

000
001
002
003
004
005
006
007
008
009
010
011
012
013
014
015
016
017
018
019
020
021
022
023
024
025
026
027
028
029
030
031
032
033
034
035
036
037
038
039
040
041
042
043
044
045
046
047
048
049
050
051
052
053

DEBUGGING TABULAR LOG AS DYNAMIC GRAPHS

Anonymous authors

Paper under double-blind review

ABSTRACT

Tabular log abstracts objects and events in the real-world system and reports their updates to reflect the change of the system, where one can detect real-world inconsistencies efficiently by debugging corresponding log entries. However, recent advances in processing text-enriched tabular log data overly depend on large language models (LLMs) and other heavy-load models, thus suffering from limited flexibility and scalability. This paper proposes a new framework, GraphLogDebugger, to debug tabular log based on dynamic graphs. By constructing heterogeneous nodes for objects and events and connecting node-wise edges, the framework recovers the system behind the tabular log as an evolving dynamic graph. With the help of our dynamic graph modeling, a simple dynamic Graph Neural Network (GNN) is representative enough to outperform LLMs in debugging tabular log, which is validated by experimental results on real-world log datasets of computer systems and academic papers.

1 INTRODUCTION

Tabular log data plays a crucial role in representing and tracking the state and evolution of real-world systems. These logs are structured as rows of log entries, each capturing an event involving certain objects and their attributes at a specific time point. Common examples include system logs recording computing services (Zhu et al., 2023a), research logs tracking scientific publication activities (Clement et al., 2019), and interaction logs from multi-agent systems powered by large language models (LLMs) (Zhang et al., 2025b). Debugging of tabular logs is essential: it allows practitioners to detect anomalies in the original systems through efficient inspection of associated log records.

Log anomaly detection (He et al., 2016) has therefore been a long-standing research field in different niche areas, where data distributions are invariant or have little change. Existing frameworks (Du et al., 2017; Meng et al., 2019; Zhang et al., 2019; Pei et al., 2020; Guo et al., 2021; Chen & Tsourakakis, 2022) benefit from manually defined data structures or templates for log parsing which are often tailored to certain domains and thus yield absolute success in specific areas like computer system log or financial event log. However, due to this domain-specific principle, designing a general-purpose log debugger always remains challenging.

Efforts to overcome this challenge have led to two main lines of work, as shown in Figure 1. One stream focuses on graph modeling of the log data (Cheng et al., 2020; Zehra et al., 2021; Pang et al., 2025), where information in tabular log is gathered in a unified data structure: the graph, such as constructing knowledge graphs or text-rich dynamic graphs for computer system log (Sui et al., 2023; Li et al., 2023). Although these methods are both efficient and powerful, many of them lack flexibility: they still customize static graph structures for certain domains. Another stream explores LLM-based solutions, such as LLM prompting (Yu et al., 2023; Qi et al., 2023; Park, 2024) or retrieval-augmented generation (RAG) (Pan et al., 2024; Zhang et al., 2025a; Wang et al., 2025) pipelines. While these methods demonstrate general capabilities in text-based reasoning, thus showing potential of generalization, they often come with significant drawbacks: high computational costs, slow inference, and difficulty scaling to long log streams or resource-constrained settings.

Inspired by the idea to unify multimodal information in dynamic graphs (Feng et al., 2025), we propose **GraphLogDebugger**, a general and efficient framework for debugging tabular logs through dynamic graph modeling. Our core idea is to interpret tabular log entries as the evolving state of a hidden system, which can be reconstructed as a dynamic heterogeneous graph. We treat objects and events as different types of nodes with text embeddings empowered by modern language embed-

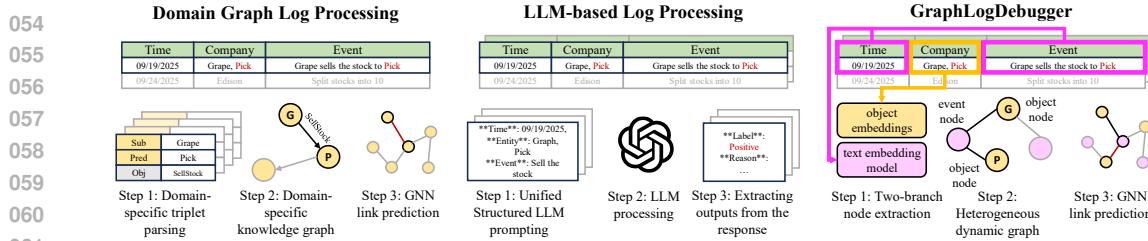


Figure 1: **Comparing GraphLogDebugger with two existing lines of works.** Processing log with domain-specific graphs requires custom text parsing, which lacks flexibility. LLM-based log processing overcomes this shortcoming by the general comprehension skills of LLMs, but suffers from poor efficiency. GraphLogDebugger combines the advantages of graph representation and those of LLMs and balances well generalizability and scalability.

ding models, and use the tabular structure to generate time-stamped connections between them. As new log entries arrive, they incrementally update the dynamic graph, capturing both structural and temporal dependencies. This formulation allows us to apply a lightweight dynamic Graph Neural Network (GNN) to perform online anomaly detection by evaluating the likelihood of new connections. Our approach avoids reliance on heavy LLMs while still capturing rich semantic and relational information in the data. Experimental results on real-world datasets from computer system logs and scientific publication logs validate the effectiveness of our approach. Despite its simplicity, our dynamic GNN framework outperforms LLM-based baselines in both accuracy and efficiency, demonstrating that dynamic graph modeling is a highly expressive yet lightweight alternative. Our contributions can be summarized as follows:

- We introduce a novel view of tabular logs as dynamic heterogeneous graphs, bridging the gap between structured attributes and semantic reasoning, and redefine the framework of online log anomaly detection, where object-event connections in each incoming log are evaluated through link prediction on the evolving graph.
- We propose a lightweight GNN-based debugger that can efficiently and accurately detect anomalies without using LLMs, and validate its performance on real-world datasets with diverse modalities.

2 RELATED WORKS

Tabular Log Processing. Many real-world logs include structured, time-stamped tabular attributes alongside annotated text fields. Examples come from financial prices paired with event series (Tetlock, 2007; Ruiz et al., 2012; Dong et al., 2024), scientific publication metadata (Clement et al., 2019; Kinney et al., 2023), healthcare records (Johnson et al., 2023), computer system logs (Zhu et al., 2023a), and multi-agent system reports (Zhang et al., 2025b). A key challenge in processing tabular logs with machine learning lies in capturing multi-attribute correlations while maintaining comprehension of their semantics (Wu et al., 2025). One common approach integrates main attributes recognized by human priors into structured data (Yang et al., 2018; Zhao & Feng, 2022; Koval et al., 2024), and then subsequently augments the representation by retrieval (Kurisinkel et al., 2024; Xiao et al., 2025). This modeling achieves good performance in domain-specific data, but lacks flexibility and generalizability for adaptation to other fields (Gardner et al., 2024).

An emerging alternative leverages Large Language Models (LLMs) (Brown et al., 2020), which have demonstrated strong generalizability in understanding, predicting, and generating tabular data (Liu et al., 2023; Zhang et al., 2024b; Fang et al., 2024; Wang et al., 2024b). By parsing diverse logs into a unified format with LLMs (Zhong et al., 2024), these models can be applied to downstream tasks that require reasoning capabilities, such as predicting stock prices (Yu et al., 2023), electricity demand (Wang et al., 2024a), and future events (Shi et al., 2023; Ye et al., 2024). However, LLM-based approaches often suffer from high overheads, complex deployment, and limited throughput. There remains a strong need for lighter-weight alternatives with comparable performance.

Dynamic Graphs. Graph Neural Networks (GNNs) have become a foundational paradigm for learning on graph-structured data (Kipf, 2016; Hamilton et al., 2017; Gilmer et al., 2017). Static

108 GNN models have benefited from advances in message passing (Battaglia et al., 2018), architectural depth (Li et al., 2021; Dwivedi et al., 2020), and inductive scalability (Hamilton et al., 2017).
 109 However, many real-world systems are dynamic, motivating models that capture both structural and
 110 temporal dependencies. Early approaches used recurrent layers or time-aware embeddings (Li et al.,
 111 2017; Seo et al., 2018) to extend static GNNs to dynamic settings (Pareja et al., 2020; Sankar et al.,
 112 2020; Kumar et al., 2019). Recent methods have embraced memory modules (Rossi et al., 2020)
 113 and temporal encoding (Xu et al., 2020) for finer-grained modeling of time-stamped interactions.
 114 Building on this trajectory, ROLAND (You et al., 2022) offers a framework that adapts static GNNs
 115 to dynamic graphs via hierarchical state propagation and live-update evaluation, which inspires new
 116 advances in benchmarks (Longa et al., 2023; Huang et al., 2023; Zhang et al., 2024a), architec-
 117 tures (Zhu et al., 2023b), explainability (Chen & Ying, 2023), and robustness (Zhang et al., 2023b).
 118

119 **Log Anomaly Detection.** Log-based anomaly detection has long been a critical task for system
 120 reliability, and early neural approaches typically rely on sequence modeling via LSTMs (Du et al.,
 121 2017), CNNs (Lu et al., 2018), and autoencoders (Zhang et al., 2021; Castillo et al., 2022; Zhang
 122 et al., 2023a). Others incorporate adversarial training (Duan et al., 2021; He et al., 2023), or temporal
 123 networks (Zhang et al., 2019; Yang et al., 2021). More recently, pretrained language models have
 124 been adopted for log anomaly detection, either via fine-tuning (Guo et al., 2021; Lee et al., 2023)
 125 or prompt-based pipelines (Qi et al., 2023; Liu et al., 2024). Retrieval-augmented (No et al., 2024;
 126 Pan et al., 2024; Zhang et al., 2025a) methods have further pushed semantic understanding in LLM-
 127 based methods. As mentioned, while machine learning-based methods are highly domain-specific,
 128 LLM-based methods show some generalizability at a high cost.

129 One potential solution towards general and scalable methods for log debugging is to introduce dy-
 130 namic graphs, where tabular log is considered as an evolving system and maintained in a dynamic
 131 graph. Early exploration makes use of knowledge graphs (Hogan et al., 2021) with domain spe-
 132 cific parsing to generate triplets (Cheng et al., 2020; Zehra et al., 2021; Sui et al., 2023). Recent
 133 advances adopt dynamic graphs with text-rich nodes to represent tabular log (Li et al., 2023; Pang
 134 et al., 2025). Nevertheless, these works are either domain specific or LLM-based, yet not escaping
 135 from the dilemma between generalizability and scalability.

136

3 PRELIMINARIES

137 Tabular log is the data modality used to report the update of real-world systems from the perspective
 138 of states and relations. It can be formally defined by a time series $X = \{x_0, x_1, x_2, \dots, x_{N-1}\}$
 139 annotated by a timestamps sequence $t_0 < t_1 < t_2 < \dots < t_{N-1}$, where each of x_n is a log entry
 140 that contains different attributes x_n^m in the table: $x_n = \{x_n^0, x_n^1, x_n^2, \dots, x_n^{M-1}\}$. Summarizing the
 141 general case of tabular log data in finance (Dong et al., 2024), healthcare (Johnson et al., 2023),
 142 academics (Clement et al., 2019), and other systems (Zhu et al., 2023a), we can separate attributes
 143 in the tabular log into three types:

144

- **Object:** Attributes that represent stand-alone objects in the tabular log, such as companies in the financial news log and cities in the medical record log.
- **Event:** Attributes that describe an event with text, for example, news content in the financial news log and record content in the medical record log. These attributes are usually the center of log entries, where other attributes supplement details and involved objects of the event. Without loss of generality, one log entry only has one Event attribute, because we could merge the text sections of different event attributes into one.
- **Feature:** Attributes that describe features related to the event or objects. For instance, the age is a feature of the patient object in the medical record log. Timestamp t_n is a special type of feature that provides the details about the time of the event.

145

In practice, we find that Objects and Features are mutually convertible. For example, the address of the company could be either an independent object or a feature of the company object in the financial log. Hence, the arrangement of Objects and Features is a hyperparameter that needs pre-definition.

146

While tabular log abstracts the change of real-world systems, it is expected that we could detect inconsistencies of the system from the corresponding tabular log. Based on the above categorization,

162 we could then define three types of anomalies and corresponding anomaly detection tasks in the
 163 tabular log. Give a log entry $x_n = \{x_n^0, x_n^1, x_n^2, \dots, x_n^{M-1}\}$ in the tabular log X :

164

- 165 • **Object anomaly:** Let $\{o_n^0, o_n^1, o_n^2, \dots, o_n^{P-1}\}$ ($P < M$) be the object set. We have label
 166 $y_n = \{y_n^0, y_n^1, y_n^2, \dots, y_n^{P-1}\}$, where $y_n^m = 0$ means that o_n^p is a normal object and $y_n^m = 1$
 167 means that o_n^p is an abnormal object for the log entry x_n .
- 168 • **Event anomaly:** Let s_n be the event. We have a label y_n , where $y_n = 0$ means that s_n is
 169 a normal event and $y_n = 1$ means that s_n is an abnormal event for the log entry x_n .
- 170 • **Feature anomaly:** Let $\{f_n^0, f_n^1, f_n^2, \dots, f_n^{Q-1}\}$ ($Q < M$) be the feature set. We have label
 171 $y_n = \{y_n^0, y_n^1, y_n^2, \dots, y_n^{Q-1}\}$, where $y_n^q = 0$ means that f_n^q is a normal feature and $y_n^q = 1$
 172 means that f_n^q is an abnormal feature for the log entry x_n .

173 Considering the tabular log X as an
 174 online system where new log entries
 175 come dynamically in time order, we
 176 could then define the anomaly de-
 177 tection in tabular log as an online
 178 anomaly detection task:

179 **Definition 3.1 (Online Anomaly
 180 Detection of Tabular Log).** Given
 181 an online system that dynamically
 182 produces log entry x_n , online
 183 anomaly detection for tabular
 184 log predicts its anomaly label y_n
 185 based on historical log entries
 186 $X_n = \{x_0, x_1, x_2, \dots, x_{n-1}\}$

187 Notably, object anomaly and feature
 188 anomaly are isomorphic. Considering
 189 the fact that objects and features are
 190 convertible, the rest of this paper
 191 only studies object anomaly and
 192 event anomaly. [Table 3 summarizes
 193 all used variables.](#)

4 GRAPHLOGDEBUGGER

197 We first integrate tabular log to a
 198 heterogeneous dynamic graph (Section
 199 4.1). Then, we reformulate on-
 200 line anomaly detection of tabular log
 201 as dynamic graph anomaly detec-
 202 tion (Section 4.2). Finally, we apply
 203 a dynamic GNN to debug the tabular
 204 log (Section 4.3).

4.1 INTEGRATING ONLINE TABULAR LOG TO DYNAMIC GRAPHS

207 Objects and events in the same log entry are naturally connected in the tabular log, from which we
 208 could construct graphs. To this end, we first define the graph structure within one log entry. As
 209 shown in Figure 2 (upper section), [we build nodes \$v\$ for both objects and the unique event in the new
 210 log entry. Each event and all its objects are connected by an edge \$e\$.](#) This yields a sub-graph g_n for
 211 each log entry x_n .

212 Figure 2 (middle section) illustrates the composition of a dynamic graph \mathcal{G} that stores the information
 213 of all historical log entries. This dynamic graph gathers all sub-graphs of log entries. We merge
 214 identical object nodes so that these sub-graphs are connected. Note that every event node should be
 215 unique. Every time a new log entry x_n emerges, we construct a sub-graph g_n accordingly and merge
 it into the dynamic graph \mathcal{G} . We denote the snapshot of \mathcal{G} at time point t_n by G_n .

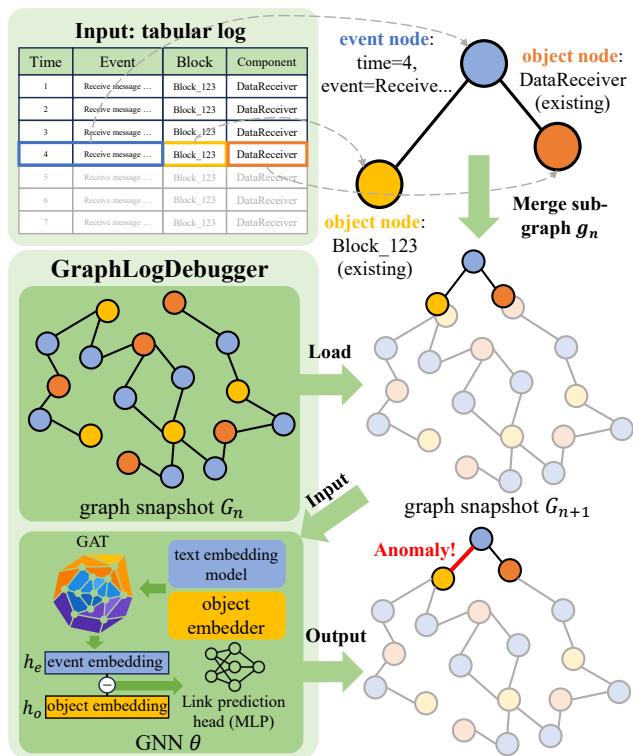


Figure 2: **GraphLogDebugger** framework. The framework checkpoints the GNN θ and the dynamic graph snapshot G_n . When a new log entry emerges, we first extract a sub-graph g_n and use it to update the dynamic graph. Then, we predict the links introduced by g_n in the dynamic graph by GNN θ , whose results indicate the anomaly.

216 **Algorithm 1:** GraphLogDebugger: Online
 217 training for dynamic-graph anomaly detec-
 218 **Input :** Training log $X_{\text{train}} = \{x_{t_0}, \dots, x_{t_{K-1}}\}$;
 219 GNN θ ; text embedding model \mathcal{F} ;
 220 negative sampling ratio ρ ; threshold τ ;
 221 **Output:** Trained parameters θ^* , dynamic graph
 222 snapshot G_{t_K}

1 Initialize $\mathcal{G} : \mathcal{V} = \emptyset, \mathcal{E} = \emptyset$
 2 **for** $k = 0, \dots, K - 1$ **do**
 3 // **Integrate the incoming log entry**
 4 Build sub-graph g_k : creating nodes for
 5 object set \mathcal{V}_k^o and event set \mathcal{V}_k^e in x_{t_k} and
 6 connecting object-event pairs in x_{t_k} with
 7 edges
 8 $\mathcal{G} \leftarrow (\mathcal{V} \cup \mathcal{V}_k^o, \mathcal{E})$
 9 Positive set $\mathcal{E}_k^+ \leftarrow$ object-event links in g_k .
 10 Negatives \mathcal{E}_k^- by drawing $\rho \cdot |\mathcal{E}_k^+|$
 11 non-existent object–event pairs in \mathcal{G} .
 12 // **Embed nodes**
 13 Compute object embeddings h_o with GNN
 14 θ : $h_o = f_\theta(\mathcal{G})$
 15 Compute new event embeddings h_e with text
 16 embedding model \mathcal{F} : $h_e = \mathcal{F}(\mathcal{V}_k^e)$
 17 // **Predict links & Compute the loss**
 18 For each pair $(o, e) \in \mathcal{E}_k^+ \cup \mathcal{E}_k^-$, compute
 19 score $s_{o,e} = \sigma(\text{MLP}(\text{reduce}(h_o, h_e)))$.
 20 Compute balanced BCE loss \mathcal{L}_k on labels (1
 21 for \mathcal{E}_k^+ , 0 for \mathcal{E}_k^-) and update
 22 $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_k$.
 23 // **Updating the dynamic graph**
 24 $\mathcal{G} \leftarrow (\mathcal{V} \cup \mathcal{V}_k^e, \mathcal{E})$
 25 **12** Repeat Step 1-10 for epochs
 26 **13** **return** θ

27 **Algorithm 2:** GraphLogDebugger: Online
 28 evaluation for dynamic-graph anomaly de-
 29 tection

30 **Input :** Test log $X_{\text{test}} = \{x_{t_K}, \dots, x_{t_{N-1}}\}$;
 31 trained GNN θ^* ; last snapshot G_{t_K} ;
 32 text embedding model \mathcal{F} ; threshold τ
 33 **Output:** Per-time link predictions $\{\mathcal{R}_{t_n}\}_{n=K}^{N-1}$;
 34 updated snapshot G_{t_N}

1 Initialize $\mathcal{G} \leftarrow G_{t_K}$ and load θ^*
 2 **for** $n = K, \dots, N - 1$ **do**
 3 // **Integrate the incoming log entry**
 4 Build sub-graph g_n from x_n with object set
 5 \mathcal{V}_n^o , new event set \mathcal{V}_n^e , and observed links
 6 \mathcal{E}_n^+
 7 $\mathcal{G} \leftarrow (\mathcal{V} \cup \mathcal{V}_n^o, \mathcal{E})$
 8 // **Embed nodes**
 9 Compute object embeddings on current
 10 snapshot: $h_o = f_{\theta^*}(\mathcal{G})$
 11 Compute embeddings for new events
 12 (time-aware): $h_e = \mathcal{F}(\mathcal{V}_n^e)$
 13 // **Predict links**
 14 For each $(o, e) \in \mathcal{E}_n^+$, compute
 15 $s_{o,e} = \sigma(\text{MLP}(\text{reduce}(h_o, h_e)))$ and set
 16 $\hat{\ell}_{o,e} = \mathbb{1}[s_{o,e} \geq \tau]$
 17 **Link prediction results:**
 18 $\mathcal{R}_{t_n} \leftarrow \{(o, e, s_{o,e}, \hat{\ell}_{o,e}) \mid (o, e) \in \mathcal{E}_n^+\}$
 19 // **Update the dynamic graph**
 20 **Accepted links**
 21 $\hat{\mathcal{E}}_n^+ = \{(o, e) \in \mathcal{E}_n^+ \mid \hat{\ell}_{o,e} = 1\}$
 22 **Accepted new events**
 23 $\hat{\mathcal{V}}_n^e = \{e \in \mathcal{V}_n^e \mid \exists o : (o, e) \in \hat{\mathcal{E}}_n^+\}$.
 24 $\mathcal{G} \leftarrow (\mathcal{V} \cup \hat{\mathcal{V}}_n^e, \mathcal{E} \cup \hat{\mathcal{E}}_n^+)$
 25 **12** **return** $\{\mathcal{R}_{t_n}\}_{n=K}^{N-1}$ and $G_{t_N} = \mathcal{G}$

4.2 DEBUGGING TABULAR LOG GRAPHS AS DYNAMIC GRAPHS

248 Following the above integration, we transfer the online anomaly detection of tabular log defined in
 249 Section 3 into an anomaly detection problem of dynamic graph \mathcal{G} (Ekle & Eberle, 2024):
 250

251

- **Object anomaly detection:** The object anomaly occurs when the edge e between an object
 252 node and an event node is abnormal in the latest sub-graph g_n . This anomaly could then
 253 be detected by link prediction in the dynamic graph \mathcal{G} (You et al., 2022), where a GNN is
 254 applied to predict the likelihood of e . If the likelihood exceeds a threshold, we consider the
 255 edge as normal. Otherwise, we consider the edge as an anomaly.
- **Event anomaly detection:** The event anomaly occurs when an abnormal event s is placed
 256 in the wrong entry in the latest sub-graph g_n . This means that all edges between this event
 257 are anomalies. We can therefore apply link prediction to all edges in the sub-graph and
 258 threshold the overall predicted likelihoods to determine the anomaly label of the event.

259 In a nutshell, the goal of our anomaly detection in Section 3 is equivalent to predicting the likelihood
 260 of all edges in the latest log entry sub-graph g_n , based on the dynamic graph snapshot G_n that stores
 261 all historical log entries of tabular log X_n . We finally transfer the online anomaly detection of
 262 tabular log into an anomaly detection problem in dynamic graphs, with notations omitted to Table 3:

263 **Definition 4.1 (Online Anomaly Detection of Tabular Log (Dynamic Graph)).** Given the snap-
 264 shot G_n of a dynamic graph $\mathcal{G} = \{G_n\}_{n=0}^{N-1}$ and the new coming sub-graph $g_n = G_{n+1} \setminus G_n$, the
 265 goal is to predict the label y_n of links in g_n .

270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
Table 1: Statistics and details of the four datasets for tabular log debugging.

Dataset	Domain	#Entries	#Objects	Event Attr.	Obj Attr.	Anomaly Type
Arxiv	Sci. Pub.	20,000	17316	title	authors	Event/Object
HDFS	System Log	20,000	2150	Content	Component,EventId,BlockId	Object
Analyst	Finance	20,000	3901	headline	publisher	Event
Landslide	Geology	20,000	8565	description	title, category, trigger, country	Object

4.3 DESIGNING THE GNN FOR DYNAMIC GRAPH ANOMALY DETECTION

Figure 2 (bottom section) demonstrates the basic process of using our GNN to predict link anomaly labels. For details, our GNN θ takes the dynamic graph snapshot G_n and the incoming sub-graph g_n as inputs and predicts the likelihood of all links in g_n . The GNN consists of three parts: the node embedder, the GNN backbone, and the prediction head. First, the node embedder offers heterogeneous embeddings for all objects in G_n and g_n and event nodes in G_n . We exclude new events in g_n because we do not expect the outputs of the model will be interfered with by the graph structure in g_n . We assign a unique learnable embedding for each object, and use a pre-trained text embedding model to embed existing events. We also concatenate a time embedding to the event embedding based on the coming time t_n for event s . Our GNN backbone is adapted from the graph attention network (GAT) (Veličković et al., 2017), where we use two separate MLPs to map objects and events to the same space and apply GAT layers for message passing. The prediction head predicts the link between all object-event pairs in g_n . We first compare object embeddings after GAT layers and event text embeddings by reduction. We then pass the result to an MLP with Sigmoid activation to get the likelihood. We omit more details in the design space of the model in Appendix A.3.

Our GNN is trained under the setting of unsupervised anomaly detection (Pang et al., 2021): We separate the dataset into a training split and a test split by chronological order. In the training stage, all links in g_n are normal, and we provide negative examples for training by randomly sampling object-event pairs that are not connected. We append these fraud links to the ground-truth links to balance the label distribution and use them to train the GNN. After backpropagation, we finally update the dynamic graph snapshot G_n with subgraph g_n . Alg 1 summarizes the training algorithm.

In the test stage, we first resume the GNN as well as the latest dynamic graph snapshot G_n . This achieves the warm start of our debugger system. Then, we construct sub-graphs from the coming log entries and take them as parts of the evolving dynamic graph G that we succeed from the pre-trained GNN. The evaluation process is summarized in Alg 2.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETTINGS

Datasets. Our work provides a general framework of debugging different types of tabular log under the online setting. To validate this point, we span our experiments over datasets covering four different fields: (1) **Arxiv**: Tabular log recording the timestamps (from 2007-2025), the title and the authors of machine learning papers from the Arxiv (Clement et al., 2019) API; (2) **HDFS**: system log of Hadoop Distributed File System designed to run on commodity hardware (Xu et al., 2009; Zhu et al., 2023a), including the event content together with the objects related to the event; (3) **Analyst**: the commentary records on the finance by analysts, including title, author, and other features of posts¹; (4) **Landslide**: event catalog reporting the global landslide². These datasets contain both text-rich attributes and categorical attributes with diversified semantics, thus being challenging to process in one framework efficiently. Table 1 demonstrates the basic statistics of our four datasets.

We limit the maximum length of all tabular logs to 20,000 by slicing the original datasets. This is because LLM-based baseline methods are costly and not scalable, as discussed in the introduction. To ensure randomness, we randomly pick slices with a length of 20,000 from the whole sliced dataset. For datasets with multiple object attributes, we evaluate object anomaly detection, while for those with only one object attribute, we evaluate event anomaly detection, where event and object anomaly detection are equivalent. We summarize the basic setting of our four datasets in Table 1.

¹www.kaggle.com/datasets/miguelaenlle/massive-stock-news-analysis-db-for-nlpbacktests

²<https://catalog.data.gov/dataset/global-landslide-catalog-export>

Table 2: **Our proposed GraphLogDebugger outperforms representative baselines on detection effectiveness and efficiency across diverse methods and datasets.** Higher is better for detection effectiveness; lower GFLOPs and higher Throughput are preferred for efficiency. “*” suggests some baselines always predict non-anomaly cases, leading to a 0 prediction, recall, and F1 score.

Method	Dataset: Arxiv Task: Event Anomaly				Efficiency	
	Acc.	Prec.	Recall	F1	GFLOPs	Throughput (it/s)
MLP	0.570 ± 0.205	0.556 ± 0.189	0.893 ± 0.331	0.676 ± 0.028	11.35	825.0 ± 1429.0
RAG (Llama3-70b, $k=5$)	0.408 ± 0.038	0.426 ± 0.026	0.527 ± 0.029	0.471 ± 0.019	~10 ⁵	0.204 ± 0.009
RAG (GPT-oss-20b, $k=5$)	0.770 ± 0.106	0.771 ± 0.157	0.773 ± 0.014	0.772 ± 0.085	~10 ⁴	0.145 ± 0.018
RAG (Llama3-70b, $k=10$)	0.377 ± 0.014	0.400 ± 0.004	0.493 ± 0.038	0.442 ± 0.014	~10 ⁵	0.204 ± 0.015
RAG (GPT-oss-20b, $k=10$)	0.803 ± 0.090	0.798 ± 0.099	0.813 ± 0.087	0.805 ± 0.087	~10 ⁴	0.149 ± 0.014
GraphLogDebugger (Ours)	0.957 ± 0.040	0.920 ± 0.069	1.000 ± 0.000	0.959 ± 0.037	40.39	627.662 ± 7.623
Method	Dataset: Arxiv Task: Object Anomaly				Efficiency	
	Acc.	Prec.	Recall	F1	GFLOPs	Throughput (it/s)
MLP	0.570 ± 0.000	0.538 ± 0.000	0.990 ± 0.000	0.697 ± 0.000	11.35	552.0 ± 167.0
RAG (Llama3-70b, $k=5$)	0.455 ± 0.033	0.468 ± 0.026	0.667 ± 0.100	0.550 ± 0.052	~10 ⁵	0.208 ± 0.018
RAG (GPT-oss-20b, $k=5$)	0.597 ± 0.019	0.564 ± 0.016	0.850 ± 0.025	0.678 ± 0.004	~10 ⁴	0.039 ± 0.018
RAG (Llama3-70b, $k=10$)	0.463 ± 0.019	0.474 ± 0.011	0.673 ± 0.052	0.556 ± 0.012	~10 ⁵	0.154 ± 0.114
RAG (GPT-oss-20b, $k=10$)	0.598 ± 0.038	0.563 ± 0.023	0.880 ± 0.066	0.687 ± 0.035	~10 ⁴	0.039 ± 0.004
GraphLogDebugger (Ours)	0.685 ± 0.065	0.637 ± 0.082	0.870 ± 0.099	0.734 ± 0.024	40.39	592.073 ± 16.513
Method	Dataset: HDFS Task: Object Anomaly				Efficiency	
	Acc.	Prec.	Recall	F1	GFLOPs	Throughput (it/s)
MLP	0.799 ± 0.053	0.801 ± 0.052	0.989 ± 0.000	0.885 ± 0.032	1.4	479.0 ± 194.0
RAG (Llama3-70b, $k=5$)	0.165 ± 0.022	0 ± 0*	0 ± 0*	0 ± 0*	~10 ⁵	0.162 ± 0.085
RAG (GPT-oss-20b, $k=5$)	0.138 ± 0.029	0 ± 0*	0 ± 0*	0 ± 0*	~10 ⁴	0.183 ± 0.010
RAG (Llama3-70b, $k=10$)	0.173 ± 0.040	0 ± 0*	0 ± 0*	0 ± 0*	~10 ⁵	0.194 ± 0.004
RAG (GPT-oss-20b, $k=10$)	0.138 ± 0.029	0 ± 0*	0 ± 0*	0 ± 0*	~10 ⁴	0.192 ± 0.027
GraphLogDebugger (Ours)	0.989 ± 0.023	1.000 ± 0.000	0.987 ± 0.029	0.993 ± 0.015	5.57	529.999 ± 201.308
Method	Dataset: Analyst Task: Event Anomaly				Efficiency	
	Acc.	Prec.	Recall	F1	GFLOPs	Throughput (it/s)
MLP	0.948 ± 0.019	0.922 ± 0.064	0.980 ± 0.043	0.950 ± 0.016	5.58	1996.0 ± 454.0
RAG (Llama3-70b, $k=5$)	0.408 ± 0.038	0.426 ± 0.026	0.527 ± 0.029	0.471 ± 0.019	~10 ⁵	0.204 ± 0.009
RAG (GPT-oss-20b, $k=5$)	0.770 ± 0.106	0.771 ± 0.157	0.773 ± 0.014	0.772 ± 0.085	~10 ⁴	0.145 ± 0.018
RAG (Llama3-70b, $k=10$)	0.377 ± 0.014	0.400 ± 0.004	0.493 ± 0.038	0.442 ± 0.014	~10 ⁵	0.204 ± 0.015
RAG (GPT-oss-20b, $k=10$)	0.803 ± 0.090	0.798 ± 0.099	0.813 ± 0.087	0.805 ± 0.087	~10 ⁴	0.149 ± 0.014
GraphLogDebugger (Ours)	0.957 ± 0.040	0.921 ± 0.069	1.000 ± 0.000	0.959 ± 0.037	8.97	1037.6 ± 286.2
Method	Dataset: Landslide Task: Object Anomaly				Efficiency	
	Acc.	Prec.	Recall	F1	GFLOPs	Throughput (it/s)
MLP	0.831 ± 0.079	0.842 ± 0.180	0.841 ± 0.149	0.838 ± 0.056	5.58	5391.0 ± 328.0
RAG (Llama3-70b, $k=5$)	0.543 ± 0.074	0.944 ± 0.056	0.095 ± 0.156	0.168 ± 0.267	~10 ⁵	0.355 ± 0.044
RAG (GPT-oss-20b, $k=5$)	0.611 ± 0.068	0.701 ± 0.050	0.389 ± 0.246	0.495 ± 0.195	~10 ⁴	0.155 ± 0.136
RAG (Llama3-70b, $k=10$)	0.551 ± 0.034	0.904 ± 0.204	0.119 ± 0.102	0.208 ± 0.160	~10 ⁵	0.321 ± 0.193
RAG (GPT-oss-20b, $k=10$)	0.623 ± 0.074	0.723 ± 0.081	0.397 ± 0.149	0.511 ± 0.144	~10 ⁴	0.166 ± 0.008
GraphLogDebugger (Ours)	0.840 ± 0.080	0.798 ± 0.117	0.929 ± 0.059	0.858 ± 0.062	19.70	1334.13 ± 108.40

Baselines. Our framework naturally generalizes to tabular log in different domains. Hence, we mainly compare it to baselines which are generally capable of dynamically processing different types of tabular log that contains text-rich and categorical attributes. **MLP** exploits a pretrained text embedding model to embed events and a learnable embedding for objects. A 3-layer MLP is then applied to map the embeddings to anomaly scores. We also compare a series of baselines based on retrieval augmented generation (**RAG**) (Lewis et al., 2020), which is the mainstream method to process general tabular log in the realistic scenario (Akhtar et al., 2025). We deploy RAG based on two advanced open-sourced LLMs, Llama-3-70b (Dubey et al., 2024) and GPT-oss-20b (Agarwal et al., 2025) with the 5 and 10 retrieval entries. The retrieval database is built on the whole training

378 split and the seen log entries during the online evaluation. We construct specified prompts for
 379 different datasets and omitted the description to Appendix A. Both MLP and all RAG baselines
 380 use all-MiniLM-L6-v2 ³ (Reimers & Gurevych, 2019) as the text embedding model. **We do not**
 381 **compare to baselines on log anomaly detection because all these methods are either template-based**
 382 **or domain-specific, which cannot be applied to datasets other than computer system log.**

383 **Task.** Following the setting of unsupervised anomaly detection (Liu et al., 2021; Schmidl et al.,
 384 2022), our basic task is to output an anomaly score for each log entry x_n at timestamp t_n , where
 385 higher scores denote more outlyingness (Han et al., 2022). In our task, we use 1 to denote anomalies
 386 and 0 to denote normal examples in the ground-truth. We use the first 90% split of the dataset
 387 for training, where both the log entries and their anomaly labels are available to access for methods.
 388 Methods train the model on this training split or use it for the retrieval database. For the rest 10%, we
 389 use it as the test split in our online evaluation, where methods can make use of the seen log entries but
 390 their anomaly labels are not accessible. We study two types of anomalies in our experiments: object
 391 anomalies and event anomalies. Following the definition in Section 3, we inject object anomalies
 392 by swapping an object in the log entry with another existing object. To ensure that historical data
 393 contains useful information, we only perturb existing objects in the history. Event anomalies are
 394 generated similarly by swapping events. The anomaly rate is set to be 0.05.

395 **Evaluation.** We calculate the metrics for information retrieval: accuracy, precision, recall, and f1
 396 score for the dataset. We also evaluate the efficiency of different methods by GFlops and through-
 397 puts. During evaluation, we notice that LLM-based baselines tend to be very slow in processing
 398 speed. Hence, we include all anomaly log entries and 50 random normal entries in a subset and
 399 run RAG only on this subset. For other baselines and our methods, we obtain the prediction result
 400 for the full test split but only compute the metric on the above subset for fair comparison. We run
 401 experiments three times and post the average value of metrics with error bars.

402 **GraphLogDebugger.** The GNN architecture in GraphLogDebugger is a 3-layer GAT backbone
 403 with a two-branch node embedder and an MLP prediction head. The node embedder uses 512-d
 404 embeddings for objects and the text embedding of all-MiniLM-L6-v2 (Reimers & Gurevych, 2019)
 405 for events, with an MLP to map them into the same space. The embedding size of GAT and the
 406 prediction head is also 512. We train the GNN for 10 epochs under the learning rate 0.0001 on
 407 Adam and the negative ratio 10 on the training split. Following You et al. (2022), we set the batch
 408 size as 1 and use a window length of 100 to accelerate the processing.

409 5.2 MAIN RESULTS

410 Table 2 shows that **GraphLogDebugger** consistently outperforms both MLP and RAG-based base-
 411 lines across all tabular log datasets on five tasks, in terms of detection performance and efficiency.

412 **Effectiveness:** GraphLogDebugger achieves the highest F1 scores across all tasks, outperforming
 413 RAG baselines—especially in structurally complex domains like HDFS, where RAG methods fail
 414 to detect meaningful anomalies ($F1 = 0.0$). Specifically, RAG baselines are completely fooled by
 415 the anomaly pattern that their predicted labels depend on whether there is an existing record with
 416 the same format in the retrieved examples, which does not contribute to a reasonable prediction.
 417 Two tasks on the Arxiv dataset are the most difficult, where GraphLogDebugger still beats baselines
 418 with a higher precision in not abusing anomaly prediction. Even in semantically rich settings such
 419 as Analyst and LandSlide, where RAG baselines are expected to excel, our model surpasses them.

420 **Efficiency:** RAG approaches exhibit extremely low throughput (typically below 0.3 iterations per
 421 second) due to the computational overhead of large language models. In contrast, GraphLogDe-
 422 bugger achieves throughput of at least 500 per second, with significantly lower GFLOPs, enabling
 423 real-time anomaly detection in high-throughput environments.

424 5.3 CASE STUDY: WHERE DOES RAG FAIL?

425 It is natural that GraphLogDebugger yields advantages in
 426 efficiency compared to the RAG-baseline, for the latter
 427 relies on LLMs with billions of parameters. However, the

Method	Correlation
RAG	-0.1087
GraphLogDebugger	0.1561

³<https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2>

432 leading performance of GraphLogDebugger in detection needs further explanation, while RAG en-
 433 joys the general comprehension and reasoning ability of modern LLMs. To this end, we study cases
 434 from event anomaly detection of the Arxiv dataset. We choose this task because the degree of event
 435 nodes can directly reflect the local graph density of the node-of-interest. We calculate the correlation
 436 between event node degrees and the accuracy of GraphLogDebugger and that of RAG(GPT-oss-
 437 20b, $k=10$). The result in the table shows that **the accuracy of GraphLogDebugger is positively**
 438 **correlated with the node degree, while the accuracy of RAG is negatively correlated with the**
 439 **node degree.** This indicates that GraphLogDebugger outperforms RAG on event nodes with rich
 440 connections with objects, where semantics of these objects are necessary to detect the anomaly.

441

442 **Case 1: Label=negative, RAG=positive, GraphLogDebugger=negative**

443 **Title:** "SymbioSim: Human-in-the-loop Simulation Platform for Bidirectional Continuing
 444 Learning in Human-Robot Interaction"

445 **Authors:** "Haoran Chen", "Yiming Ren", "Xinran Li", "Ning Ding", "Ziyi Wang", "Yuhan
 446 Chen", "Zhiyang Dou", "Yuexin Ma", "Changhe Tu" (9 objects)

447 **Reason (RAG):** The author team composition, research domain mismatch, and unclear collabora-
 448 tion patterns raise suspicions about the coherence of the record.

449 **Case 2: Label=positive, RAG=negative, GraphLogDebugger=positive**

450 **Title:** "VERA: Explainable Video Anomaly Detection via Verbalized Learning of Vision-
 451 Language Models"

452 **Authors:** "Shubham Gupta", "Zichao Li", "Tianyi Chen", "Cem Subakan", "Siva Reddy", "Per-
 453 ouz Taslakian", "Valentina Zantedeschi" (7 objects)

454 **Reason (RAG):** The record seems coherent, with individual authors' expertise areas aligning
 455 with the paper's topic, although the team size is slightly larger than expected.

456 **Case 3: Label=positive, RAG=positive, GraphLogDebugger=negative**

457 **Title:** "Transformer⁻¹: Input-Adaptive Computation for Resource-Constrained Deployment"

458 **Authors:** "Yitong Yin" (1 objects)

459 **Reason (RAG):** The record consists of a single author, which is consistent with similar papers
 460 in the same research domain.

461

462 We further raise three cases above to investigate when and how GraphLogDebugger and RAG fail. In
 463 Case 1, RAG predicts the normal example as abnormal because the limited retrieved examples do not
 464 provide enough evidence to prove the coherence of the author team. By contrast, GraphLogDebugger
 465 validates overall team consistency by checking the research background of every author, which
 466 correctly predicts the negative label. Case 2 is complementary to Case 1, where GraphLogDebugger
 467 is able to scan the research interest of every author and detect the anomaly accurately. However,
 468 when the connected objects are few, such as in Case 3, GraphLogDebugger may not have enough
 469 references based on the graph to make a correct judgment. In similar cases, RAG could then outper-
 470 form GraphLogDebugger to recognize patterns in the number of authors in the same domain.

471

472 These cases provide insights on how graphs can benefit retrieval augmented generation. When the
 473 key entry has dense connections with other entries, traditional retrieval based on similarity can-
 474 not efficiently include enough entries to enhance the generation quality. With the help of modern
 475 embedding models, graphs can be introduced to gather information in these multi-entry scenarios.

476

6 CONCLUSION

477

478 We propose a general framework to cover online debugging for heterogeneous tabular logs. By
 479 modeling online log debugging as anomaly detection of dynamic graphs, our framework integrates
 480 different types of log data into a unified modality by text embedding models, where a dynamic GNN
 481 debugs the log through link prediction. Our framework shows good performance in four different
 482 datasets while maintaining high efficiency compared to the mainstream RAG-based method.

483

484 **Limitation.** Our work explores combining dynamic GNNs and text embedding models to process
 485 log data under the online setting, which indicates the potential to accelerate the online process of
 486 data streams by a graph-based method. Nevertheless, our experiments mainly show this potential in
 487 the bug detection setting. We leave the exploration of online bug correction to future work.

486 ETHICS STATEMENT
487488 Our work focuses on detecting inconsistencies in general tabular log data, which enhances the
489 progress of automated log data processing in real-world scenarios. While automation of log pro-
490 cessing may raise issues concerning hallucination or fraud reporting, our work does not explicitly
491 introduce new risks compared to existing research.
492493 REPRODUCIBILITY STATEMENT
494495 All implementation details of our method and baselines are given in Section 5 and Appendix A. We
496 will release at the time of publication.
497498 REFERENCES
499500 Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K
501 Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv*
502 *preprint arXiv:2508.10925*, 2025.
503504 Siraj Akhtar, Saad Khan, and Simon Parkinson. Llm-based event log analysis techniques: A survey.
505 *arXiv preprint arXiv:2502.00677*, 2025.
506507 Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi,
508 Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al.
509 Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*,
2018.
510511 Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal,
512 Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are
513 few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
514515 Martina Castillo, Antonio Pecchia, and Ugo Villano. Autolog: Anomaly detection by deep autoen-
516 coding of system logs. *Expert Systems with Applications*, 191, 2022.
517518 Jialin Chen and Rex Ying. Tempme: Towards the explainability of temporal graph neural networks
via motif discovery. *Advances in Neural Information Processing Systems*, 36:29005–29028, 2023.
519520 Tianyi Chen and Charalampos Tsourakakis. Antibenford subgraphs: Unsupervised anomaly detec-
521 tion in financial networks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge
Discovery and Data Mining*, pp. 2762–2770, 2022.
522523 Dawei Cheng, Fangzhou Yang, Xiaoyang Wang, Ying Zhang, and Liqing Zhang. Knowledge graph-
524 based event embedding framework for financial quantitative investments. In *Proceedings of the
43rd International ACM SIGIR Conference on Research and Development in Information Re-
trieval*, pp. 2221–2230, 2020.
525526 Colin B Clement, Matthew Bierbaum, Kevin P O’Keeffe, and Alexander A Alemi. On the use of
527 arxiv as a dataset. *arXiv preprint arXiv:1905.00075*, 2019.
528529 Zihan Dong, Xinyu Fan, and Zhiyuan Peng. Fnspid: A comprehensive financial news dataset in time
530 series, 2024.
531532 Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. Deeplog: Anomaly detection and diagnosis
533 from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on
computer and communications security*, pp. 1285–1298, 2017.
534535 Xiaoyu Duan, Shi Ying, Wanli Yuan, Hailong Cheng, and Xiang Yin. A generative adversarial
536 networks for log anomaly detection. *Computer Systems Science & Engineering*, 37(1), 2021.
537538 Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha
539 Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models.
arXiv e-prints, pp. arXiv–2407, 2024.
540

540 Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson.
 541 Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

542

543 Ocheme Anthony Ekle and William Eberle. Anomaly detection in dynamic graphs: A comprehensive survey. *ACM Transactions on Knowledge Discovery from Data*, 18(8):1–44, 2024.

544

545 Xi Fang, Weijie Xu, Fiona Anting Tan, Jiani Zhang, Ziqing Hu, Yanjun Qi, Scott Nickleach, Diego
 546 Socolinsky, Srinivasan Sengamedu, and Christos Faloutsos. Large language models (llms) on tab-
 547 ular data: Prediction, generation, and understanding—a survey. *arXiv preprint arXiv:2402.17944*,
 548 2024.

549

550 Tao Feng, Yixin Wu, Guanyu Lin, and Jiaxuan You. Graph world model. *arXiv preprint*
 551 *arXiv:2507.10539*, 2025.

552

553 Josh Gardner, Juan C Perdomo, and Ludwig Schmidt. Large scale transfer learning for tabular data
 554 via language modeling. *Advances in Neural Information Processing Systems*, 37:45155–45205,
 555 2024.

556

557 Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural
 558 message passing for quantum chemistry. In *International conference on machine learning*, pp.
 559 1263–1272. Pmlr, 2017.

560

561 Haixuan Guo, Shuhan Yuan, and Xintao Wu. Logbert: Log anomaly detection via bert. In *2021*
 562 *international joint conference on neural networks (IJCNN)*, pp. 1–8. IEEE, 2021.

563

564 Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs.
 565 *Advances in neural information processing systems*, 30, 2017.

566

567 Songqiao Han, Xiyang Hu, Hailiang Huang, Minqi Jiang, and Yue Zhao. Adbench: Anomaly de-
 568 tection benchmark. *Advances in neural information processing systems*, 35:32142–32159, 2022.

569

570 Shilin He, Jieming Zhu, Pinjia He, and Michael R Lyu. Experience report: System log analysis for
 571 anomaly detection. In *2016 IEEE 27th international symposium on software reliability engineer-
 572 ing (ISSRE)*, pp. 207–218. IEEE, 2016.

573

574 Zhangyue He, Yanni Tang, Kaiqi Zhao, Jiamou Liu, and Wu Chen. Graph-based log anomaly detec-
 575 tion via adversarial training. In *International Symposium on Dependable Software Engineering:
 576 Theories, Tools, and Applications*, pp. 55–71. Springer, 2023.

577

578 Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d’Amato, Gerard De Melo, Claudio Gutier-
 579 rez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, et al.
 580 Knowledge graphs. *ACM Computing Surveys (Csur)*, 54(4):1–37, 2021.

581

582 Shenyang Huang, Farimah Poursafaei, Jacob Danovitch, Matthias Fey, Weihua Hu, Emanuele Rossi,
 583 Jure Leskovec, Michael Bronstein, Guillaume Rabreau, and Reihaneh Rabbany. Temporal
 584 graph benchmark for machine learning on temporal graphs. *Advances in Neural Information
 585 Processing Systems*, 36:2056–2073, 2023.

586

587 Alistair EW Johnson, Lucas Bulgarelli, Lu Shen, Alvin Gayles, Ayad Shammout, Steven Horng,
 588 Tom J Pollard, Sicheng Hao, Benjamin Moody, Brian Gow, et al. Mimic-iv, a freely accessible
 589 electronic health record dataset. *Scientific data*, 10(1):1, 2023.

590

591 Rodney Kinney, Chloe Anastasiades, Russell Author, Iz Beltagy, Jonathan Bragg, Alexandra Bu-
 592 raczynski, Isabel Cachola, Stefan Candra, Yoganand Chandrasekhar, Arman Cohan, et al. The
 593 semantic scholar open data platform. *arXiv preprint arXiv:2301.10140*, 2023.

594

595 TN Kipf. Semi-supervised classification with graph convolutional networks. *arXiv preprint*
 596 *arXiv:1609.02907*, 2016.

597

598 Ross Koval, Nicholas Andrews, and Xifeng Yan. Financial forecasting from textual and tabular
 599 time series. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp.
 600 8289–8300, 2024.

594 Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in tem-
 595 poral interaction networks. In *Proceedings of the 25th ACM SIGKDD international conference*
 596 *on knowledge discovery & data mining*, pp. 1269–1278, 2019.

597

598 Litton Jose Kurisinkel, Pruthwik Mishra, and Yue Zhang. Text2timeseries: Enhancing financial
 599 forecasting through time series prediction updates with event-driven insights from large language
 600 models. *arXiv preprint arXiv:2407.03689*, 2024.

601 Yukyung Lee, Jina Kim, and Pilsung Kang. Lanobert: System log anomaly detection based on bert
 602 masked language model. *Applied Soft Computing*, 146:110689, 2023.

603

604 Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal,
 605 Heinrich Kütller, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented gener-
 606 ation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:
 607 9459–9474, 2020.

608

609 Guohao Li, Matthias Müller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as
 610 cnns? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

611

612 Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural net-
 613 work: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*, 2017.

614

615 Yufei Li, Yanchi Liu, Haoyu Wang, Zhengzhang Chen, Wei Cheng, Yuncong Chen, Wenchao Yu,
 616 Haifeng Chen, and Cong Liu. Glad: Content-aware dynamic graphs for log anomaly detection.
 617 In *2023 IEEE International Conference on Knowledge Graph (ICKG)*, pp. 9–18. IEEE, 2023.

618

619 Tianyang Liu, Fei Wang, and Muhao Chen. Rethinking tabular data understanding with large lan-
 620 guage models. *arXiv preprint arXiv:2312.16702*, 2023.

621

622 Yilun Liu, Shimin Tao, Weibin Meng, Feiyu Yao, Xiaofeng Zhao, and Hao Yang. Logprompt:
 623 Prompt engineering towards zero-shot and interpretable log analysis. In *Proceedings of the 2024*
624 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings,
 625 pp. 364–365, 2024.

626

627 Yixin Liu, Zhao Li, Shirui Pan, Chen Gong, Chuan Zhou, and George Karypis. Anomaly detection
 628 on attributed networks via contrastive self-supervised learning. *IEEE transactions on neural*
 629 *networks and learning systems*, 33(6):2378–2392, 2021.

630

631 Alessandro Longa, Valerio Lachi, Giovanni Santin, Monica Bianchini, Bruno Lepri, et al. Graph
 632 neural networks for temporal graphs: State of the art, open challenges, and opportunities. *arXiv*
 633 *preprint arXiv:2305.12472*, 2023.

634

635 Siyang Lu, Xiang Wei, Yandong Li, and Liqiang Wang. Detecting anomaly in big data system logs
 636 using convolutional neural network. In *2018 IEEE 16th Intl Conf on Dependable, Autonomic and*
637 Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big
638 Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/-
639 DataCom/CyberSciTech), pp. 151–158. IEEE, 2018.

640

641 Weibin Meng, Ying Liu, Yichen Zhu, Shenglin Zhang, Dan Pei, Yuqing Liu, Yihao Chen, Ruizhi
 642 Zhang, Shimin Tao, Pei Sun, et al. Loganomaly: Unsupervised detection of sequential and quan-
 643 titative anomalies in unstructured logs. In *IJCAI*, volume 19, pp. 4739–4745, 2019.

644

645 Geon No et al. Training-free retrieval-based log anomaly detection with pre-trained language mod-
 646 els. *Engineering Applications of Artificial Intelligence*, 133, 2024.

647

648 Jonathan Pan, Wong Swee Liang, and Yuan Yidi. Raglog: Log anomaly detection using retrieval
 649 augmented generation. In *2024 IEEE World Forum on Public Safety Technology (WFPST)*, pp.
 650 169–174. IEEE, 2024.

651

652 Guansong Pang, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel. Deep learning for
 653 anomaly detection: A review. *ACM computing surveys (CSUR)*, 54(2):1–38, 2021.

648 Yunhe Pang, Bo Chen, Fanjin Zhang, Yanghui Rao, Evgeny Kharlamov, and Jie Tang. Guard:
 649 Effective anomaly detection through a text-rich and graph-informed language model. In *Proceed-
 650 ings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining* V. 2, pp.
 651 2222–2233, 2025.

652 Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kaneza-
 653 shi, Tim Kaler, Tao Schardl, and Charles Leiserson. Evolvegcn: Evolving graph convolutional
 654 networks for dynamic graphs. In *Proceedings of the AAAI conference on artificial intelligence*,
 655 volume 34, pp. 5363–5370, 2020.

656 Taejin Park. Enhancing anomaly detection in financial markets with an llm-based multi-agent frame-
 657 work. *arXiv preprint arXiv:2403.19735*, 2024.

658 Yulong Pei, Fang Lyu, Werner Van Ipenburg, and Mykola Pechenizkiy. Subgraph anomaly detection
 659 in financial transaction networks. In *Proceedings of the First ACM International Conference on
 660 AI in Finance*, pp. 1–8, 2020.

661 Jiaxing Qi, Shaohan Huang, Zhongzhi Luan, Shu Yang, Carol Fung, Hailong Yang, Depei Qian, Jing
 662 Shang, Zhiwen Xiao, and Zhihui Wu. Loggpt: Exploring chatgpt for log-based anomaly detection.
 663 In *2023 IEEE International Conference on High Performance Computing & Communications,
 664 Data Science & Systems, Smart City & Dependability in Sensor, Cloud & Big Data Systems &
 665 Application (HPCC/DSS/SmartCity/DependSys)*, pp. 273–280. IEEE, 2023.

666 Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-
 667 networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language
 668 Processing*. Association for Computational Linguistics, 11 2019. URL <https://arxiv.org/abs/1908.10084>.

669 Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael
 670 Bronstein. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint
 671 arXiv:2006.10637*, 2020.

672 Eduardo J Ruiz, Vagelis Hristidis, Carlos Castillo, Aristides Gionis, and Alejandro Jaimes. Corre-
 673 lating financial time series with micro-blogging activity. In *Proceedings of the fifth ACM interna-
 674 tional conference on Web search and data mining*, pp. 513–522, 2012.

675 Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. Dysat: Deep neural rep-
 676 resentation learning on dynamic graphs via self-attention networks. In *Proceedings of the 13th
 677 international conference on web search and data mining*, pp. 519–527, 2020.

678 Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. Anomaly detection in time series: a
 679 comprehensive evaluation. *Proceedings of the VLDB Endowment*, 15(9):1779–1797, 2022.

680 Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. Structured sequence
 681 modeling with graph convolutional recurrent networks. In *International conference on neural
 682 information processing*, pp. 362–373. Springer, 2018.

683 Xiaoming Shi, Siqiao Xue, Kangrui Wang, Fan Zhou, James Zhang, Jun Zhou, Chenhao Tan, and
 684 Hongyuan Mei. Language models can improve event prediction by few-shot abductive reasoning.
 685 *Advances in Neural Information Processing Systems*, 36:29532–29557, 2023.

686 Yicheng Sui, Yuzhe Zhang, Jianjun Sun, Ting Xu, Shenglin Zhang, Zhengdan Li, Yongqian Sun,
 687 Fangrui Guo, Junyu Shen, Yuzhi Zhang, et al. Logkg: Log failure diagnosis through knowledge
 688 graph. *IEEE Transactions on Services Computing*, 16(5):3493–3507, 2023.

689 Paul C Tetlock. Giving content to investor sentiment: The role of media in the stock market. *The
 690 Journal of finance*, 62(3):1139–1168, 2007.

691 Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua
 692 Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

693 Jingru Wang, Wen Ding, and Xiaotong Zhu. Financial analysis: Intelligent financial data analysis
 694 system based on llm-rag. *arXiv preprint arXiv:2504.06279*, 2025.

702 Xinlei Wang, Maike Feng, Jing Qiu, Jinjin Gu, and Junhua Zhao. From news to forecast: Inte-
 703 grating event analysis in llm-based time series forecasting with reflection. *Advances in Neural*
 704 *Information Processing Systems*, 37:58118–58153, 2024a.

705 Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang,
 706 Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, et al. Chain-of-table: Evolving
 707 tables in the reasoning chain for table understanding. *arXiv preprint arXiv:2401.04398*, 2024b.

708 Xiaofeng Wu, Alan Ritter, and Wei Xu. Tabular data understanding with llms: A survey of recent
 709 advances and challenges. *arXiv preprint arXiv:2508.00217*, 2025.

710 Mengxi Xiao, Zihao Jiang, Lingfei Qian, Zhengyu Chen, Yueru He, Yijing Xu, Yuecheng Jiang,
 711 Dong Li, Ruey-Ling Weng, Min Peng, et al. Enhancing financial time-series forecasting with
 712 retrieval-augmented large language models. *arXiv preprint arXiv:2503.67890*, 2025.

713 Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Inductive represen-
 714 tation learning on temporal graphs. *arXiv preprint arXiv:2002.07962*, 2020.

715 Wei Xu, Ling Huang, Armando Fox, David Patterson, and Michael I Jordan. Detecting large-scale
 716 system problems by mining console logs. In *Proceedings of the ACM SIGOPS 22nd symposium*
 717 *on Operating systems principles*, pp. 117–132, 2009.

718 Hang Yang, Yubo Chen, Kang Liu, Yang Xiao, and Jun Zhao. Dcfee: A document-level chinese
 719 financial event extraction system based on automatically labeled training data. In *Proceedings of*
 720 *ACL 2018, System Demonstrations*, pp. 50–55, 2018.

721 Lin Yang, Junjie Chen, Zan Wang, Weijing Wang, Jiajun Jiang, Xuyuan Dong, and Wenbin
 722 Zhang. Semi-supervised log-based anomaly detection via probabilistic label estimation. In
 723 *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pp. 1448–1460.
 724 IEEE, 2021.

725 Chenchen Ye, Ziniu Hu, Yihe Deng, Zijie Huang, Mingyu Derek Ma, Yanqiao Zhu, and Wei Wang.
 726 Mirai: Evaluating llm agents for event forecasting. *arXiv preprint arXiv:2407.01231*, 2024.

727 Jiaxuan You, Tianyu Du, and Jure Leskovec. Roland: graph learning framework for dynamic graphs.
 728 In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*,
 729 pp. 2358–2366, 2022.

730 Xinli Yu, Zheng Chen, Yuan Ling, Shujing Dong, Zongyi Liu, and Yanbin Lu. Temporal data meets
 731 llm-explainable financial time series forecasting. *arXiv preprint arXiv:2306.11025*, 2023.

732 Samreen Zehra, Syed Farhan Mohsin Mohsin, Shaukat Wasi, Syed Imran Jami, Muhammad Shoaib
 733 Siddiqui, and Muhammad Khaliq-Ur-Rahman Raazi Syed. Financial knowledge graph based
 734 financial report query system. *IEEE Access*, 9:69766–69782, 2021.

735 Jiasheng Zhang, Jialin Chen, Menglin Yang, Aosong Feng, Shuang Liang, Jie Shao, and Rex Ying.
 736 Dtgb: A comprehensive benchmark for dynamic text-attributed graphs. *Advances in Neural In-*
 737 *formation Processing Systems*, 37:91405–91429, 2024a.

738 Lingzhe Zhang, Tong Jia, Mengxi Jia, Yifan Wu, Hongyi Liu, and Ying Li. Xraglog: A resource-
 739 efficient and context-aware log-based anomaly detection method using retrieval-augmented gen-
 740 eration. In *AAAI 2025 Workshop on Preventing and Detecting LLM Misinformation (PDLM)*,
 741 2025a.

742 Linming Zhang, Wenzhong Li, Zhijie Zhang, Qingning Lu, Ce Hou, Peng Hu, Tong Gui, and Sanglu
 743 Lu. Logattn: Unsupervised log anomaly detection with an autoencoder based attention mecha-
 744 nism. In *International conference on knowledge science, engineering and management*, pp. 222–
 745 235. Springer, 2021.

746 Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li,
 747 Chi Wang, Huazheng Wang, Yiran Chen, et al. Which agent causes task failures and when?
 748 on automated failure attribution of llm multi-agent systems. *arXiv preprint arXiv:2505.00212*,
 749 2025b.

756 Xiaokang Zhang, Sijia Luo, Bohan Zhang, Zeyao Ma, Jing Zhang, Yang Li, Guanlin Li, Zijun Yao,
 757 Kangli Xu, Jinchang Zhou, et al. Tablellm: Enabling tabular data manipulation by llms in real
 758 office usage scenarios. *arXiv preprint arXiv:2403.19318*, 2024b.

759

760 Xinye Zhang, Xiaoli Chai, Minghua Yu, and Ding Qiu. Anomaly detection model for log based on
 761 lstm network and variational autoencoder. In *2023 4th International Conference on Information
 762 Science, Parallel and Distributed Systems (ISPDS)*, pp. 239–244. IEEE, 2023a.

763

764 Xu Zhang, Yong Xu, Qingwei Lin, Bo Qiao, Hongyu Zhang, Yingnong Dang, Chunyu Xie, Xin-
 765 sheng Yang, Qian Cheng, Ze Li, et al. Robust log-based anomaly detection on unstable log data.
 766 In *Proceedings of the 2019 27th ACM joint meeting on European software engineering conference
 767 and symposium on the foundations of software engineering*, pp. 807–817, 2019.

768

769 Zeyang Zhang, Xin Wang, Ziwei Zhang, Zhou Qin, Weigao Wen, Hui Xue, Haoyang Li, and Wenwu
 770 Zhu. Spectral invariant learning for dynamic graphs under distribution shifts. *Advances in Neural
 771 Information Processing Systems*, 36:6619–6633, 2023b.

772

773 Qihang Zhao and Xiaodong Feng. Utilizing citation network structure to predict paper citation
 774 counts: A deep learning approach. *Journal of Informetrics*, 16(1):101235, 2022.

775

776 Aoxiao Zhong, Dengyao Mo, Guiyang Liu, Jinbu Liu, Qingda Lu, Qi Zhou, Jiesheng Wu,
 777 Quanzheng Li, and Qingsong Wen. Logparser-llm: Advancing efficient log parsing with large
 778 language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discov-
 779 ery and Data Mining*, pp. 4559–4570, 2024.

780

781 Jieming Zhu, Shilin He, Pinjia He, Jinyang Liu, and Michael R Lyu. Loghub: A large collection of
 782 system log datasets for ai-driven log analytics. In *2023 IEEE 34th International Symposium on
 783 Software Reliability Engineering (ISSRE)*, pp. 355–366. IEEE, 2023a.

784

785 Yifan Zhu, Fangpeng Cong, Dan Zhang, Wenwen Gong, Qika Lin, Wenzheng Feng, Yuxiao Dong,
 786 and Jie Tang. Wingnn: Dynamic graph neural networks with random gradient aggregation win-
 787 dow. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data
 788 mining*, pp. 3650–3662, 2023b.

789

A APPENDIX

A.1 USE OF LLMs

790 We use ChatGPT to polish our introduction (Section 1) and generate the notation table (Table 3),
 791 both of which have been checked manually. We also use ChatGPT to retrieve related works in the
 792 tabular log processing part by searching machine-learning based log processing methods.
 793

A.2 NOTATION

794 Table 3: Notation

795 Symbol	796 Type	797 Meaning
798 Tabular-log basics (Sec. 3)		
800 $X = \{x_0, \dots, x_{N-1}\}$	801 sequence	802 Time-ordered tabular log (entries).
803 x_n	804 entry	805 The n -th log entry.
806 $t_0 < \dots < t_{N-1}$	807 timestamps	808 Arrival times of entries.
809 x_n^m	810 attribute value	811 The m -th attribute in entry x_n .
812 M	813 integer	814 Number of attributes per entry.
815 $\{o_n^0, \dots, o_n^{P-1}\}$	816 set	817 Object attributes extracted from x_n .
818 P	819 integer	820 Number of object attributes in x_n ($P < M$).
821 s_n	822 text / node	823 Event attribute (one per entry; possibly text).
824 $\{f_n^0, \dots, f_n^{Q-1}\}$	825 set	826 Feature attributes extracted from x_n .

827 *Continued on next page*

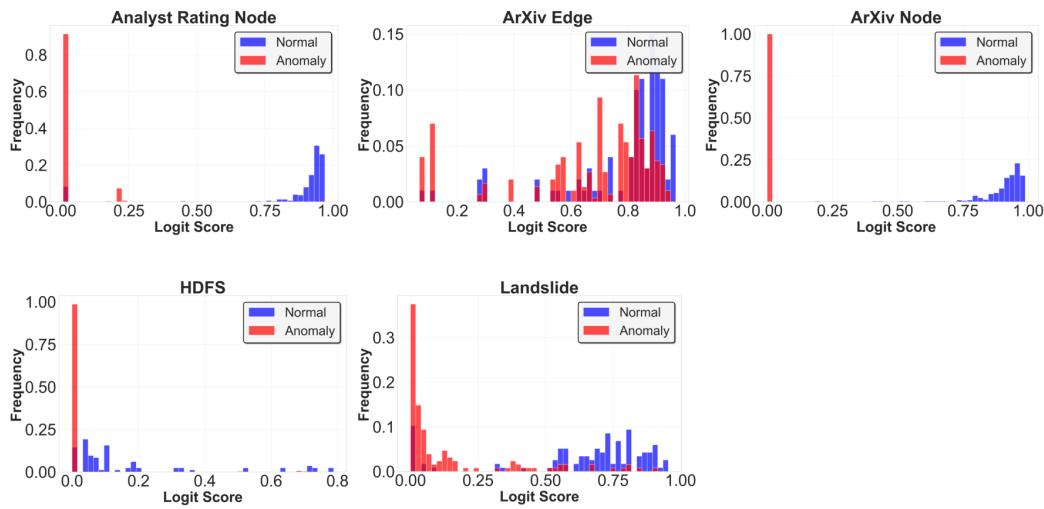
810	Symbol	Type	Meaning
811	Q	integer	Number of feature attributes in x_n ($Q < M$).
812	y_n	label	Event anomaly label for x_n (0 normal, 1 abnormal).
813	y_n^p	label	Object anomaly label for object o_n^p (0/1).
814	y_n^q	label	Feature anomaly label for feature f_n^q (0/1).
815	Graphs and dynamics (Sec. 4.1–4.2)		
816	\mathcal{G}	dynamic graph	Evolving heterogeneous graph over time.
817	G_n	snapshot	Graph snapshot at time t_n (before merging g_n).
818	g_n	subgraph	Subgraph constructed from new entry x_n .
819	$G_{n+1} \setminus G_n$	graph diff	Increment between consecutive snapshots; here equal to g_n .
820	\mathcal{V}, \mathcal{E}	sets	Node and edge sets of the current graph.
821	v, e	node, edge	A node or an edge (generic).
822	\mathcal{V}_n^o	node set	Object nodes appearing in x_n .
823	\mathcal{V}_n^e	node set	New event nodes introduced by x_n (events are unique).
824	\mathcal{E}_k^+	edge set	Positive (observed) object–event links in g_k .
825	\mathcal{E}_k^-	edge set	Negative samples (non-existent object–event pairs).
826	$\hat{\mathcal{E}}_n^+$	edge set	Accepted/predicted-positive links at t_n .
827	$\hat{\mathcal{V}}_n^e$	node set	Accepted new events incident to $\hat{\mathcal{E}}_n^+$.
828	\mathcal{R}_{t_n}	set	Per-time link predictions/results at t_n .
829	$\{G_n\}_{n=0}^{N-1}$	sequence	The sequence of snapshots defining \mathcal{G} .
830	G_{t_K}, G_{t_N}	snapshots	Snapshot after train time t_K , and final snapshot at t_N .
831	Modeling (GNN and scoring; Sec. 4.3)		
832	θ	parameters	Trainable parameters of the GNN.
833	$f_\theta(\cdot)$	mapping	GNN that computes object-node embeddings on \mathcal{G} .
834	\mathcal{F}	encoder	Text (and time-aware) embedding model for events.
835	h_o, h_e	vectors	Object and event embeddings, respectively.
836	$\text{reduce}(\cdot, \cdot)$	operator	Embedding combiner (e.g., concat/diff/dot).
837	$\text{MLP}(\cdot)$	mapping	Multi-layer perceptron used for scoring.
838	$\sigma(\cdot)$	function	Sigmoid activation.
839	$s_{o,e}$	score	Link-normality score for pair (o, e) .
840	$\hat{l}_{o,e}$	label	Predicted link label: $\mathbb{1}[s_{o,e} \geq \tau]$.
841	\mathcal{L}_k	loss	Balanced BCE loss at training step k .
842	η	scalar	Learning rate.
843	τ	threshold	Operating threshold for prediction.
844	ρ	ratio	Negative sampling ratio.
845	Data splits and indices		
846	$X_{\text{train}}, X_{\text{test}}$	sequences	Training and test splits (chronological).
847	K	integer	Index/time that separates train and test.
848	N	integer	Total number of entries/snapshots.
849	k, n	indices	Training step k , evaluation time n .
850	t_k, t_n	timestamps	Times associated with steps/entries.
851			

A.3 MODEL DESIGN SPACE

We compare two variants in our experiments: (i) Plain (ungated) GAT. We first concatenate the entity-type and entity-ID embeddings and pass them through a feed-forward projection to obtain the initial representation e_0 . We then run multi-layer, multi-head GATConv on an entity–entity graph induced by shared content to propagate messages and obtain e_{GAT} , which we use as the final entity representation. (ii) Gated fusion. Starting from the same e_0 and e_{GAT} , we introduce a global learnable scalar gate α and adaptively combine them via a sigmoid: $e = (1 - \sigma(\alpha))e_0 + \sigma(\alpha)e_{\text{GAT}}$. This biases toward e_0 when the given signal is weak (or absent) and toward e_{GAT} when the signal is strong. Both variants share the same link-prediction head: we take the entity representation and the content representation (text and time embeddings concatenated and then projected), compute their element-wise difference, and feed it to an MLP to output the link probability

864 A.4 ADDITIONAL VISUALIZATION
865

866 Figure 3 visualizes the distribution of anomaly likelihood scores of our five evaluation tasks. The
867 score distribution corroborates the main result in Table 2, that Analyst, Arxiv (Node), and HDFS are
868 three tasks relatively easy, with the score distribution of anomalies and normal examples separate
869 clearly. By contrast, the score of anomalies and normal examples mix up in Arxiv (Edge) and
870 Landslide, indicating that these datasets are more difficult.



890 **Figure 3: Anomaly score distribution of five tasks by GraphLogDebugger.** Score distributions
891 of anomalies and normal examples separate for simpler tasks and mix up for more difficult tasks.
892

893
894 A.5 PROMPTS IN RAG
895

896 We list the prompt we used in our RAG baseline as follows:

```

898 1 """Build context for ArXiv dataset (authors and paper titles)."""
899 2 context = """You are an expert at analyzing author-paper relationships in
900      academic research.
901 3
902 4 DATASET CONTEXT: This is a dataset of academic papers with their authors
903      and titles.
904 5 - Entities (authors): Research authors who wrote the papers
905 6 - Content (titles): The titles of the academic papers
906 7 - Edge: A connection between an author and a paper title (indicating the
907      author contributed to that paper)
908 8
909 9 TASK: Determine if the specific author-paper connection (edge) should
910      exist based on historical patterns.
911 10
912 11 EDGE ANALYSIS TARGET:
913 12 """
914 13
915 14 context += f"Author: {entity_name}\n"
916 15 context += f"Paper Title: {content_name}\n\n"
917 16
918 17 if similar_contents:
919 18     context += "SIMILAR PAPERS AND THEIR AUTHORS (for reference):\n"
920 19     context += "Use these examples to understand what types of authors
921      typically work on similar papers.\n\n"
922 20

```

```

918     21     for i, content_record in enumerate(similar_contents[:10]):
919     22         content = content_record.get('content', '')
920     23         entities = content_record.get('related_entities', [])
921     24         similarity = content_record.get('similarity', 0.0)
922     25         num_records = content_record.get('num_records', 0)
923     26
924     27         context += f"{i+1}. Paper Title: {content} (Similarity: {
925     28             similarity:.3f}, {num_records} records)\n"
926     29         context += f"    Authors who worked on this paper: {', '.join(
927             entities) if entities else 'None'}\n\n"
928     30     else:
929     31         context += "No similar papers found in historical data.\n\n"
930     32     context += """ANALYSIS QUESTION:
931     33     Based on the similar papers and their author patterns, should the
932     34     specified author-paper connection exist?
933     35     EVALUATION CRITERIA:
934     36     1. Research Domain Match: Does the author's expertise align with the
935         paper's topic?
936     37     2. Historical Patterns: Do authors with similar expertise appear in
937         similar papers?
938     38     3. Authorship Likelihood: Is it reasonable that this author would
939         contribute to this type of research?
940     39     4. Anomaly Detection: Does this connection seem unusual or out of place
941         compared to patterns in similar papers?
942     40
943     41     DECISION GUIDELINES:
944     42     - edge_exists = True: The author-paper connection makes sense based on
945         research area and historical patterns
946     43     - edge_exists = False: The author seems misplaced or unlikely to work on
947         this type of paper (anomalous edge)
948     44     - Consider the research fields, methodologies, and typical author
949         patterns shown in similar papers
950     45     - An edge is anomalous if the author appears completely unrelated to the
951         research domain of the paper
952     46
953     47     CONFIDENCE SCORING:
954     48     - High confidence (0.8-1.0): Clear patterns in similar papers strongly
955         support/reject the connection
956     49     - Medium confidence (0.5-0.7): Some evidence but less certain
957     50     - Low confidence (0.0-0.4): Limited historical data or unclear patterns
958     51     """
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
30
```

```

972 12 TASK: Determine if the specific Block ID-log connection (edge) should
973      exist based on historical patterns.
974 13
975 14 EDGE ANALYSIS TARGET:
976 15 """
977 16
978 17 context += f"Block ID: {entity_name}\n"
979 18 context += f"Log Content: {content_name}\n"
980 19 context += f"Content Analysis: Does '{entity_name}' appear in the log
981      content? {'YES' if entity_name in content_name else 'NO'}\n\n"
982 20
983 21 if similar_contents:
984     context += "SIMILAR LOG MESSAGES AND THEIR BLOCK IDs (for reference)
985         :\n"
986 22     context += "Use these examples to understand what types of Block IDs
987         typically appear in similar log messages.\n\n"
988 23
989 24 for i, content_record in enumerate(similar_contents[:10]):
990     content = content_record.get('content', '')
991     entities = content_record.get('related_entities', [])
992     similarity = content_record.get('similarity', 0.0)
993     num_records = content_record.get('num_records', 0)
994
995 31     block_ids = [e for e in entities if e.startswith('blk_')]
996 32     other_entities = [e for e in entities if not e.startswith('blk_')
997         ]
998
999 34     context += f"{i+1}. Log Content: {content} (Similarity: {
1000         similarity:.3f}, {num_records} records)\n"
1001 35     context += f"    Block IDs in this log: {', '.join(block_ids) if
1002         block_ids else 'None'}\n"
1003 36     if other_entities:
1004         context += f"    Other entities: {', '.join(other_entities
1005             [:3])}{'...' if len(other_entities) > 3 else ''}\n"
1006 38     context += "\n"
1007 39 else:
1008 40     context += "No similar log messages found in historical data.\n\n"
1009 41
1010 42 context += """ANALYSIS QUESTION:
1011 43 Based on the similar log messages and their Block ID patterns, should the
1012      specified Block ID-log connection exist?
1013 44
1014 45 EVALUATION CRITERIA:
1015 46 1. Block ID Presence: Does the Block ID appear within the log content
1016      itself? (This is crucial for HDFS)
1017 47 2. Log Operation Match: Does the Block ID relate to the HDFS operation
1018      described in the log?
1019 48 3. Historical Patterns: Do similar Block IDs appear in similar log
1020      messages?
1021 49 4. HDFS Block Behavior: Is it reasonable that this Block ID would be
1022      involved in this type of operation?
1023 50 5. Content Consistency: Block ID should be consistent between the BlockId
1024      column and the log content
1025 51
1026 52 DECISION GUIDELINES:
1027 53 - edge_exists = True: The Block ID-log connection makes sense based on
1028      HDFS block operations and historical patterns
1029 54 - edge_exists = False: The Block ID seems unrelated to this log message (
1030      anomalous edge)
1031 55 - CRITICAL: If the Block ID does NOT appear in the log content, this is
1032      likely anomalous
1033 56 - Consider HDFS block operations like allocation, storage, replication
1034      shown in similar messages
1035 57 - An edge is anomalous if the Block ID appears completely unrelated to
1036      the log operation

```

```

1026 58
1027 59 CONFIDENCE SCORING:
1028 60 - High confidence (0.8-1.0): Clear Block ID patterns and content
1029     consistency strongly support/reject the connection
1030 61 - Medium confidence (0.5-0.7): Some evidence but less certain about Block
1031     ID relevance
1032 62 - Low confidence (0.0-0.4): Limited historical data or unclear Block ID
1033     patterns
1034 63
1035 64 IMPORTANT: Focus specifically on Block ID relationships - Components and
1036     Event IDs are secondary for this analysis.
1037 65 """
1038
1039 1 """Build generic context for unknown datasets."""
1040 2 context = f"""You are an expert at analyzing entity-content relationships
1041     .
1042 4 EDGE ANALYSIS TARGET:
1043 5 Entity: {entity_name}
1044 6 Content: {content_name}
1045 7
1046 8 TASK: Determine if this entity-content connection should exist based on
1047     historical patterns.
1048 9 """
1049 10
1050 11 if similar_contents:
1051     context += "\nSIMILAR EXAMPLES:\n"
1052     for i, content_record in enumerate(similar_contents[:5]):
1053         content = content_record.get('content', '')
1054         entities = content_record.get('related_entities', [])
1055         context += f"{i+1}. Content: {content}\n    Related entities: {',
1056             '.join(entities)}\n\n"
1057 17
1058 18 context += """
1059 19 DECISION: Should this entity-content connection exist?
1060 20 - edge_exists = True: The connection makes sense based on patterns
1061 21 - edge_exists = False: The connection seems anomalous
1062 22 """

```

Listing 2: Prompt: HDFS

```

1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079

```

Listing 3: Prompt: Analyst and Landslide