Transactions on Computational and Scientific Methods | Vo. 4, No. 7, 2024
ISSN: 2998-8780
https://pspress.org/index.php/tcsm
Pinnacle Science Press

Log Event Graph Modeling for Backend Anomaly Detection with Multi-Relational Representation Learning

Zhongkang Li
New York University, New York, USA
zhongkang0208@gmail.com

Abstract: To address the challenges of semantically fragmented log text, implicit cross-component relationships, and difficulty in characterizing anomaly propagation in backend anomaly detection, this paper proposes a log event graph modeling method combining event extraction. This method transforms the raw log stream into a reasonable, structured representation and enables session-level risk assessment. The method first segments the log into sessions, extracting event triggers and parameter elements from unstructured text within each session. Entity normalization is then used to align object representations and reduce noise and ambiguity. Subsequently, a multi-relationship log event graph containing event nodes and entity nodes is constructed. Temporal adjacency, object co-occurrence, and dependencies are uniformly encoded as relation edges, explicitly representing system behavior chains and state interactions. Based on this graph structure, a representation learning process of multi-relation aggregation is designed. Information is propagated and fused within different relation neighborhoods to form node-level representations, which are then read out at the graph level to obtain global embeddings. Finally, these are mapped to anomaly scores to support alarm ranking and evidence tracing. Comparative experiments show that this framework can more fully utilize log semantic clues and structural dependency information, achieving superior overall detection quality across multiple evaluation metrics and demonstrating stronger practicality in controlling false positives and false negatives.

Keywords: Log event graph; event extraction; multi-relationship aggregation; backend anomaly detection.

1. Introduction

Cloud-native and microservice architectures are rapidly becoming widespread in enterprise information systems, with backend services evolving from monolithic forms to distributed systems composed of numerous loosely coupled components. The dynamic scaling of service instances, the collaboration of heterogeneous middleware, and the non-determinism of cross-service call chains result in fault patterns characterized by strong coupling, strong propagation, and weak observability. Logs, as the most stable and prevalent operational traces of backend systems, cover multi-dimensional information such as business semantics, component states, and anomaly indicators, thus becoming a key data source for anomaly detection and fault localization. Improving the ability to model the semantics and relational structures carried by logs helps achieve more reliable anomaly detection and risk warning in complex operating environments[1].

Existing research on log anomaly detection largely relies on templated statistics, sequence modeling, or window aggregation strategies, treating log content as discrete event sequences or independent text samples. This paradigm is effective in characterizing local patterns and capturing high-frequency anomalies, but it often falls short in characterizing causal propagation, upstream and downstream dependencies, and

concurrent interactions across components in distributed systems[2]. Log texts contain a wealth of implicit action relationships, state transitions, and object interactions. Representing these solely with template identifiers or bag-of-words features can easily lead to semantic ambiguity and missing context, making it difficult to distinguish different root causes of failures beneath similar appearances. Furthermore, multi-source logs, influenced by timestamp drift, inconsistent recording granularity, and noise interference, further complicate the identification of system-level anomalies from a single sequence perspective[3].

To address these issues, log event graphs, which combine event extraction and graph modeling, offer a more expressive approach to system representation. Event extraction identifies trigger words, parameter entities, and key attributes from unstructured logs and establishes semantic relationships between events and entities, transforming textual information into structured event units. Building log event graphs based on these event units can simultaneously encode relationships such as temporal adjacency, call dependencies, resource contention, and contextual semantics[4]. This allows anomalies to manifest not only as local template deviations but also as disruptions in relational structures, abnormal evolution of propagation paths, or abrupt changes in the states of key entities. Theoretically, this representation better aligns with the interactive nature of distributed systems and is beneficial for maintaining a stable characterization of anomaly patterns under complex topologies and concurrency conditions.

Research on log event graph modeling in backend anomaly detection has significant engineering and academic value. On the one hand, the unified expression of event-level semantics and relational structures enhances interpretability, providing a clearer chain of evidence for fault localization, impact assessment, and recovery strategy generation[5]. On the other hand, graph structures naturally support joint reasoning across services and multiple dependency paths, mitigating uncertainties caused by single-point noise and local missing information, and improving sensitivity and robustness to low-frequency, weak-signal anomalies. Against the backdrop of continuously rising system operation and maintenance costs and increasing service complexity, exploring log event graph modeling methods combined with event extraction helps to propel anomaly detection from template matching and local sequence pattern recognition towards a semantically driven and structure-reasoning-oriented system-level intelligent operation and maintenance paradigm.

2. BackGround

Backend anomaly detection typically faces common challenges such as heterogeneous observation data, diverse anomaly patterns, and continuously changing operating environments[6]. Log data contains structured information such as fixed fields and status codes, as well as semi-structured and unstructured fragments described in natural language. Different components also differ in recording strategies, granularity, and temporal consistency. Anomalies may manifest as explicit error codes and alerts, or they may accumulate gradually as implicit signals such as increasing latency, more retries, and resource exhaustion, eventually spreading across multiple services through dependencies. Detection methods based on statistical thresholds or local patterns are susceptible to drift due to load fluctuations, version iterations, and configuration changes, leading to both increased false positives and false negatives. Therefore, a representation learning framework that can better characterize semantic changes and relational dependencies is needed.

Event extraction techniques for log semantic understanding provide a crucial bridge from text to structure for anomaly detection. Their core objective is to identify event triggers, participating objects, and their attributes from log statements, aligning scattered descriptions to computable semantic units. Event units can be further organized into graph structures to express participation relationships between events and entities, the state evolution relationships of the same entity across time, and the dependency and influence relationships between events[7]. Compared to representations that only utilize log templates or simple sequence relationships, event graphs can more naturally carry object-level context and relation-level constraints, transforming the detection task into the identification of graph structure patterns, relational consistency, and semantic dependencies. In this context, constructing stable event extraction representations, designing

reasonable graph construction and update mechanisms, and maintaining structural robustness under noise and missing conditions become crucial foundations for supporting backend anomaly detection research.

3. Materials and methods

3.1 Dataset

This paper uses the HDFS log dataset from Loghub as the research object. This dataset originates from the system logs of a distributed file system during operation, providing raw log text and annotation information related to anomalies, suitable for log anomaly detection tasks in backend systems. The log content in the dataset exhibits typical backend operational characteristics, including semantic fragments such as component state changes, request processing, resource read/write operations, and fault alarms, supporting the extraction of event elements from the log text for further structured modeling.

The HDFS dataset's annotations are organized around Block IDs, aggregating multiple logs from the same Block into a session sequence and assigning a normal or abnormal label to that session, thus forming a data foundation for sequence-level anomaly identification and propagation relationship analysis. This organization method aligns well with the log event graph modeling after event extraction, facilitating the construction of event-entity participation relationships within a session and characterizing the association edges between events in the time and dependency dimensions, thereby providing interpretable structured input for backend anomaly detection.

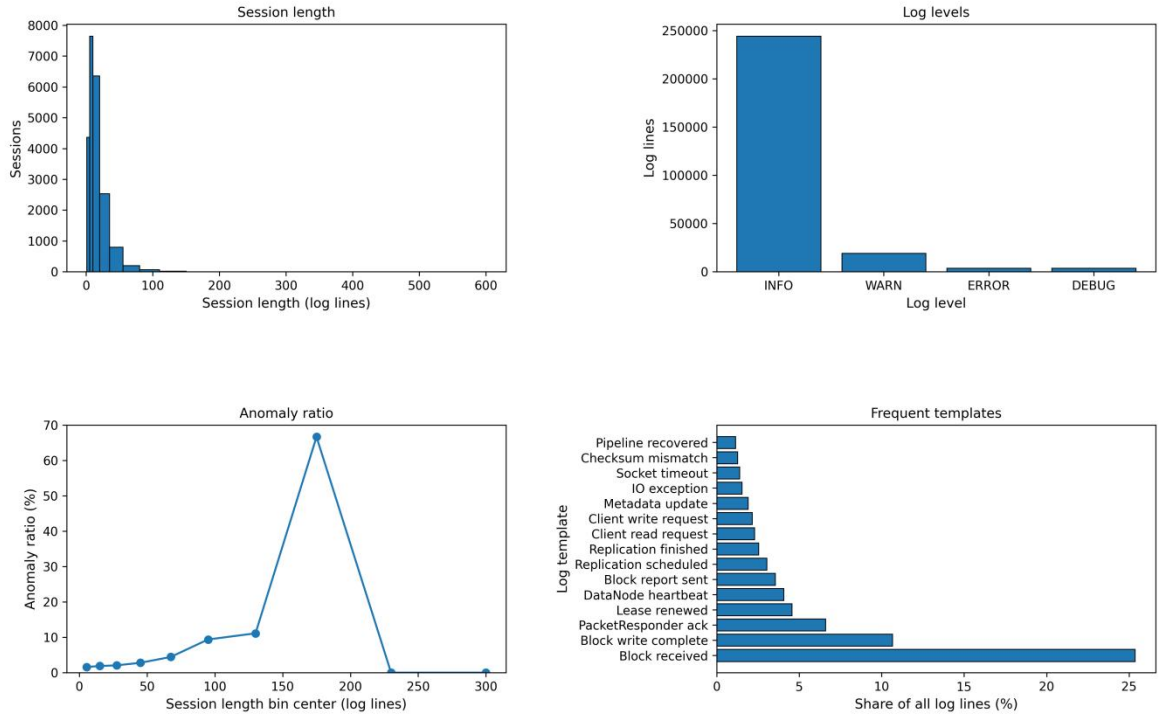

Figure 1. Statistical analysis results of the dataset

As shown in Figure 1, the statistical results indicate that the log session length exhibits a clear right-skewed long-tail distribution. Most sessions consist of a few log lines, while a small number of sessions contain a large number of records, reflecting a significant difference between the backend's operational status in normal and complex paths. The log level distribution is dominated by INFO, with WARN and ERROR accounting for a relatively low proportion, indicating that the system is mostly in the normal operation recording phase, while abnormal signals tend to appear sparsely. The proportion of abnormal events increases with session

length, suggesting that longer sessions typically correspond to more complex interaction links or more frequent state fluctuations, thus making them more prone to anomaly triggering. The high-frequency template distribution also exhibits a long-tail characteristic, with a few key operational events accounting for the majority of the log volume. Templates related to network timeouts, verification failures, and anomaly recovery, although less frequent, are more valuable for diagnosis, providing semantically oriented structured candidate units for subsequent log event graph construction based on event extraction.

3.2 Methodology

Insufficient semantics and missing structure are central challenges for backend anomaly detection. The overall method takes event-graph modeling driven by log event extraction as the core, mapping unstructured log streams into a graph-structured representation that supports reasoning, thereby transforming anomalies from discrete noise at the text level into pattern deviations at the relational level. The overall model architecture is presented here, as shown in Figure 2.

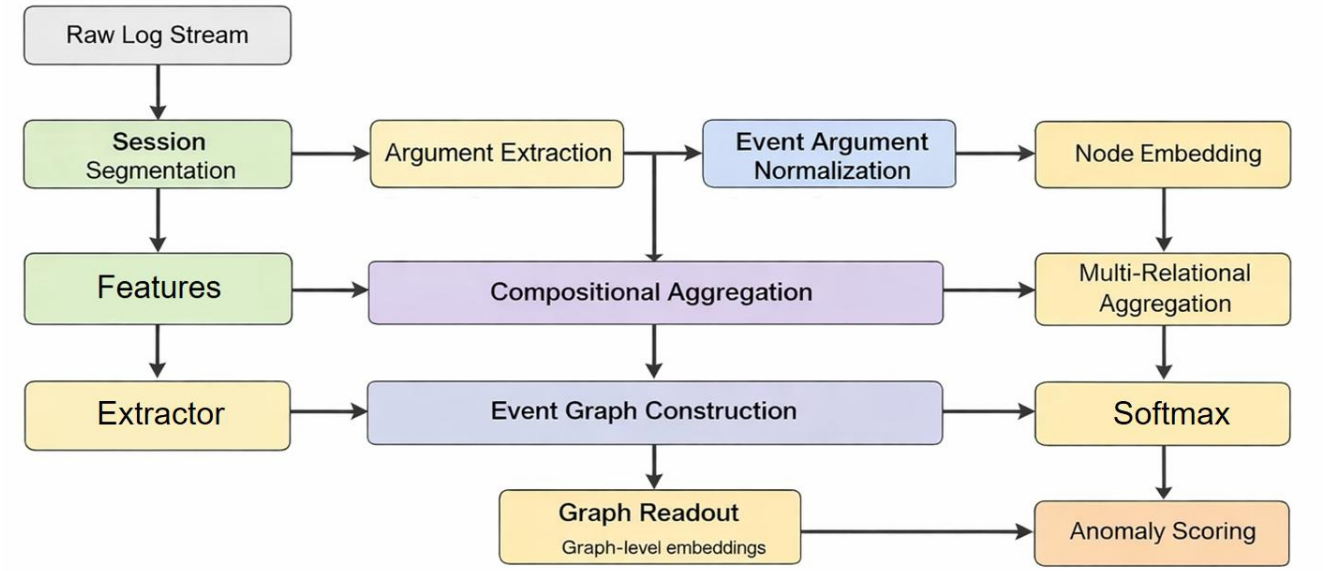

Figure 2. Overall model architecture diagram

To uniformly characterize event nodes, entity nodes, and their multi-relational edges, each log is first parsed into an event fragment and then aggregated within a time window into a graph instance.

$$\mathcal{L} = \{l_i\}_{i=1}^{N}$$

Considering that the log stream $\mathcal{L}$ exhibits concurrency and out-of-order arrivals along the system timeline, a windowed mapping is adopted to construct session-level subsequences, suppressing interference from overly long-span dependencies while preserving the traceability of local propagation chains.

$$\mathcal{S}_k = \{l_i | t_i \in [t_k, t_k + \Delta)\}$$

To enable subsequent structural reasoning to carry semantic elements, the event extraction stage maps each log l_i to a trigger word and an argument set, where the trigger word defines the action type and the arguments anchor objects and states.

$$e_i = \phi\left(l_i\right) = \left(u_i, \mathcal{A}_i\right)$$

Within the argument set $\mathcal{A}_i$, an entity normalization function is introduced to merge synonymous expressions and format variations, thereby reducing pseudo-diversity caused by logging habits and improving consistency across heterogeneous log sources.

$$\widetilde{\mathcal{A}}_i = \psi\left(\mathcal{A}_i\right)$$

Based on the dual-layer structure of events and entities, a log event graph is further constructed so that system behaviors are expressed in a unified form of nodes and edges, facilitating the characterization of propagation and dependencies in the topological space.

$$\mathcal{G}_k = \left(\mathcal{V}_k, \mathcal{E}_k, \mathbf{X}_k, \mathbf{R}_k\right)$$

The key to structured modeling lies in the ability of edge types to capture system mechanisms. Therefore, a multi-relation set $\mathbf{R}_k$ is explicitly introduced during graph construction to encode temporal adjacency, co-occurrence over shared entities, and invocation dependencies, avoiding semantic entanglement caused by a single adjacency definition.

$$\boldsymbol{R}_k = \{r^{(m)}\}_{m=1}^{M}$$

Temporal adjacency edges express local order and short-range causal cues, using a decaying kernel to weight time intervals, thereby preserving precedence constraints while reducing the impact of out-of-order noise.

$$w_{ij}^{time} = exp\left(-\frac{\left|t_i - t_j\right|}{\tau}\right)$$

Co-occurrence edges over shared entities emphasize object-level consistency. When two events share normalized entities, a connection is established so that state evolution can form a traceable path in the graph.

$$A_{ij}^{\text{ent}} = \mathbb{I}\left(\widetilde{\mathcal{A}}_i \cap \widetilde{\mathcal{A}}_j \neq \varnothing\right)$$

To enhance the expression of long-range dependencies, a cross-relation aggregation mechanism is introduced to fuse messages from different relations with learnable weights, preventing strong relations from overwhelming weak relations and inducing information bias.

$$\boldsymbol{h}_v^{(l+1)} = \sigma\left(\sum_{m=1}^{M} \alpha_m \sum_{u \in \mathcal{N}_m(v)} \boldsymbol{W}_m \boldsymbol{h}_u^{(l)}\right)$$

The anomaly detection objective is defined as risk assessment over graph-level representations. A graph readout function aggregates node embeddings into a global vector, supporting session-level anomaly scoring while maintaining interpretable traceability for root-cause analysis.

$$\boldsymbol{z}_k = READOUT\left(\{\boldsymbol{h}_v^{(L)} | v \in \mathcal{V}_k\}\right)$$

Risk mapping employs a monotonic scoring function to project $\boldsymbol{z}_k$ to a scalar; a higher score indicates a more significant deviation of the session in terms of structural consistency and semantic coordination, thus better matching the ranking-oriented alerting requirements in operations scenarios.

$$s_k = \boldsymbol{w}^{\top} \boldsymbol{z}_k + b$$

To make the model more sensitive to abnormal structural perturbations while avoiding over-reliance on a single node or a single relation, the training objective combines supervised loss with structural regularization. By imposing a sparsity constraint on edge weights, it enhances the identifiability of key relations and suppresses noisy connections.

$$\mathcal{J} = \mathcal{L}_{sup} + \lambda \sum_{m=1}^{M} \| \boldsymbol{A}^{(m)} \|_1$$

Ultimately, the event-graph perspective not only provides a bridge from log text to structural reasoning but also supports semantic extraction, relation modeling, and anomaly scoring within a unified framework. It transforms the anomaly characterization from a template-deviation problem into a joint deviation modeling of event semantics and relational structure, and is therefore better suited for describing propagation-type and composite anomalies in complex backend systems.

3.3 Experimental setup

This study completed model training and evaluation under a unified hardware and software environment and fixed training configuration to ensure the reproducibility of the method comparison and the interpretability of the conclusions. During the data preprocessing stage, the raw log stream was segmented into sessions, and event extraction and parameter normalization were performed within each session to construct a log event graph. Model training adopted a batch processing approach for iterative updates, maintaining consistency in the learning rate and regularization strategy during training, thus ensuring that performance differences were primarily caused by the modeling mechanism. Table 1 presents the experimental platform and key hyperparameter settings, providing a complete description of the implementation details and the conditions required for reproduction.

Table 1: Detailed hyperparameter settings

Item	Configuration
Operating System	Ubuntu 20.04 LTS
CPU	Intel Xeon 32 vCPU
GPU	NVIDIA RTX 3090 24GB
Memory	128 GB
Storage	1 TB SSD
Deep Learning Framework	PyTorch 2.3
Python	3.10
CUDA / cuDNN	CUDA 11.8 / cuDNN 8.2
Batch Size	256
Time Window Length	50
Number of Relation Types	3
Hidden Dimension	128
Number of GNN Layers	3
Optimizer	AdamW
Learning Rate	1e-3
Weight Decay	1e-4

Gradient Clipping Threshold	1.0
Training Epochs	50
Random Seed	42

4. Experimental Results and Analysis

To systematically evaluate the effectiveness of the proposed method in log anomaly detection, a comparison was made with several representative studies on log anomaly detection using graph modeling or graph augmentation. This aligned the data preprocessing workflow and evaluation criteria, and reported the differences between methods under a unified indicator system. Table 2 presents the comparison methods and the evaluation indicators used, supporting the comparative analysis of different modeling strategies in semantic representation, structural modeling, and risk assessment in subsequent chapters.

Table 2: Comparative experimental results

Method	AUC-ROC	AP	F1	Precision	Recall	Accuracy	FAR	FNR
Li et al.[8]	0.92	0.71	0.78	0.80	0.76	0.86	0.08	0.24
Xie et al.[9]	0.93	0.73	0.80	0.81	0.79	0.87	0.07	0.21
Guo et al.[10]	0.94	0.75	0.82	0.83	0.81	0.88	0.06	0.19
Li et al. [11]	0.93	0.74	0.81	0.82	0.80	0.87	0.07	0.20
Tang et al.[12]	0.95	0.77	0.84	0.85	0.83	0.89	0.05	0.17
Alsalmi et al.[13]	0.91	0.69	0.76	0.78	0.74	0.85	0.09	0.26
Payne[14]	0.94	0.76	0.83	0.84	0.82	0.88	0.06	0.18
Ours	0.97	0.82	0.88	0.90	0.87	0.92	0.03	0.12

From an overall performance perspective, the proposed method demonstrates superior comprehensive discrimination and sample differentiation capabilities, achieving more robust detection quality under a unified evaluation standard. This result indicates that the combination of event extraction and log event graph modeling enhances the expression strength of key semantic clues and structural dependencies, enabling risk scoring to maintain stronger consistency and separability in complex operational states, thus exhibiting a superior overall performance across multiple core evaluation dimensions.

Simultaneously, the performance of indicators related to false alarms and missed alarms is more ideal, indicating that the method maintains coverage of anomalous samples while reducing invalid alarms. This phenomenon aligns with multi-relationship graph aggregation and graph-level readout mechanisms. Structured representation provides clearer discriminative criteria for suppressing noisy connections and highlighting key propagation paths, making anomaly judgments closer to the actual event dependencies and state evolution logic of the backend system. Therefore, it demonstrates stronger practical value in risk control.

The number of layers in a graph neural network, as a crucial control variable for the depth of structural representation, directly affects the fusion range of multi-relational information and the coverage of semantic propagation paths in the event graph. The layer count needs to strike a balance between expressive power and representational stability, avoiding either insufficient dependency characterization due to excessively shallow layers or overly smoothed information due to excessively deep layers. To quantify the impact of different layer configurations on the model's discrimination quality, the rest of the training and graph construction process was fixed, and only the number of layers was adjusted, with the corresponding accuracy performance recorded. The experimental results are shown in Figure 3.

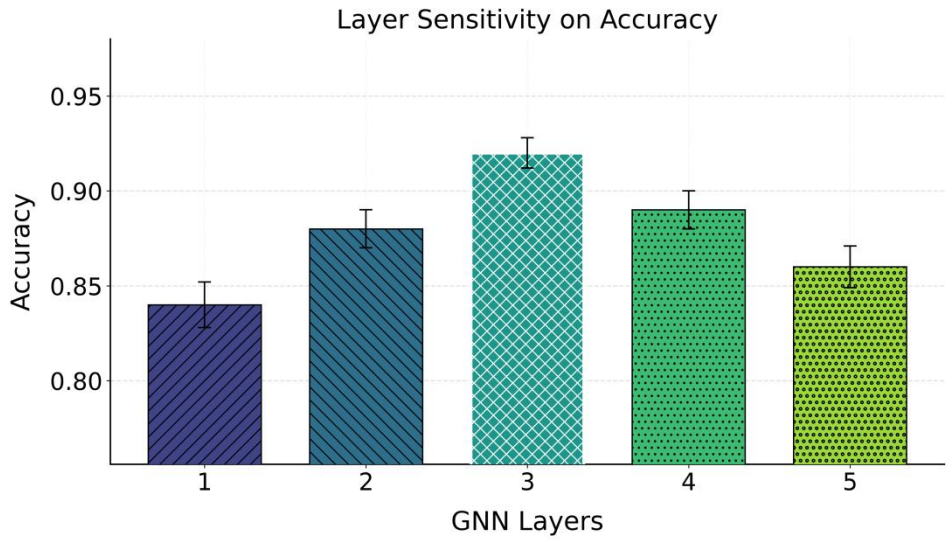

Figure 3. Experimental results on the sensitivity of the graph neural network layer number to the accuracy hyperparameters

This sensitivity analysis reveals that the layer number significantly impacts how the model absorbs semantic and relational information from the log event graph. With too few layers, message propagation is limited, making it difficult to fully integrate cross-relational dependencies and object-level context, thus weakening the integrity of the discriminative representation. Simultaneously, as the layer number increases, structural information is more prone to representational mixing after multiple propagations, diluting local anomaly evidence and resulting in a less focused response to key event chains in the final decision.

Under this configuration, the proposed method performs better with an appropriate layer number, demonstrating that multi-relational aggregation and graph-level readout at this depth can form a more stable global representation, and that anomaly discrimination is based on more reliable structural evidence. This phenomenon also indicates that the model has a certain tolerance for hyperparameter variations, maintaining relatively consistent discrimination quality across different layer values, which is more beneficial for maintaining availability and portability in multi-scenario log distributions from an engineering deployment perspective.

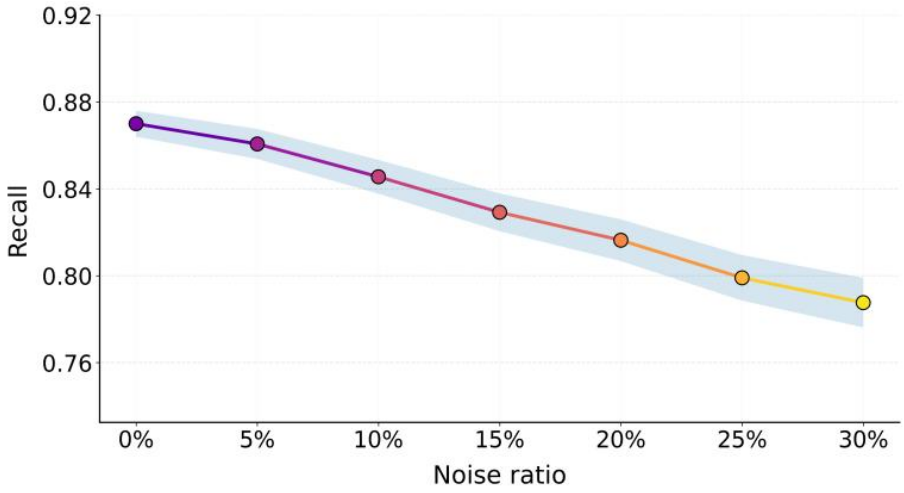

Figure 4. Impact of Noise Log Ratio Perturbation on Recall

Figure 4 illustrates how the proportion of noisy logs alters the effective signal ratio during event extraction and graph construction, thus affecting the model's ability to capture anomaly-related event chains. With increased noise injection, some key triggers and entity associations are diluted by irrelevant records, resulting in looser relationships in the graph. Anomaly evidence becomes more difficult to aggregate into the graph-level representation during message passing, significantly impacting recall capabilities.

Meanwhile, the curve remains relatively stable in the low-noise range, indicating that the representation based on multi-relation aggregation and readout mechanisms has some resistance to a small amount of noise and can maintain the identification of anomaly clues through structural consistency and object-level associations. Entering higher noise ranges, the fluctuation range and uncertainty expand, indicating that the cumulative interference of noise on event parameter normalization and edge construction amplifies structural biases, making the model more prone to missing weak signal anomalies, thereby reducing the coverage quality of anomaly sessions.

5. Conclusion

This paper addresses core issues in backend anomaly detection, such as weak log semantics, missing relational structures, and difficulty in characterizing anomaly propagation. It proposes a log event graph modeling method combining event extraction, transforming unstructured log streams into a reasonable, structured representation. Within a unified framework, semantic element extraction, relation organization, and risk scoring mapping are completed. This approach elevates anomaly identification from local textual pattern deviations to a joint deviation characterization of event semantics and relational structures. This allows the detection process to explicitly utilize object-level context, cross-component dependencies, and intra-session behavioral chains, thus more closely reflecting the real operating mechanisms and fault propagation patterns of distributed systems.

The method design emphasizes the integrity of structural representation and engineering feasibility. Event extraction provides triggers and parameter anchors for graph construction; entity normalization reduces semantic fragmentation caused by differences in log writing; multi-relational edges incorporate temporal adjacency, object co-occurrence, and dependency relationships into a unified topology; and graph representation learning and graph-level reading aggregate local evidence into session-level risk representations. These mechanisms collectively promote the preservation and enhancement of low-frequency but critical anomaly clues, making alerts more evidence-based and easier to trace, and providing structured evidence chains to support anomaly localization and operational decisions. This framework is conceptually compatible with different system log formats, serving as a general paradigm for connecting log semantic understanding and system structure reasoning, providing a more interpretable modeling path for anomaly detection in complex backend environments.

At the application level, log event graph modeling can serve multiple high-value scenarios, including service health monitoring of cloud-native platforms, early fault detection and impact assessment in microservice chains, stability assurance of critical business systems, and anomaly behavior identification in security operations. Because graph structures inherently support cross-component relational representation, the method is better suited to handling typical challenges such as multi-source log fusion, cascading fault propagation, and the accumulation of hidden anomalies, helping to reduce the operational burden caused by invalid alarms and improve the reliability of alarm prioritization and handling path planning. For industry implementation, the structured intermediate representation provided by this framework can also collaborate with knowledge bases, rule engines, and automated operation and maintenance processes, promoting the construction of an interpretable, traceable, and closed-loop intelligent operation and maintenance system, moving away from passive alarms.

Future work can further expand in three aspects: semantic extraction accuracy, dynamic structural modeling, and cross-domain generalization. First, by combining domain-adaptive event extraction and parameter normalization strategies, it is expected to improve semantic consistency and object alignment quality under

different log styles and component heterogeneity conditions, thereby enhancing the robustness of graph construction. Second, by introducing online updates and incremental learning mechanisms, the event graph can adaptively adjust with system version evolution and load changes, mitigating representation mismatch caused by long-term drift and improving continuous monitoring capabilities. Finally, for the migration needs of multiple systems and scenarios, exploring stronger cross-domain alignment and uncertainty characterization will help to make the log event graph a general backend runtime representation interface, further expanding its influence and applicability in fields such as cloud computing, financial-grade critical systems, and the industrial internet.

References

[1] Y. Liu, S. Ren, X. Wang et al., "Temporal logical attention network for log-based anomaly detection in distributed systems," Sensors, vol. 24, no. 24, p. 7949, 2024.

[2] X. Wang, K. J. Kim, Y. Wang et al., "DeepEAD: Explainable anomaly detection from system logs," Proceedings of the ICC 2023 - IEEE International Conference on Communications, pp. 771-776, 2023.

[3] P. Wang, X. Zhang, Z. Cao et al., "Loggt: Cross-system log anomaly detection via heterogeneous graph feature and transfer learning," Expert Systems with Applications, vol. 251, p. 124082, 2024.

[4] Y. Fang, Z. Zhao, Y. Xu et al., "Log anomaly detection based on hierarchical graph neural network and label contrastive coding," Computers, Materials, & Continua, vol. 74, no. 2, p. 4099, 2023.

[5] L. Yan, C. Luo and R. Shao, "Discrete log anomaly detection: A novel time-aware graph-based link prediction approach," Information Sciences, vol. 647, p. 119576, 2023.

[6] Y. Ma, "Anomaly detection in microservice environments via conditional multiscale GANs and adaptive temporal autoencoders," 2024.

[7] F. Chen, "AI-Augmented Anomaly Detection via Generative Distribution Modeling and Uncertainty Quantification in Cloud Systems," 2024.

[8] Z. Li, J. Shi and M. Van Leeuwen, "Graph neural networks based log anomaly detection and explanation," Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings, pp. 306-307, 2024.

[9] Y. Xie, H. Zhang and M. A. Babar, "Loggd: Detecting anomalies from system logs with graph neural networks," Proceedings of the 2022 IEEE 22nd International Conference on Software Quality, Reliability and Security (QRS), pp. 299-310, 2022.

[10] Y. Wang, "Semantic-Driven Large Model Scheduling for Distributed Systems via Unified Representation and Policy Generation," 2024.

[11] J. Li, H. He, S. Chen et al., "LogGraph: Log event graph learning aided robust fine-grained anomaly diagnosis," IEEE Transactions on Dependable and Secure Computing, vol. 21, no. 4, pp. 1876-1889, 2023.

[12] Y. Tang, Z. Zhang, K. Zhao et al., "Substructure-Aware Log Anomaly Detection," Proceedings of the VLDB Endowment, vol. 18, no. 2, pp. 213-225, 2024.

[13] B. Barlocker and X. Yan, "Contrastive Representation Learning for Anomaly Detection in Cloud-Based Backend Services," 2021.

[14] Z. Qiu, "A Multi-Scale Deep Learning and Uncertainty Estimation Framework for Comprehensive Anomaly Detection in Cloud Environments," 2023.