable, and user-friendly log anomaly detection framework. These objectives guide both the system design and implementation.

1. To develop a lightweight and stand-alone log analysis system

The system is designed to run on a local machine without requiring cloud infrastructure or high-end resources, making it suitable for academic and small-scale environments.

2. To enable automated log ingestion and normalization

The system should accept multiple log formats and convert them into a structured representation for consistent processing.

3. To implement template-based log mining using Drain3

Log messages are grouped into templates to reduce noise and improve pattern recognition, allowing efficient analysis of repetitive log structures.

4. To perform feature extraction using multiple techniques

The system integrates TF-IDF vectorization, optional semantic embeddings, and statistical feature generation to capture different aspects of log behavior.

5. To apply unsupervised anomaly detection models

Models such as Isolation Forest, ECOD, and Local Outlier Factor are used to identify abnormal patterns without requiring labeled data.

6. To provide an interactive user interface for system operation

Users can upload logs, process data, detect anomalies, and visualize results through a structured and easy-to-use interface.

7. To generate visual insights for better interpretation

The system includes timeline charts, distribution plots, and filtered tables to help users understand anomaly patterns effectively.

8. To support export of results for further analysis

Outputs can be exported in formats such as CSV, JSON, and PDF, enabling external usage and reporting.

9. To ensure reproducibility through local storage

All processed data, models, and results are stored locally, allowing users to revisit and validate previous analyses.

These objectives collectively ensure that the system not only performs anomaly detection but also provides a complete, interactive, and understandable workflow for log analysis.

System Overview

The system is designed as an interactive and modular platform that allows users to perform complete log analysis from ingestion to anomaly detection and visualization within a single interface. It combines multiple processing stages into a unified workflow, enabling users to understand and control each step of the analysis process. [3]

At a high level, the system begins with log file input, where users upload raw log data in supported formats. These logs are then passed through a preprocessing layer, where normalization ensures that all entries follow a consistent structure. This step prepares the data for further analysis and reduces inconsistencies across different log sources.

The next stage involves template mining, where similar log messages are grouped into patterns using the Drain3 approach. This reduces the

complexity of raw logs by focusing on recurring structures instead of individual text variations. As a result, the system can analyze behavior at a pattern level rather than at the raw message level.[10]

Following this, the system performs feature extraction, where log templates are transformed into numerical representations. This includes techniques such as TF-IDF vectorization, statistical feature generation, and optional semantic embeddings. These features capture both textual and behavioral characteristics of logs, forming the basis for anomaly detection.[1]

The processed features are then passed to the anomaly detection module, which applies unsupervised models such as Isolation Forest, ECOD, and Local Outlier Factor. These models identify unusual patterns by learning what constitutes normal behavior and detecting deviations without requiring labeled data.

To enhance usability, the system integrates a visualization layer that presents results through interactive charts and tables. Users can observe anomaly distributions, time-based trends, and filtered log entries, making it easier to interpret system behavior. Additionally, the system supports exporting results for further analysis or reporting.

All stages are connected through a single user interface, allowing seamless navigation between operations such as parsing logs, extracting features, running detection models, and visualizing outputs. This design ensures that even users without deep technical expertise can operate the system effectively.

Overall, the system provides a complete pipeline that balances functionality, simplicity, and interpretability. It transforms raw log data into actionable insights while maintaining a lightweight and user-friendly structure.

System Architecture

The system architecture is designed as a layered and modular pipeline where each component performs a specific function while interacting seamlessly with other modules. The architecture ensures that raw log data is progressively transformed into meaningful anomaly insights through structured processing stages. [7]

At the highest level, the architecture consists of five major layers: Input Layer, Processing Layer, Feature Engineering Layer, Detection Layer, and Visualization Layer. Each layer contributes to a specific stage in the workflow, enabling clarity, scalability, and maintainability.

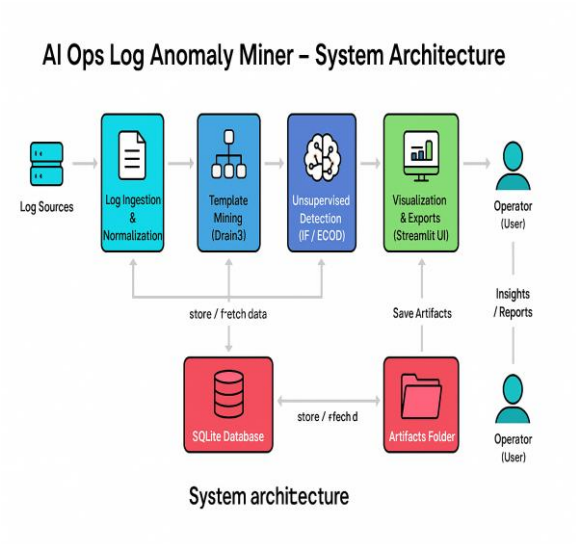

Figure 1: System Architecture

A. Input Layer (Log Ingestion)

The process begins with the input layer, where users upload log files in formats such as .log or .txt. This layer is responsible for reading raw log entries and preparing them for preprocessing. Since logs may originate from different systems, this layer acts as the entry point for handling heterogeneous data sources.

B. Processing Layer (Normalization + Template Mining)

After ingestion, logs pass through the processing layer, which includes two key components:

Normalization Module: Converts raw logs into a structured format by extracting fields such as timestamp, log level, and message. This ensures uniformity across all log entries.

Template Mining Module (Drain3): Groups similar log messages into templates by identifying constant and variable parts. This reduces noise and simplifies large volumes of logs into manageable patterns.

This layer plays a critical role in reducing data complexity and preparing it for machine learning.

C. Feature Engineering Layer

Once templates are generated, the system transforms them into numerical representations. This layer includes:

TF-IDF Vectorization: Captures textual importance of log terms

Statistical Features: Represents frequency and distribution patterns

Optional Semantic Embeddings: Enhances contextual understanding of logs

These features collectively describe system behavior in a format suitable for anomaly detection models.

D. Detection Layer (Unsupervised Models)

The feature vectors are passed to the anomaly detection layer, where multiple unsupervised models are applied:

Isolation Forest: Detects anomalies based on data isolation

ECOD: Identifies statistically extreme observations

Local Outlier Factor (LOF): Measures local density deviations

Each model contributes a different perspective, ensuring that both rare events and distribution-based anomalies are captured effectively.

E. Visualization and Output Layer

The final layer presents results to the user through an interactive interface. It includes:

Anomaly Visualization: Timeline charts and distribution graphs

Filtered Log Views: Display of anomalous log entries

Export Options: Saving results in formats such as CSV, JSON, and PDF

This layer bridges the gap between system processing and user understanding by converting model outputs into interpretable insights.

F. Data Storage and Reproducibility

Alongside the main pipeline, a storage component maintains all intermediate and final outputs:

Parsed logs

Extracted templates

Feature vectors

Trained models and results

All data is stored locally using SQLite and artifact files, ensuring reproducibility and traceability of analysis.

G. Overall Interaction Flow

The architecture follows a linear yet modular flow:

Log Input → Normalization → Template Mining → Feature Extraction → Anomaly Detection → Visualization & Storage

Each module can operate independently, allowing future extensions or replacements without affecting the entire system.

This architecture ensures that the system remains lightweight, transparent, and easy to extend, while still providing a complete end-to-end solution for log anomaly detection and analysis.

Methodology

The system follows a structured workflow where raw log data is gradually transformed into meaningful insights through a sequence of well-defined steps. Each stage in the workflow is designed to be simple, modular, and easy to understand, allowing users to interact with the system at different levels.

Step 1: Log Upload and Input Handling

The workflow begins when the user uploads a log file through the interface. The system reads the file and prepares it for processing. Since logs may vary in format, the system handles them in a flexible manner to ensure compatibility with different sources.

Step 2: Log Normalization

Once the logs are loaded, they are converted into a structured format. Key fields such as timestamp, log level, and message content are extracted. This step ensures consistency across all log entries and removes irregularities that could affect further analysis.

Step 3: Template Mining (Drain3 Processing)

The structured logs are then processed using the Drain3 algorithm. Similar log messages are grouped into templates by identifying common patterns. This reduces redundancy and allows the system to focus on meaningful patterns instead of individual variations in text.

Step 4: Feature Extraction

After template generation, the system converts the processed logs into numerical features. Multiple techniques are used to capture different aspects of log behavior:

Text-based features using TF-IDF

Statistical features such as frequency and distribution

Optional semantic representations for contextual understanding

These features represent the system's operational behavior in a machine-readable format.

Step 5: Anomaly Detection

The extracted features are passed to unsupervised models for anomaly detection. The system applies multiple algorithms to ensure robust detection:

Isolation Forest identifies rare and isolated patterns

ECOD detects statistically extreme values

Local Outlier Factor analyzes local density variations

Each model assigns anomaly scores, and the system combines these insights to identify abnormal log entries.

Step 6: Result Generation and Interpretation

Once anomalies are detected, the system processes the results to make them understandable. It highlights unusual patterns such as sudden increases in specific log templates or irregular behavior over time. This step converts raw model outputs into meaningful insights.

Step 7: Visualization and User Interaction

The results are presented through interactive visualizations. Users can explore anomaly

trends, filter log entries, and observe distributions using charts and tables. This helps in quick understanding and decision-making.
Step 8: Export and Storage
Finally, the system allows users to export results in different formats such as CSV, JSON, and PDF. All processed data, models, and outputs are also stored locally, ensuring reproducibility and future analysis.
Overall Workflow Summary
Log Upload → Normalization → Template Mining → Feature Extraction → Anomaly Detection → Visualization → Export
This workflow ensures that the system remains user-friendly while maintaining a clear and logical processing pipeline. It allows users to move step by step from raw data to actionable insights without requiring deep technical knowledge.

System Implementation

The system implementation focuses on providing an interactive and user-friendly interface where all stages of log analysis are accessible through clearly defined controls. The implementation integrates log ingestion, preprocessing, feature extraction, anomaly detection, and visualization into a single workflow, allowing users to operate the system step by step.

A. Main Dashboard Interface

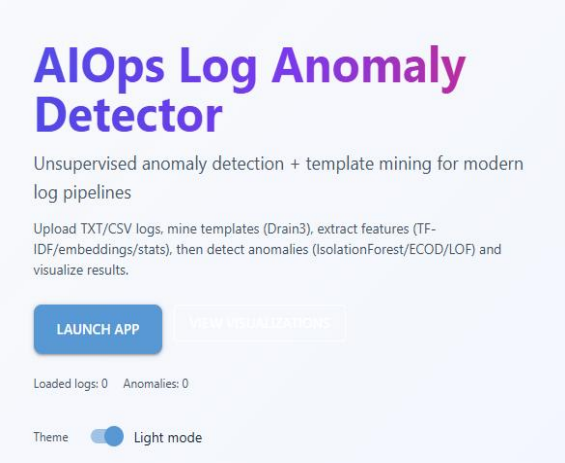

Figure 2: Main Dashboard Interface

This screen represents the central control panel of the system. It provides options for uploading log files, selecting processing steps, and initiating different operations such as parsing, feature extraction, and anomaly detection. The dashboard acts as the entry point for all functionalities, ensuring that users can navigate the system easily without switching between multiple tools.

B. Log Upload and Parsing Module

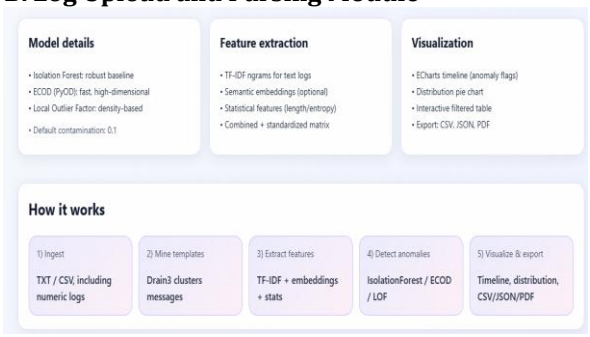

Figure 3: Log Upload and Parsing Screen

This interface allows users to upload log files in supported formats. Once uploaded, the system parses the logs and converts them into a structured format. This step is important because it prepares raw log data for further processing and ensures consistency across different log sources.

C. Template Mining and Preprocessing View

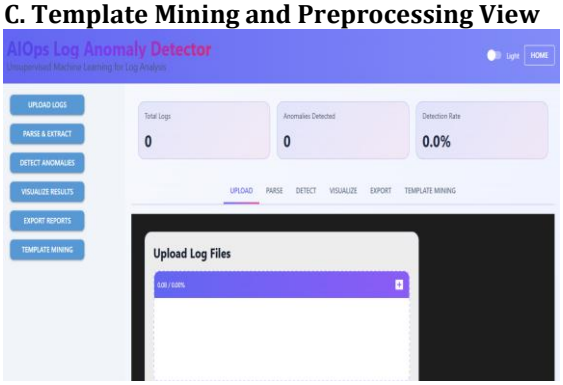

Figure 4: Template Mining Output

This screen displays the results of template extraction using the Drain3 approach. Similar log messages are grouped into templates, reducing redundancy in the data. This stage helps in simplifying large volumes of logs and enables efficient pattern analysis in later stages.

D. Feature Extraction Module

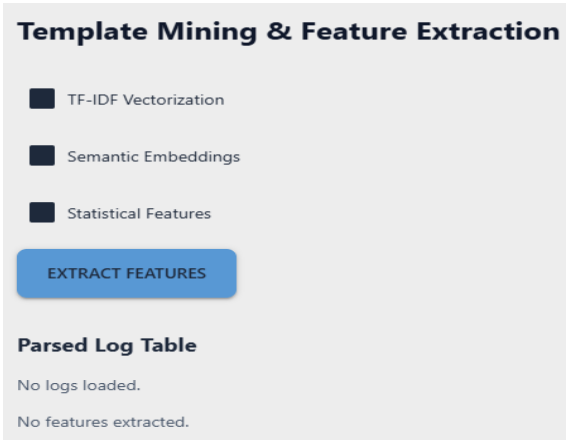

Figure 5: Feature Extraction and Representation

In this interface, the system generates numerical features from log templates. It shows how text-based and statistical features are derived from processed logs. This step is critical as it converts log data into a format suitable for machine learning models.

E. Anomaly Detection Module

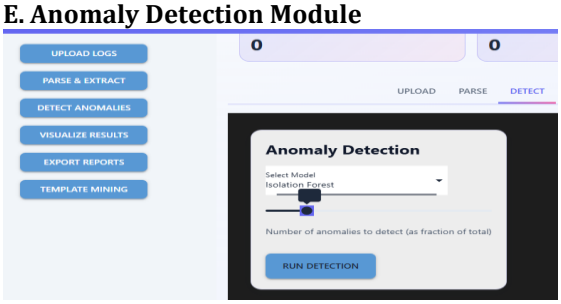

Figure 6: Anomaly Detection Results

This screen presents the output of anomaly detection models such as Isolation Forest, ECOD, and Local Outlier Factor. It highlights anomalous log entries and assigns scores to indicate their deviation from normal behavior. This module forms the core analytical component of the system.

F. Visualization and Analysis Dashboard

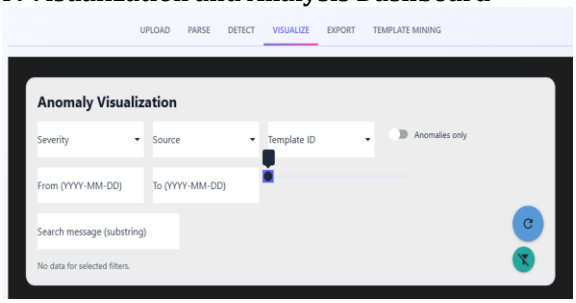

Figure 7: Visualization Dashboard – Timeline and Distribution

This interface provides graphical representations of anomalies, including time-based trends and distribution patterns. Users can visually inspect when anomalies occur and how they are distributed across log templates. This improves interpretability and helps in identifying patterns quickly.

G. Filtered Results and Export Module

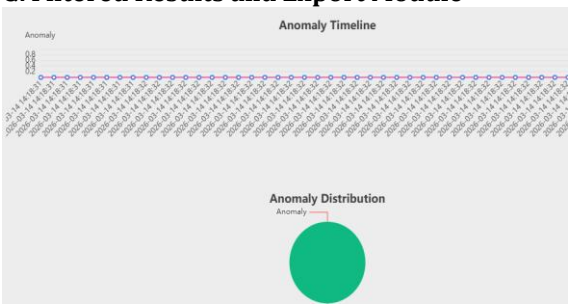

Figure 8: Filtered Results and Export Options

This screen allows users to filter detected anomalies and export results in formats such as CSV, JSON, or PDF. It supports further analysis and reporting, making the system useful for both operational and academic purposes.

Overall Implementation Flow

The system implementation connects all modules into a seamless workflow where users can move from log upload to final analysis without interruption. Each screen corresponds to a specific stage in the pipeline, ensuring clarity and ease of use.

The design emphasizes simplicity, interactivity, and transparency, allowing users to understand not only the results but also the process behind them.

Discussion

The developed system demonstrates a practical approach to log anomaly detection by focusing on usability, transparency, and lightweight deployment. Instead of relying on complex infrastructure or black-box models, the system provides a step-by-step workflow that users can easily understand and operate.

One of the key strengths of the system is its interactive design. Users are not limited to a single automated pipeline; instead, they can control each stage of processing—from log upload to anomaly detection and visualization. This makes the system suitable for learning, experimentation, and real-world debugging scenarios where users need visibility into intermediate steps.

Another important aspect is the use of unsupervised learning models. Since real-world log data rarely comes with labeled anomalies, the system avoids dependency on predefined datasets. Models such as Isolation Forest, ECOD, and Local Outlier Factor enable detection of unknown and evolving anomalies. This makes the system adaptable to different types of logs without requiring retraining with labeled data.

The integration of template mining and feature extraction significantly improves analysis quality. By grouping similar log messages into templates, the system reduces noise and focuses on meaningful patterns. Feature extraction techniques further enhance this by representing logs in a structured numerical format, allowing models to capture both textual and behavioral characteristics.

The visualization capabilities play a critical role in making results understandable. Instead of presenting raw anomaly scores, the system displays trends, distributions, and filtered results through charts and tables. This helps users quickly identify unusual behavior and understand when and where anomalies occur.

Such visual feedback is especially useful for debugging and decision-making.

From a practical perspective, the system offers several advantages:

- Lightweight Deployment: Runs on a local machine without cloud dependency
- Ease of Use: Simple interface suitable for beginners and researchers
- Explainability: Clear interpretation of anomaly results
- Reproducibility: Local storage of data and models for repeated analysis

However, the system also has certain limitations. Since it is designed to be lightweight, it may not scale efficiently for very large enterprise-level log data. Additionally, while unsupervised models are flexible, they may require tuning for optimal performance depending on the dataset characteristics.

Overall, the system achieves a balance between functionality and simplicity. It provides a complete workflow for log analysis while maintaining transparency and ease of use, making it suitable for academic projects, research applications, and small-scale deployments.

Conclusion

The developed system provides a complete and practical solution for log anomaly detection by combining system design, implementation, and visualization within a single framework. It addresses the limitations of traditional log analysis methods by introducing an automated and interactive approach that does not rely on predefined rules or heavy infrastructure.

The system successfully integrates key components such as log ingestion, normalization, template mining, feature extraction, and unsupervised anomaly detection into a unified workflow. By using models like Isolation Forest, ECOD, and Local Outlier Factor, it is able to identify abnormal patterns without requiring labeled data. This makes the system adaptable to real-world scenarios where anomalies are not explicitly defined.

A major contribution of the system is its focus on usability and explainability. Through an intuitive interface and visual dashboards, users can easily understand the analysis process and interpret results. The inclusion of visualization tools and export options further enhances its practical usefulness for both academic and operational purposes.

The lightweight and offline nature of the system ensures that it can be deployed on standard machines without dependency on cloud services. This makes it especially valuable for educational institutions, research environments, and small organizations that require efficient yet simple log analysis solutions.

In summary, the system demonstrates that effective log anomaly detection can be achieved without complex infrastructure, while still maintaining clarity, flexibility, and user interaction. The modular design also allows future enhancements, enabling the system to evolve with new techniques and requirements.

References

Du, M., Li, F., Zheng, G., & Srikumar, V. (2017). DeepLog: Anomaly detection and diagnosis from system logs through deep learning. Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS), 1285–1298. https://doi.org/10.1145/3133956.3134015

Zhang, X., Xu, H., Zhao, W., Lin, Y., Xu, J., Xu, Z., & Pei, D. (2019). Robust log-based anomaly detection on unstable log data. Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE), 807–817. https://doi.org/10.1145/3338906.3338931

Meng, W., Li, Y., Zhang, L., Zhang, X., Xu, H., Zhang, Y., & Pei, D. (2019). LogAnomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI), 4739–4745. https://doi.org/10.24963/ijcai.2019/657

Lu, S., Wei, J., Li, Q., Wang, Z., & Wang, Z. (2018). Detecting anomaly in big data system logs using convolutional neural networks. 2018 IEEE International Conference on Data Mining Workshops (ICDMW), 127–134. https://doi.org/10.1109/ICDMW.2018.00025

Guo, H., Chen, Z., & Xu, H. (2021). LogBERT: Log anomaly detection via BERT. arXiv preprint arXiv:2103.04475. https://arxiv.org/abs/2103.04475

Chen, Z., et al. (2021). Deep learning-based system log analysis for anomaly detection: A toolkit and benchmark evaluation. arXiv preprint arXiv:2109.08006. https://arxiv.org/abs/2109.08006

Yu, L., Chen, Z., Lou, J. G., Zhang, D., & Zhang, H. (2022). Adaptive and semantic-aware log parsing with high accuracy. IEEE Transactions on Knowledge and Data Engineering, 34(10),

4690–4703. https://doi.org/10.1109/TKDE.2021.3059432

Liu, Q., Fu, Y., Cheng, Z., & Pei, D. (2022). Log2vec++: Contrastive learning and multi-scale masking for robust log anomaly detection. arXiv preprint arXiv:2205.01883. https://arxiv.org/abs/2205.01883

He, P., Zhu, J., He, S., Li, J., Lyu, M. R., & Zhang, M. (2021). Drain3: Streaming log template mining in real time. 2021 IEEE International Conference on Software Maintenance and Evolution (ICSME), 186–190. https://doi.org/10.1109/ICSME52107.2021.00027

Zhu, J., He, S., Chen, Z., & Lyu, M. R. (2020). Loghub: A large collection of system log datasets towards automated log analytics. arXiv preprint arXiv:2008.06448. https://arxiv.org/abs/2008.06448

Kim, T., & Kim, S. (2022). Deep learning for anomaly detection in system logs: A survey. Information and Software Technology, 143, 106722. https://doi.org/10.1016/j.infsof.2022.106722

Zhang, J., et al. (2020). A component-aware approach to log-based anomaly detection and localization. IEEE Transactions on Reliability, 69(1), 12–24. https://doi.org/10.1109/TR.2019.2909277

Hadadi, F., et al. (2024). Embedding and model type combinations for anomaly detection in system logs: An empirical study. Empirical Software Engineering, 29(3), 1–27. https://doi.org/10.1007/s10664-024-10322-y

Khan, Z. A., et al. (2024). On the impact of log parsing accuracy for anomaly detection. Empirical Software Engineering, 29(2), 17–39. https://doi.org/10.1007/s10664-024-10305-z

Himler, P., et al. (2024). Federated anomaly detection in system log sequences. Journal of Big Data, 11(4), 97. https://doi.org/10.1186/s40537-024-00972-3

Han, X., et al. (2023). LogGPT: Log anomaly detection using generative pretrained transformers. arXiv preprint arXiv:2306.01985. https://arxiv.org/abs/2306.01985

Guan, W., et al. (2025). LogLLM: Combining BERT and Llama for log anomaly detection. arXiv preprint arXiv:2502.00456. https://arxiv.org/abs/2502.00456

Nipa, N. A., et al. (2025). Comparative study of log anomaly detection techniques on real-world datasets. Proceedings of the 10th International Conference on Internet of Things, Big Data and Security (IoTBDS 2025), 190–201. https://doi.org/10.5220/0012345600003456