A Critical Review of Common Log Data Sets Used for Evaluation of Sequence-Based Anomaly Detection Techniques

MAX LANDAUER, Austrian Institute of Technology, Austria
FLORIAN SKOPIK, Austrian Institute of Technology, Austria
MARKUS WURZENBERGER, Austrian Institute of Technology, Austria

Log data store event execution patterns that correspond to underlying workflows of systems or applications. While most logs are informative, log data also include artifacts that indicate failures or incidents. Accordingly, log data are often used to evaluate anomaly detection techniques that aim to automatically disclose unexpected or otherwise relevant system behavior patterns. Recently, detection approaches leveraging deep learning have increasingly focused on anomalies that manifest as changes of sequential patterns within otherwise normal event traces. Several publicly available data sets, such as HDFS, BGL, Thunderbird, OpenStack, and Hadoop, have since become standards for evaluating these anomaly detection techniques, however, the appropriateness of these data sets has not been closely investigated in the past. In this paper we therefore analyze six publicly available log data sets with focus on the manifestations of anomalies and simple techniques for their detection. Our findings suggest that most anomalies are not directly related to sequential manifestations and that advanced detection techniques are not required to achieve high detection rates on these data sets.

CCS Concepts: • **Software and its engineering** → **Software reliability**; • **Computing methodologies** → **Anomaly detection**.

Additional Key Words and Phrases: log data analysis, anomaly detection, data sets

ACM Reference Format:
Max Landauer, Florian Skopik, and Markus Wurzenberger. 2024. A Critical Review of Common Log Data Sets Used for Evaluation of Sequence-Based Anomaly Detection Techniques. *Proc. ACM Softw. Eng.* 1, FSE, Article 61 (July 2024), 22 pages. https://doi.org/10.1145/3660768

1 INTRODUCTION

Sound evaluations of machine learning algorithms are essential to validate their correct functioning, measure the accuracy of classifications, and conduct comparative studies with state of the art algorithms. Data sets are the basis for these evaluations and the selection of appropriate data sets is key to obtain representative results with general validity.

To be suitable for the purpose of evaluations, data sets are generally expected to fulfill several quality criteria [16], such as correctness, completeness, relevance, timeliness, realism, etc. When it comes to the evaluation of anomaly detection techniques that aim to identify rare and unexpected events in otherwise normal data [4], data sets and specifically the manifestations of anomalies must also meet the characteristics suitable for the detector under test. For example, anomaly detection techniques that process and analyze data instances as chronologically ordered sequences need

Authors' Contact Information: Max Landauer, Austrian Institute of Technology, Vienna, Austria, max.landauer@ait.ac.at; Florian Skopik, Austrian Institute of Technology, Vienna, Austria, florian.skopik@ait.ac.at; Markus Wurzenberger, Austrian Institute of Technology, Vienna, Austria, markus.wurzenberger@ait.ac.at.

ACM 2994-970X/2024/7-ART61
https://doi.org/10.1145/3660768

data sets where anomalies manifest as changes in sequential patterns rather than appearances of entirely new sequence elements that could be more effectively detected by other approaches.

There is an active research community that focuses on anomaly detection in system log data. Logs create a permanent record of almost all activities that take place on a system or within an application and are therefore a valuable source of information when it comes to failure analysis or forensic investigations of cyber incidents [4, 19]. Due to their large size and repeating patterns, anomaly detection approaches can capture the normal system behavior reflected in log data, disclose sudden deviations from these normal behavior models, and trigger alerts for anomalous states. Since log data follows the underlying workflow of applications, generated logs often form similar patterns of chronologically ordered event sequences. Logically, it is fair to assume that adverse activities also manifest as sequences and are thus detectable within sequential patterns, such as changes of certain event positions [41]. Accordingly, many anomaly detection algorithms make use of sequential patterns; specifically, deep learning methods such as Long Short-Term Memory Recurrent Neural Networks (LSTM RNNs) that are also commonly applied for text processing have been widely used in recent years [19]. Thereby, studies showed that approaches based on deep learning are generally able to outperform conventional machine learning such as clustering [5].

Accordingly, it is easy to get the impression that advanced detection techniques are necessary to achieve high detection performance on log data sets that are commonly used in scientific evaluations [19]. However, during manual analysis of these data sets, we noticed that many anomalies are either straightforward to detect or not directly related to changes of sequential patterns. Inspired by the work of Wolsing et al. [34], who show that SIMPLE (Sufficient, Independent, Meaningful, Portable, Local & Efficient) detection methods achieve competitive detection rates in comparison to complex approaches using neural networks in the field of industrial control systems, we develop and apply a set of simple yet effective and broadly applicable detection techniques, including detection of previously unseen event types and sub-sequences, unusually short or long sequences, sequences with deviating event occurrence counts, changes of event ordering, and delayed event occurrences. With this experimental study we aim to answer the following two research questions: *How do anomalies manifest themselves in common log data sets? What are drawbacks that render these data sets inadequate for evaluation of sequence-based anomaly detection techniques?*

The aim of this study is not to recreate or compare results from state of the art approaches, which has already been done in other surveys [5, 23, 39]. Instead, our focus lies on log data sets and their appropriateness for evaluating anomaly detection techniques. We provide scripts to reproduce the results presented in this paper online. In this paper we present insights into common data sets and show through a reproducible baseline that anomalies are not primarily related to sequences, which affects scientific evaluations using these data sets. We summarize our contributions as follows:

- A review of common log data sets and their properties relevant for anomaly detection,
- a baseline evaluation using a set of simple detection techniques, and
- a critical discussion on the suitability of the data sets for anomaly detection evaluations.

The remainder of this paper is structured as follows. Section 2 provides background of this research area and reviews related publications. Section 3 describes the log data sets covered in this study. Section 4 first outlines the detection techniques applied on the data sets and then provides the results of our evaluation study. Section 5 comprises a critical discussion of our findings with respect to the appropriateness of the data sets for scientific evaluations. Section 6 concludes the paper.

2 BACKGROUND & RELATED WORK

In their survey on anomaly detection in log data with deep learning, Landauer et al. [19] found that only a few publicly available log data sets are used by almost all of the reviewed publications.

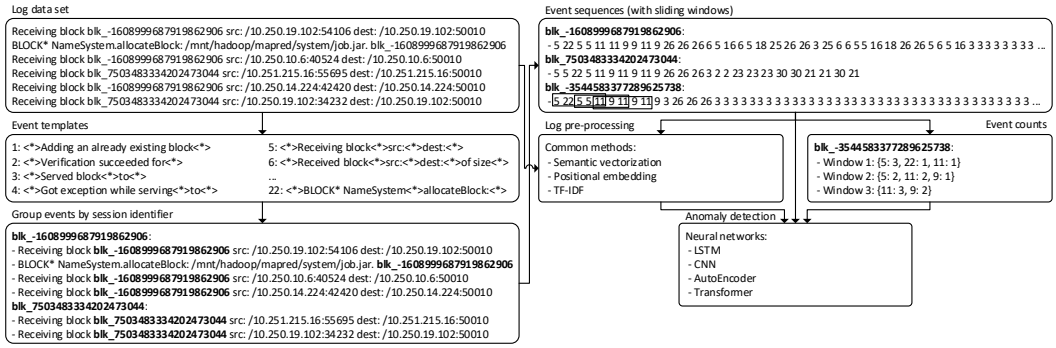

Fig. 1. Workflow for anomaly detection in log data.

The most commonly used data sets stem from applications that produce heterogeneous log events, i.e., each log message corresponds to a specific type of event, comprising static parts and variable parameters following a specific syntax. Consider the logs in the top of Fig. 1 as an example. In each log line except for the second one, "Receiving block" as well as "src:" and "dest:" are static, but the block identifier as well as source and destination address are variable. Event templates (sometimes also referred to as log keys or simply events [5]) are used to describe these event syntaxes and can be automatically generated by algorithms such as Drain [11]. The aforementioned sample lines correspond to event type 5, while the second line in the sample logs corresponds to event type 22 as visible in the "Event templates" block in Fig. 1.

The first step in the common anomaly detection workflow depicted in Fig. 1 consists of parsing the log data with templates, for the purpose of (i) assigning an event identifier to each log line and (ii) extracting parameters in a structured way. Several of the commonly used log files involve a so-called sequence identifier that can be extracted as one of the parameters and used to group events together that belong to the same process or trace. In the sample logs, this identifier corresponds to a data block (starting with "blk_" and followed by a unique ID) that is processed by the application. When such identifiers are not available, sequences are sometimes also generated by sliding windows of fixed size or time-windows over the parsed data set [23]. Either way, the resulting sequences consist of ordered lists of event types that are represented as distinct integers in the "Event sequences" block in Fig. 1. Some approaches additionally apply a sliding window on the sequences (a window of size 5 with step width of 2 is displayed in the figure as an example) [23], compute event counts in sequences or windows [26], transform log messages into numeric vectors with embedding techniques [41], apply weighting schemes such as TF-IDF [40], use neural networks on the raw log messages [10], etc. Eventually, the data fed into anomaly detection systems usually directly relates to event sequences, as sequential patterns are assumed to be the key indicator for anomalies.

Existing surveys on sequence-based anomaly detection techniques generally focus on deep learning applications. For example, Le et al. [23] quantitatively compare five state of the art anomaly detection models on four data sets with focus on data pre-processing strategies, data set imbalance, and robustness to noise. Their results suggest that detection capabilities of complex models are heavily influenced by these aspects and that actually achieved detection rates are often not as good as expected as a consequence. Chen et al. [5] provide another quantitative survey that involves four unsupervised and two supervised deep learning approaches that are evaluated on two data sets. They found that anomalies incorrectly inserted in training data as well as unknown event types can strongly impact detection capabilities. In addition, they point out that conventional machine learning models such as clustering, PCA, or invariant mining, are typically more efficient than their deep learning counterparts in terms of runtime.

Table 1. Overview of common log data sets

		HDFS (Xu et al. [38])		BGL (CFDR [27])		Thunderbird (CFDR [27])		OpenStack (Loghub [12])		Hadoop (Loghub [12])		ADFA (Creech et al. [6])	
		Abs.	Rel.	Abs.	Rel.	Abs.	Rel.	Abs.	Rel.	Abs.	Rel.	Abs.	Rel.
Number of lines	Total	11,197,705	100%	4,747,963	100%	211,212,192	100%	207,820	100%	394,308	100%	2,747,550	100%
Number of parsed events	Total	12,580,989	112.4%	4,747,963	100%	211,212,192	100%	57,784	27.8%	393,433		2,747,550	100%
	Normal	12,255,230	97.4%	4,399,265	92.7%	193,744,733	91.7%	52,290	90.5%	25,285	6.4%	2,430,162	88.5%
	Anom.	325,759	2.6%	348,698	7.3%	17,467,459	8.3%	5,494	9.5%	368,148	93.6%	317,388	11.5%
Number of event types	Total	33	100%	394	100%	6,178	100%	30	100%	349	100%	175	100%
	Normal	20	60.6%	339	86.0%	5,577	90.3%	30	100%	229	65.6%	174	99.4%
	Anom.	32	97.0%	60	15.2%	652	10.6%	27	90.0%	345	98.9%	90	51.4%
Number of sequences	Total	575,061	100%	69,252	100%	5,855	100%	2,069	100%	978	100%	5,951	100%
	Normal	558,223	97.1%	37,823	54.6%	1,546	26.4%	1,871	90.4%	167	17.1%	5,205	87.5%
	Anom.	16,838	2.9%	31,429	45.4%	4,309	73.6%	198	9.6%	811	82.9%	746	12.5%
Number of unique sequences	Total	26,814	4.7%	27,571	39.8%	5,855	100%	27	1.3%	223	22.8%	3,852	64.7%
	Normal	21,690	80.9%	8,874	32.2%	1,546	26.4%	24	88.9%	52	23.3%	3,122	81.0%
	Anom.	5,133	19.1%	18,697	67.8%	4,309	73,6%	12	44.4%	195	87.4%	732	19.0%
Number of sequences that also occur in other class	Normal	14	0.003%	0	0.0%	0	0.0%	1,842	98.5%	139	83.2%	2	0.04%
	Anom.	17	0.1%	0	0.0%	0	0.0%	195	98.5%	612	75.5%	6	0.8%
Number of unique event count vectors	Total	666	2.5%	15,579	56.5%	5,855	100.0%	16	59.3%	198	88.8%	3,502	90.9%
	Normal	257	38.6%	4,605	29.6%	1,546	26.4%	14	87.5%	51	25.8%	2,772	79.2%
	Anom.	418	62.8%	10,974	70.4%	4,309	73.6%	7	43.8%	174	87.9%	732	20.9%
Number of event count vectors that also occur in other class	Normal	230	0.04%	0	0.0%	0	0.0%	1850	98.9%	143	85.6%	2	0.04%
	Anom.	316	1.9%	0	0.0%	0	0.0%	196	99%	619	76.3%	6	0.8%
Collection date		Nov. 2008		2005-2006		2005-2006		May 2017		Oct. 2015		≈ 2013	
Duration		≈ 38.7 hours		≈ 214 days		≈ 244 days		≈ 58.8 hours		≈ 50.5 hours		Unknown	
Distinct anomaly labels		1		43		33		1		3		6	
Label type		Sequences		Events		Events		Sequences		Sequences		Sequences	

Landauer et al. [19] provide a qualitative survey of 62 state of the art approaches. Their work emphasizes the diversity of deep learning models, pre-processing strategies, and methods to transform log events into representations suitable for ingestion by neural networks. Yadav et al. [39] provide another qualitative survey of detection approaches and discuss common challenges, model architectures, and pre-processing strategies. They also summarize the data sets used in scientific publications, but do not delve into their individual properties.

The focus of aforementioned publications always lies on detection techniques; we are not aware of any works that critically analyze data sets used to evaluate these approaches. Kenyon et al. [16] review the appropriateness of publicly available data sets in the intrusion detection domain, however, these are not the data sets typically used to evaluate sequence-based anomaly detection techniques [19]. With this paper we aim to close this gap and support researchers to better understand these data sets, challenge evaluations with detection benchmarks, and avoid misinterpretation of results.

3 ANALYSIS OF LOG DATA SETS

This section describes the most common log data sets used in scientific evaluations. The data sets are selected and ordered based on the survey by Landauer et al. [19]. We include one additional data set that we suggest as a potentially useful alternative. The following sections outline the origin and properties of each data set. Thereby, we generally refer to Table 1, which quantitatively summarizes the data sets with respect to the distribution of normal and anomalous instances.

3.1 HDFS

The HDFS log data set is the most frequently used data set for evaluations of anomaly detection techniques [19] and thus the focus point of this study. The logs stem from the Hadoop Distributed File System (HDFS), which allows storage and processing of large files. Each log event contains one or more data block identifiers that enable grouping of events into sequences as discussed in the previous section. In fact, the sample logs shown in Fig. 1 are taken from the HDFS data set. The main idea of detecting anomalies in this data set is that some data blocks are not processed correctly by the system, which is reflected by the log events generated as the block goes through an abnormal execution flow, in which case the entire sequence should be detected as anomalous [37]. Note that lines containing multiple block identifiers have to be replicated for each corresponding sequence, thus the number of parsed events exceeds the number of lines (cf. Table 1).

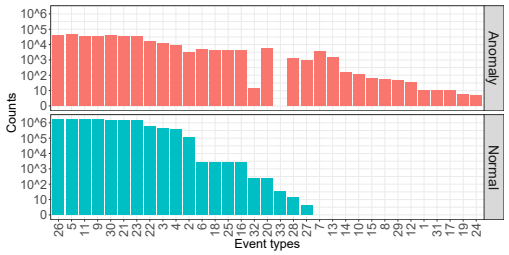

Fig. 2. Event frequencies in HDFS event sequences.

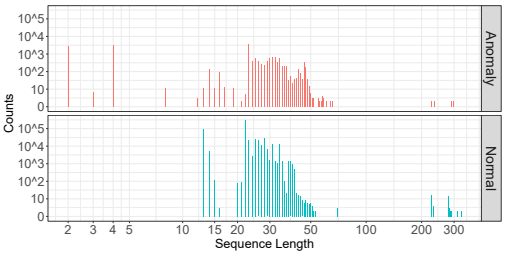

Fig. 3. Distribution of HDFS event sequence lengths.

```
Occurrence frequencies of the most common normal sequences of event types:
73691: 5 5 5 22 11 9 11 9 11 9 26 26 26 23 23 23 30 30 30 21 21 21
40156: 5 22 5 5 11 9 11 9 11 9 26 26 26 23 23 23 30 30 30 21 21 21
37788: 5 5 22 5 11 9 11 9 11 9 26 26 26 23 23 23 30 30 30 21 21 21
31213: 5 5 5 22 11 9 11 9 11 9 26 26 26
19311: 5 22 5 5 11 9 11 9 11 9 26 26 26
18297: 5 5 22 5 11 9 11 9 11 9 26 26 26
18078: 22 5 5 5 26 26 26 11 9 11 9 11 9 23 23 23 30 30 30 21 21 21
```

```
Occurrence frequencies of the most common anomalous sequences of event types:
1643:  5 22
1361:  22 5 5 7
1307:  22 5
1252:  5 5 22 7
612:   5 22 5 7
176:   22 5 5 5 26 26 26 11 9 11 9 11 9 23 23 23 30 30 30 21 21 28 26 30 21
158:   5 5 5 22 11 9 11 9 11 9 26 26 26 23 23 23 21 30 30 30 20 21 21
```

Fig. 4. Top seven most common normal (left) and anomalous (right) sequences and their respective occurrence counts in the HDFS data set.

The data set was originally collected in 2008 by Xu et al. [36–38] on a Hadoop cluster comprising more than 200 nodes. The data set was labeled by domain experts on the granularity of individual block identifiers. As stated by the original authors, labels were assigned to more than 575,000 event sequences by clustering them into event count vectors, which reduced their size to the manageable amount of only 680 unique vectors [36]. Note that we obtain a total of 666 unique count vectors (cf. Table 1), likely due to the fact that publicly available event templates do not fully align with the templates used by the original authors. Even though the original logs are still available, the data set has also been provided by the log data set collection project Loghub [12]. A close inspection of that data set reveals that around 22,000 lines are missing in comparison to the original data set for unknown reasons. In addition, many authors rely on parsed versions of the HDFS data set that are available in public repositories, such as the LogDeep project [1]. Given that the parsed versions lack sequence identifiers, it is difficult to ascertain their completeness. Moreover, since this version of the data set only comprises sequences but lacks timestamps, some authors incorrectly assume that timestamp information is not available even though it is present in the original logs [23].

Figure 2 depicts the frequencies of event types represented as integer values in anomalous (top) and normal (bottom) sequences, sorted in ascending order. The plot shows that the most common event types have a similar distribution of relative occurrence frequencies in both normal and anomalous sequences (their absolute numbers diverge as normal sequences are more frequent). However, many event types only occur in anomalous sequences and can thus be regarded as basic indicators for anomalies. On the other hand, event type 33 acts as an indicator for normal events, even though it is relatively rare, and event type 20 occurs in both normal and anomalous sequences, but is more likely to indicate anomalies as it occurs far more frequently in anomalous sequences.

Figure 3 shows that many normal and anomalous sequences also differ in length. In particular, while all normal event sequences have lengths larger than 12, more than a third of the anomalous sequences have short lengths of 2, 3, or 4.

Another important aspect to consider when evaluating anomaly detection techniques with the HDFS data set is that many sequences are identical. As stated in Table 1, the total number of 575,061 event sequences can be reduced to only 26,814 (4.7% of total sequences) unique sequences. Even more peculiar is the fact that these unique sequences comprise only 666 (2.5% of unique sequences or 0.1% of total sequences) unique count vectors, i.e., lists of event frequencies that are

independent from event positions in sequences. One of the reasons for this is that events that occur more or less simultaneously end up in random order. Consider the most common normal sequences displayed in Fig. 4, where the three most common sequences are identical except for the position of event type 22 among the first four events, which usually occur simultaneously according to their timestamp. Given that this effect also occurs with other event types in the sequence, the number of possible event combinations exhibited by otherwise identical sequences becomes enormous, thus resulting in the large gap between unique sequences and unique count vectors. Figure 4 also shows the compositions of the most common anomalous sequences, which only comprise few elements. While event type 7 acts as an indicator for anomalies, some of these sequences only involve normal events and thus need to be detected through their short lengths. The last two sequences show that anomalous sequences generally involve the same patterns as normal sequences, but contain additional event types such as 28 or 20 that are indicators for anomalies as visible in Fig. 2.

3.2 BlueGene/L (BGL)

The BlueGene/L (BGL) log data set is provided by the Computer Failure Data Repository (CFDR) [28] and was originally described by Oliner et al. [27]. The log data set stems from a supercomputer located at the Lawrence Livermore National Laboratory (LLNL) that first comprises 32,768 dual-processor nodes and was upgraded to 65,536 nodes during log data collection in 2005 and 2006. We refer to the publication by Taerat et al. [33] for a detailed technical description of the system. Similar to the HFDS log data set, Loghub [12] provides another version of the data set where few lines are labeled differently for unknown reasons.

Aside from log messages, the data set contains location identifiers for components such as individual compute nodes, I/O nodes, service cards, links cards, etc., that can be used to group events into sequences. Interestingly, only some authors make use of the node identifiers to generate sequences [13, 23], while others state that it is not possible to distinguish different job executions and thus simply partition the whole data set into time-windows [5]. According to our analysis, leveraging the node identifiers to group events into sequences is a reasonable approach since many similar sequences emerge, indicating that different nodes go through similar event execution flows. However, we acknowledge that processes may call functions from multiple components, thereby creating sequential dependencies that only become apparent when considering logs across nodes. In the following, we consider both sequence identifiers and time windows as grouping strategies.

One major difference compared to the HDFS log data set is that labels are provided on the granularity of events rather than sequences. Thereby, the labeled events form almost completely disjoint sets, i.e., the same event types are consistently labeled throughout the data set. In other words, the label of a specific event does not depend on the context of occurrence of the event but can be inferred from the event message itself. To enable comparability in Table 1, we therefore adopt the approach of existing works [5, 23] that tackle this issue by considering the whole sequence as anomalous if it contains at least one anomalous event. Analogous to our analysis of the HDFS file, we plot the event frequencies of normal and anomalous sequences in Fig. 5, which indicates that a large fraction of events act as basic indicators for either normal or anomalous sequences.

We also analyze the distribution of sequence lengths in the BGL data set but omit the plot for brevity. Other than for the HDFS logs, there is no simple way to discern anomalous and normal sequences based on their lengths; however, the plot reveals that even though the vast majority of sequences comprises less than 1000 events, there are a few long anomalous sequences comprising more than 10,000 and even 100,000 events. Given that a single labeled event in an otherwise normal sequence is enough so that the whole sequence is regarded anomalous, it stands to reason to partition long sequences into contiguous sub-sequences with individual labels. To keep evaluations consistent across data sets, we leave the sequences unchanged for the purpose of this paper.

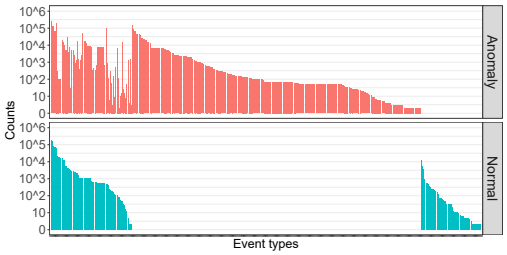

Fig. 5. Event frequencies in BGL event sequences.

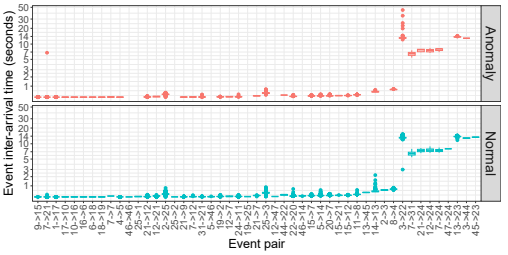

Fig. 6. Event inter-arrival times in OpenStack.

3.3 Thunderbird

Thunderbird is yet another supercomputer log data set provided by CFDR [27, 28]. The data set was collected at Sandia National Labs (SNL) at the same time as the BGL data set, but involves considerably more lines and distinct event types. Due to the high complexity of the data set, authors commonly only use a fraction of the data set, such as the first 5 million [23] or 20 million lines [9]. For the purpose of this paper, we consider the whole data set for completeness.

Similar to the BGL data set, even though there are sequence identifiers in the data, they are usually not leveraged by existing works, which resort to window- or time-based grouping [23]. These sequence identifiers mainly correspond to node or interface names and may be divided into groups: 77.1% comprise two characters and digits (e.g., “bn251”), 17.1% comprise a period and digits (e.g., “.741”), 4.8% comprise complex descriptors (e.g., “dr13iblsw2”), 0.4% are user names (e.g., “tbird-admin1”), 0.3% start with “ibr” (e.g., “ibr6-northern”), 0.2% start with number signs (e.g., “#31#”), and 0.1% are IP addresses; sequence patterns have different characteristics across groups, e.g., identifiers starting with periods generally involve shorter sequences than others. As for the BGL data set, we point out that component-based sequence formation conceals dependencies of events across nodes; we therefore consider multiple grouping strategies in the following. More similarities to BGL are that the labels of the data set are assigned to single events and that the events labeled as anomalous are discernible from normal events by the syntax of the log message, even without considering their context of occurrence, i.e., events occurring before or after.

3.4 OpenStack

The OpenStack log data set was generated by the authors of DeepLog [7], one of the most influential papers in the research area around anomaly detection in log data [19]. Other than the previous log data sets, the OpenStack data set was synthetically produced for the purpose of evaluating anomaly detection techniques. In particular, the authors executed a script that repeatedly carries out tasks from a pre-defined list, such as starting, stopping, pausing, and resuming virtual machines. At specific points in time while running this script, the authors injected three types of anomalies, including a timeout and errors during destruction and cleanup of virtual machines. Some of the logs contain identifiers for virtual machine instances that enable log grouping and the formation of event sequences. However, as visible in Table 1, only 27.8% of all lines contain such an identifier; all other lines are omitted from our analysis. Unfortunately, the original data set seems to be no more available. We therefore resort to the version provided by Loghub [12].

As Table 1 shows, there is a high overlap of 98.5% between normal and anomalous sequences, i.e., these sequences are identical. This renders application of anomaly detection techniques that only consider event sequences without contextual information infeasible. This fact by itself is not surprising: According to the original authors, anomalies should manifest by the time taken to build instances, which is reflected by the inter-arrival time of events in sequences; however, this does

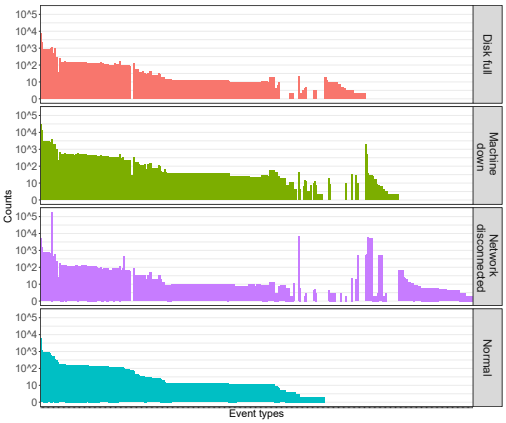

Fig. 7. Event frequencies in Hadoop event sequences.

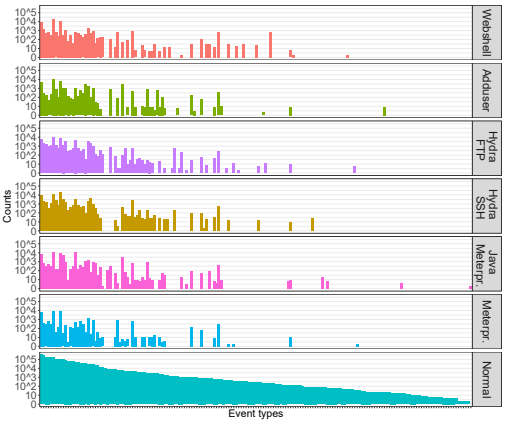

Fig. 8. Event frequencies in ADFA event sequences.

not seem to be the case for the log data at hand. Figure 6 depicts the distribution of inter-arrival times between all consecutive events as boxplots for normal and anomalous sequences, only to reveal that there are no significant deviations with respect to event inter-arrival times, specifically the times to reach events corresponding to starting a virtual machine (event type 22), stopping a virtual machine (event type 23), and deleting a virtual machine (event type 24). Only 5 out of 198 anomalous sequences show an increase of inter-arrival time between event type 3 and event type 22. Similar issues regarding anomalies have also been observed by other authors using the OpenStack data set [10, 15]. Therefore, we do not further investigate the data set in this study.

3.5 Hadoop

Similar to OpenStack, the Hadoop data set was synthetically generated in a lab environment to evaluate log sequence clustering [25]. The authors follow the approach from Shang et al. [30] and execute WordCount and PageRank applications on a Hadoop cluster. After some normal runs, the authors manually inject three distinctly labeled anomaly cases by turning off the server (machine down), disconnecting the server (network disconnected), and filling up the hard disk (disk full). While the original data set is not available anymore, a Hadoop data set containing the same anomalies is provided by Loghub [12]. Other than the previous data sets, logs from each run of an application are placed in separate files, which makes it easy to group events into sequences.

Another similarity to the OpenStack data set is that there is a high overlap between normal and anomalous sequences, with 83.2% of normal sequences having at least one identical counterpart in the anomalous set and 75.5% of anomalous sequences being identical to one or more normal sequences. Accordingly, analysis of sequences alone is not an adequate approach for anomaly detection. Figure 7 shows that the overall distribution of event frequencies is similar across all classes, except for few anomalous sequences that involve new event types that do not occur in the normal case. Closer inspection shows that many of the events only occurring in anomalous classes are the result of a single application run, for example, the log message “Failed to renew lease” is printed every single second in affected applications. Given that most sequences are identical, it is also not possible to recognize anomalies by their sequence lengths. We manually compared the event parameters of common identical sequences but were also unable to identify any differences that discern normal from anomalous instances. Despite these problems, we use this data set in our evaluation study to emphasize issues with misleading results obtained from evaluation metrics.

3.6 ADFA

The Australian Defence Force Academy Linux Dataset (ADFA-LD) was generated by Creech et al. [6] in 2013 to overcome issues with log data sets that were commonly used for evaluation of intrusion detection techniques at that time. Other than the aforementioned data sets that focus on system failures, the ADFA data set makes use of cyber attacks to generate anomalies in log data. In particular, the authors of the data set created a test environment with web applications vulnerable to known attacks and collected low-level system call logs during normal operation (e.g., web browsing or document processing) as well as execution of six attack cases.

Figure 8 shows that all event types that occur in the data set also occur in sequences of the normal class, i.e., there are no event types that act as indicators for specific attacks, except for a single occurrence of event type 173 occurring during the "Java Meterpreter" attack. The overall event frequency distributions show that the system calls generated by attacks differ from the event frequencies during normal operation, which suggests that detection of anomalies and even classification of attack cases is feasible. We also analyzed the sequence lengths and confirm that anomalous sequences are not shorter or longer than normal ones.

Even though this data set is rarely used for anomaly detection evaluations (no publication reviewed by Landauer et al. [19] uses this data set), we add it to our evaluation study in an attempt to propose a new data set for future evaluation that overcomes issues with commonly used data sets. In particular, using system calls avoids the need for parsing since the operations are represented as distinct integer numbers and thus reduces the influence of parsing on the detection accuracy. Moreover, system calls are generally ordered in a consistent way and are therefore not affected by permutations of simultaneously occurring events as it is the case in the HDFS log data set.

3.7 Data Set Complexity

We compare the data sets described in the previous sections with respect to the complexities of their sequential patterns. To this end, we apply measures for entropy and repetitiveness.

3.7.1 Entropy. We use entropy to measure how evenly distributed certain parts of the sequences (i.e., contiguous sub-sequences) are across the data set, where low entropy indicates that some of these parts are occurring much more often than others and high entropy corresponds to a more random distribution. Since entropy does not account for sequential ordering, we leverage N-grams of various sizes and compute the entropy for each N separately. To compensate for the varying numbers of distinct event types in the data sets, we also compute the normalized entropy that bases on the maximum entropy that is reached if all N-grams occur the same number of times. Figure 9 shows that the Thunderbird and ADFA data sets yield the highest total entropy, suggesting that they involve many diverse and complex sequential patterns. For the Thunderbird data set, this is caused by the large number of distinct events, as visible by the normalized entropy that is comparatively low. The ADFA data set yields the highest normalized entropy and is closely followed by the HDFS data set, especially for large N, suggesting that these data sets are the most complex ones considering their number of distinct events. The entropy computed for OpenStack data set is comparatively low and does not increase for higher N, indicating that there is hardly any complexity in the data. This is confirmed by the normalized entropy, which starts out with a comparatively high value for $N = 1$, meaning that single event frequencies are more evenly distributed than in other data sets, but quickly diminishes for larger N.

3.7.2 Repetitiveness. Arguably, the number of distinct event types appearing in a data set is an indicator for its complexity as larger numbers of events have the potential to form more diverse patterns (the numbers of distinct events are stated in Table 1). To assess whether they actually

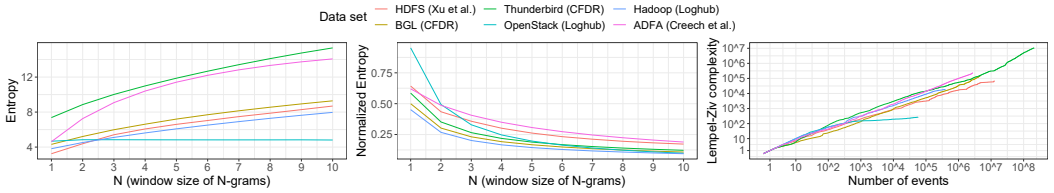

Fig. 9. Entropy (left) and normalized entropy (center) of N-grams as well as sequence repetitiveness measured with Lempel-Ziv complexity (right).

form such complex sequences or instead occur in the same patterns that repeat over and over, we leverage the Lempel-Ziv complexity [24]. In short, this measure counts the number of different contiguous sub-sequences encountered when processing all sequences from beginning to end, where already observed sub-sequences are stored in a dictionary for all sequences in a data set. The plot on the right hand side of Fig. 9 shows the Lempel-Ziv complexity with respect to the total number of events within each consecutively processed sequence. The plot shows that all data sets except for OpenStack roughly exhibit the same level of complexity, with the HDFS data set ending up with a slightly lower complexity than the other ones. OpenStack, however, breaks out of the overall trend at around 1,000 processed events, after which most of the subsequently analyzed sequences contain already observed sub-sequences, causing that the total complexity levels off.

4 EVALUATION STUDY

In this section we evaluate how anomalies manifest in common log data sets. We first outline the setup of our experiment and then briefly describe every detection technique we apply on the data sets before summarizing the results.

4.1 Setup

The purpose of our study is to assess the appropriateness of five of the previously described data sets (HDFS, BGL, Thunderbird, Hadoop, and ADFA) for evaluating anomaly detection techniques leveraging event sequences. We therefore design an experiment that evaluates simple detection mechanisms on the data sets and measures whether they are sufficient to achieve competitive detection rates. Comparing different detection mechanisms with each other then allows us to better understand how anomalies manifest themselves in each data set. We ensure that the selected detection techniques are as generic as possible to be applicable to all sorts of data sets and simple enough so that it is easy to understand why specific instances are reported as anomalies. Similar to common deep learning methods [19], our detection techniques also only process grouped event type sequences without any contextual information other than the timestamp.

The focus of our evaluation study lies on semi-supervised detection, where only instances of a fraction of the normal class are available for training. This is the most common scenario for anomaly detection as anomalies generally correspond to unexpected or unusual system behavior that is non-trivial to define in advance [19]. We follow the strategy from Du et al. [7] and randomly sample 1% of the normal sequences for training, except for the Hadoop data set where we use 10% since only 167 normal sequences are available. After training, we run the detector on the test data, which comprises the remaining normal sequences as well as the anomalous sequences, and measure its ability to discern these two classes. In particular, we count true positives (TP) as correctly detected anomalous sequences, true negatives (TN) as correctly undetected normal sequences, false positives (FP) as incorrectly detected normal sequences, and false negatives (FN) as incorrectly undetected anomalous sequences. Based on these counts we then compute precision ($Prec = \frac{TP}{TP+FP}$), recall or true positive rate ($Rec = TPR = \frac{TP}{TP+FN}$), true negative rate ($TNR = \frac{TN}{TN+FP}$), and F1 score

($F1 = \frac{2 \cdot Prec \cdot Rec}{Prec + Rec}$). We repeat sampling and evaluation 25 times for each data set and each detector to also capture the variance of our results. Similar to Le et al. [23], we also run our evaluation with chronologically sorted sequences, i.e., the earliest 1% of sequences are used for training.

Our detection techniques rely either on none or on one single threshold for detection. In the latter case, we iterate over all thresholds between 0 and 1 in steps of 0.01 and select the one that maximizes F1. For approaches based on event prediction using deep learning, the number of considered event candidates acts similar to a threshold; to ensure fair comparison with simple detectors, we iterate over candidate sets of size 1 to 100 and select the one that maximizes F1. We investigate the influence of these thresholds in Sect. 4.3.1 and Sect. 4.3.2. Due to the nature and label granularity of BGL and Thunderbird data sets (cf. Sect. 3.2 and Sect. 3.3), we additionally evaluate them by computing aforementioned metrics using time-window grouping as well as individual events and present the results in Sect. 4.3.2 and Sect. 4.3.3.

4.2 Detection Techniques

This section describes each detection technique that is applied on the selected data sets.

4.2.1 New Event Types. The detection for new event types monitors all events that appear in any sequence of the training data set to build a model of distinct event types that are known to occur during normal operation. In the detection phase, all sequences that contain one or more event types that are not known from the training phase are reported as anomalies. This is the most basic detection technique but has been effectively used in intrusion detection systems such as the AMiner [22]. It is expected to work well in data sets where normal and anomalous event types resemble disjoint sets, such as the BGL and Thunderbird data set, but also achieve good performance in data sets with many event types that act as indicators for anomalies, such as the HDFS data set.

4.2.2 Deviating Sequence Lengths. This detection technique learns the minimum and maximum sequence lengths of all sequences in the training data. Subsequently, all sequences in the test data set with lengths shorter than the minimum or longer than the maximum are reported as anomalies. This detection technique specifically aims to recognize unusually short sequences in the HDFS data set (cf. Fig. 3). We also combine this technique with the detection for new events, so that sequences that are reported by either one of those two techniques are considered anomalies.

4.2.3 Event Count Vector Clustering (ECVC). The idea behind this detection technique is that normal sequences that occur in the test data are similar to one or more sequences in the training data in terms of event types and their respective frequencies, while anomalous sequences are dissimilar to every sequence of the training data. For this purpose, we create event count vectors for sequences, where each vector index corresponds to a specific event type and the value at that index reflects the number of times the event type occurs in a sequence. After transforming all training sequences into count vectors, we use the L_1 norm[1] as a similarity metric that classifies count vectors from the test data set as anomalous if their similarity to all of the training instances is lower than a threshold. Note that we use the L_1 norm due to the fact that it handles high-dimensional data better than higher-order norms [2] and has already been applied with count vectors for anomaly detection in user behavior patterns [20]. In the following, we also consider a variant of this method where the count vector indices are weighted higher when the corresponding event types only appear in few sequences, analogous to the well-known TF-IDF measure [29]. In addition, we combine this technique with detection based on new events and sequence lengths in the following, where sequences are detected as anomalous if any of the combined techniques reports them as such.

[1]Distance between vectors a and b is computed by $\sum_i |a_i - b_i|$.

4.2.4 N-grams. This detection technique runs a sliding window of size N at a step width of 1 over all training sequences and learns the ordered sub-sequences of event types inside the window (note that the term sub-sequences always refers to contiguous chunks of sequences in this paper). In the detection phase, a window of the same size slides over the test sequences and the sub-sequences inside the window are compared with the ones known from training. Forrest et al. [8] proposed to count the number of mismatching sub-sequences and determine the whole sequence as anomalous if the normalized count exceeds a certain threshold. The optimal value for N is non-trivial to determine and is highly dependent on the data. In our experiments, we therefore use 2, 3, and 10 as values for N as these window sizes have shown to be useful choices in other works [6–8, 19].

4.2.5 Edit Distance. This detection technique is similar to the ECVC in the sense that it computes a distance between a test sequence and every training sequence and classifies it as anomalous if no sufficiently similar pair is found. However, other than the ECVC that makes use of unordered count vectors, this detector computes the normalized edit distance, which counts the number of insertions, deletions, and replacements of event types to transform one sequence into another, which inherently relies on the ordering of event types in sequences [29]. As such, this detection technique is able to detect anomalies that manifest as additional or missing event types as well as changes of event ordering. Accordingly, this detection technique should work better than ECVC for anomalies with sequential manifestations, since ECVC only considers event occurrences independent from their position in the sequences.

4.2.6 Event Timing. Most activities that occur in specific states of processes take a certain amount of time that remains relatively steady or at least within a certain range throughout multiple executions. This is reflected in the inter-arrival times of log events that usually mark the start, stop, or intermediate steps of such processes. For example, consider the inter-arrival times displayed in Fig. 6, where the time to start a virtual machine is always in the range of 12 to 15 seconds. The assumption of this detection method is that in some anomalous executions, the event types remain the same, but the activities carried out in between take much longer or shorter. For our simple detection method we therefore compute the minimum and maximum passed time between each pair of consecutive event types in the training data set and then classify test sequences as anomalies if the inter-arrival time between any of the involved event pairs deviates too much from the range learned specifically for this event pair, i.e., the relative difference to the range boundary exceeds a certain threshold. We opted against statistical tests (e.g., for normal distributions) on the inter-arrival times since some event pairs occur few or even just a single time in the training data.

4.2.7 Deep Learning. Among the many approaches that have been proposed for anomaly detection in log data using deep learning models, we select DeepLog [7] and LogAnomaly [26] since they are capable of semi-supervised operation, widely used as benchmarks in scientific publications, and available as open-source implementations [1]. Both approaches leverage LSTM RNNs to process event sequences in chunks and predict the upcoming event type. In case that the actual event type is not among the top candidates of the prediction, the entire sequence is considered anomalous. While DeepLog [7] only relies on sequential information, LogAnomaly [26] additionally ingests event count vectors. We refer to the respective publications for detailed descriptions of the approaches. We relied on default parameters, i.e., a window size of 10 events, two hidden layers with 64 nodes each, a learning rate of 0.001 with a decay of 0.1, and trained the neural networks using the Adam optimizer for 370 epochs. Due to computational constraints and the large size of the Thunderbird data set, we randomly selected 5% of the sequences and set the number of epochs to 100 when processing this data set. All computations were carried out on a Tesla V100S 32GB GPU.

Table 2. F1 scores achieved on randomly/chronologically sorted sequences

	Sequence-identifier grouping						Time-window grouping	
	HDFS (LogDeep [1])	HDFS (Xu et al. [38])	BGL (CFDR [27])	Thunderbird (CFDR [27])	Hadoop (Loghub [12])	ADFA (Creech et al. [6])	BGL (CFDR [27])	Thunderbird (CFDR [27])
Event	63.3/-	53.9/73.3	98.8/-	92.4/-	28.8/27.5	22.1/-	68.0/47.0	**83.1**/76.4
Length	54.1/-	56.0/4.6	2.8/-	65.1/-	19.4/10.8	2.9/-	3.4/2.5	6.1/43.5
Event + Length	90.4/-	72.0/5.6	98.7/-	91.9/-	33.5/28.9	22.4/-	67.0/46.7	82.8/76.4
ECVC	96.0/-	96.0/53.9	93.5/-	94.7/-	**91.5**/**91.5**	33.9/-	53.0/42.5	80.4/76.2
Event + Length + ECVC	**98.8**/-	96.4/6.6	98.7/-	91.9/-	**91.5**/**91.5**	31.6/-	66.4/46.7	82.8/76.4
ECVC (idf)	96.3/-	96.5/82.9	93.7/-	94.7/-	**91.5**/**91.5**	31.0/-	54.5/42.6	81.6/76.2
Event + Length + ECVC (idf)	96.9/-	**97.1**/7.0	98.7/-	91.9/-	**91.5**/**91.5**	30.6/-	66.5/46.7	82.8/76.4
2-gram	85.2/-	86.2/53.8	92.3/-	84.9/-	**91.5**/**91.5**	27.9/-	34.1/31.4	75.8/75.8
2-gram + Length	95.1/-	87.8/5.6	96.1/-	84.9/-	**91.5**/**91.5**	26.5/-	47.0/41.9	75.8/75.8
3-gram	81.5/-	85.3/53.8	62.7/-	84.9/-	**91.5**/**91.5**	28.4/-	28.6/28.6	75.8/75.8
10-gram	33.5/-	5.7/5.7	62.7/-	84.9/-	**91.5**/**91.5**	28.2/-	28.6/28.6	75.8/75.8
Edit	71.8/-	71.6/53.9	93.4/-	**95.2**/-	**91.5**/**91.5**	**34.3**/-	47.0/38.5	79.4/77.0
Event + Length + Edit	97.8/-	83.3/6.8	98.7/-	91.9/-	**91.5**/**91.5**	32.4/-	65.8/46.6	82.8/76.4
Event Timing	-/-	45.0/13.7	81.6/-	89.1/-	89.1/**91.5**	-/-	24.3/24.3	74.5/75.8
DeepLog	94.5/-	85.6/86.8	**98.9**/-	94.4*/-	**91.5**/**91.5**	31.1/-	68.2/**48.5**	80.2*/80.5*
LogAnomaly	97.6/-	87.4/**88.1**	**98.9**/-	94.5*/-	**91.5**/**91.5**	31.3/-	**69.3**/48.4	80.2*/**85.4***
Le et al. [23]	96.6/-	-/-	99.9/-	-/-	-/-	-/-	94.9/74.4	41.4/40.0
He et al. [13]	83.4/-	-/-	90.2/-	-/-	-/-	-/-	-/-	89.7/-
Guo et al. [9]	82.3/-	-/-	-/-	-/-	-/-	-/-	90.4/-	96.6/-
Yang et al. [40]	-/-	-/98.8	-/-	-/-	-/-	-/-	-/98.2	-/-
Meng et al. [26]	-/-	-/95.0	-/-	-/-	-/-	-/-	-/96.0	-/-
Chen et al. [5]	-/-	94.5/-	-/-	-/-	-/-	-/-	96.7/-	-/-
Catillo et al. [3]	-/-	-/-	-/-	-/-	97.0/-	-/-	95.0/-	-/-
Sudqi et al. [32]	-/-	-/-	-/-	-/-	-/-	94.9/-	-/-	-/-

** Results marked with asterisks are computed with only 5% of the total number of sequences due to computational constraints.*

4.3 Results

This section summarizes the detection results achieved on the selected data sets using sequence identifiers, time-windows, and simple events as grouping strategies.

4.3.1 Detection using Sequence-identifier Groups. We present the F1 scores of 25 runs with randomly sampled and 1 run of chronologically sampled training sequences in Table 2, which we obtained by applying aforementioned detection techniques and combinations on the data sets (since several versions of the data sets are available, we reference the paper or source for the specific version in brackets). Highest average scores achieved for each data set are in bold. Aligned with previous studies [23], we find that chronological sorting deteriorates detection performance; this is due to the fact that it yields less diverse training samples than random sorting. For example, the training sample of the chronologically sorted HDFS contains only similar sequence lengths, leading to a low F1 score due to many false positives. In the bottom of the table, we also provide some benchmark results that have been reported in state of the art surveys and publications on semi-supervised anomaly detection. We point out that comparability is limited as these works use different splits of training and test data sets and often only report the best scores achieved from multiple runs [5].

For the HDFS data set, we use both the pre-processed version from the LogDeep [1] repository as well as the original data set provided by Xu et al. [38] since the obtained results do not coincide. In particular, the combination of detection based on new events and sequence lengths yields an F1 score of 90.4% on the LogDeep version, but only an average of 72.0% on the original data set with randomly selected sequences. A closer inspection of the reason for this peculiarity is that the training sequences in the LogDeep version of the HDFS data set do not involve event type 20, which is an indicator for anomalies (cf. Sect. 3.1) but also occurs in 219 (0.04%) of the normal sequences and thus deteriorates detection performance if randomly drawn into the set of training sequences. This effect shows the importance of repeating evaluations with multiple randomly drawn samples to obtain representative and comparable results.

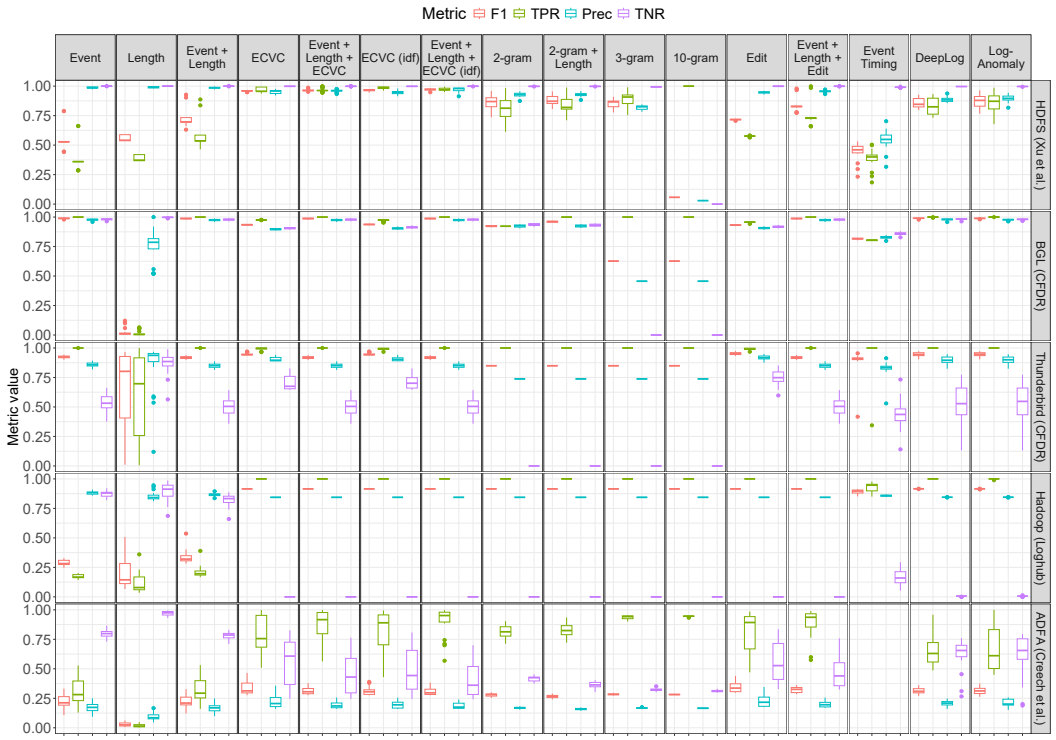

Fig. 10. Evaluation results for sequence-based detection on HDFS, BGL, Thunderbird, Hadoop, and ADFA.

Figure 10 provides boxplots for a more detailed view on the results that also includes TPR (identical to recall), precision, and TNR in addition to the F1 score. The plot confirms that for the HDFS data set, more than half of all anomalies are very simple to detect based on the combination of new event types and sequence lengths. ECVC detection further improves the scores and achieves competitive performance compared to results reported in literature (cf. Table 2). Another interesting observation is that ECVC generally yields better scores compared to edit distance and N-gram based detection, which indicates that event position in sequences is less relevant for detection than event occurrence frequencies, and that random event orders due to simultaneous occurrence have an adverse effect on the detection rates. IDF weighting shows a positive influence on ECVC, which is reasonable as relevant event types such as event type 20 receive a higher weight compared to event types that occur in many sequences and are not related to anomalies.

Even though DeepLog and LogAnomaly yield the highest F1 scores on the randomly sampled BGL data set, they only surpass simple event-based detection by a negligible margin of 0.1%. The effectiveness of event-based detection is intuitively reasonable, since the normal and anomalous event types are almost completely disjoint sets as stated in Sect. 3.2. A sample size of 1% appears sufficient to train the detector for almost all normal event types and avoid many false alarms. The best results for the Thunderbird data set are achieved by edit distance detection, but ECVC, LogAnomaly, DeepLog, and event based detection only fall short by a small margin. This indicates that event order is a relevant factor when discerning normal from anomalous sequences in this data set, but again emphasizes that anomalies are primarily linked to new event types.

Regarding the Hadoop data set, the results for F1 (91.5%), TPR (100%), and precision (84.4%) indicate adequate detection performance at first glance. However, TNR of 0% reveals that in fact the

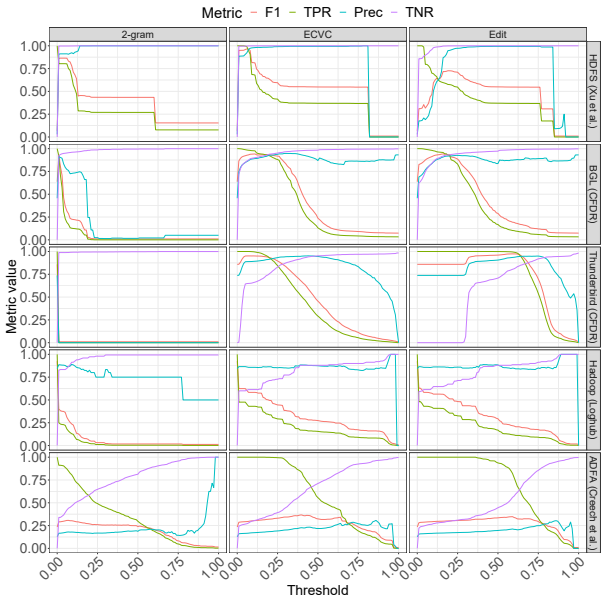

Fig. 11. Influence of detection thresholds on the metrics.

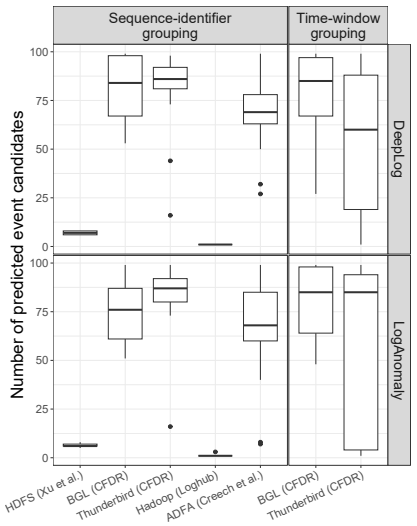

Fig. 12. Optimal number of predicted event candidates for deep learning approaches.

detector just reports every single sequence as anomalous and thus the detection results are of no practical value. Clearly, no detection technique is able to yield good results, because many normal and anomalous sequences are identical as described in Sect. 3.5. This demonstrates the importance of considering metrics such as TNR in addition to F1, precision, and recall, to avoid misleading results in imbalanced data sets such as Hadoop where 82,9% of all sequences are anomalous [23].

The ADFA data set appears to be more challenging in comparison to other data sets. Specifically, the achieved precision is relatively low across all detection techniques and there is no significant difference between techniques that leverage sequence ordering and those that focus on event frequencies. Note that log events do not include timestamps and thus detection based on event timing is omitted from the plot. We suggest this data set as a useful candidate for future evaluations of sequence-based anomaly detection techniques that improve upon our baseline results.

As stated in Sect. 4.1, we optimized the results in the previous section by fine-tuning the detection thresholds to maximize F1. In practice, however, such a fine-tuning is not feasible due to the lack of a ground truth. We therefore investigate the parameter influence on the results of 2-gram, ECVC, and edit distance detection, by iterating the threshold in the range 0 to 1 in steps of 0.01. Figure 11 shows the progression of evaluation metrics for each combination of data set and detection technique from a single evaluation run. For the HDFS data set, ECVC is clearly the most preferable detection technique and yields high F1 scores for thresholds lower than 0.1. Both ECVC and 2-gram achieve nearly perfect TNR for any thresholds, indicating that almost all normal sequences are straightforward to classify as such. Regarding the BGL data set, both ECVC and edit distance perform well for thresholds smaller than 0.25 and yield high precision independent from the threshold, while 2-gram detection shows a much narrower band where adequate results are achieved. On the Thunderbird data set, no true positives are detected by the 2-gram method for any threshold larger than 0, because a few very dissimilar sequences cause that almost all anomaly scores are close to 0 after normalizing them across all sequences. ECVC and edit distance detection techniques are not affected by this problem as their anomaly scores are normalized per sequence rather than across all sequences; accordingly, both techniques yield better results than 2-gram detection.

The results obtained for the Hadoop data set provide more insights into the problems arising from a dominating anomaly class that we mentioned in the previous section. Specifically, the highest F1 scores are achieved when the threshold is 0 and all instances are detected as anomalous, even though TNR is 0% at that point. While TNR increases for higher thresholds, the F1 score decreases as TPR drops. All detectors yield comparatively low precision on the ADFA data set across all thresholds; 2-gram detection has high precision for high thresholds, but this is only because few true positives are found. Overall, the diversity of the plots suggests that parameter tuning needs to be carried out on each data set separately to optimize results.

As visible in Table 2 and Fig. 10, LogAnomaly, which leverages event count vectors in addition to sequences, generally outperforms DeepLog, which relies solely on sequential patterns. Neither of these deep learning approaches is able to outperform simple detection techniques by more than a small margin; their F1 scores are usually in a similar range as those obtained from the best performing simple detection techniques. We also investigate the influence of number of predicted event candidates, which we select to maximize the F1 score (cf. Sect. 4.1). The corresponding boxplots depicted in Fig. 12 show that a value of around 8 candidates appears as a stable choice for HDFS. BGL, Thunderbird, and ADFA, on the other hand, work best with significantly larger candidate sizes; this is reasonable as these data sets involve more distinct event types, but also suggests that the actual event type is often not among the ones predicted to occur with highest probability. Moreover, in BGL and Thunderbird, almost all anomalies involve new event types, which are never predicted; thus, large candidate sizes do not negatively affect TPR.

4.3.2 Detection using Time-window Groups. The right-hand columns of Table 2 present the detection results obtained from the BGL and Thunderbird data sets using time-window grouping, where we select a window size of 30 minutes following the work by Le et al. [23]. In both data sets, simple event detection performs comparatively well and achieves similar F1 scores as deep learning detectors. Figure 13 provides a detailed view on all relevant metrics and reveals that TNR is generally low for the Thunderbird data set, reaching only around 35% by event-based detection and dropping to 0% for several other techniques, including detectors leveraging deep learning. Due to the high fraction of anomalous sequences, F1 scores are still mostly around 75%-85%. In the BGL data set, detection techniques that already performed comparatively poor when using sequence-identifier grouping again yield the lowest scores, in particular, techniques based on sequence lengths, N-grams, and event timing have comparatively low TPR.

Most authors report F1 scores of more than 90% when using time-window grouping on BGL. We noticed that the chosen size of the time-window varies significantly in these works, ranging from 5 minutes [9] up to six hours [5]. We thus apply event-based detection on BGL and Thunderbird data sets for various time-window sizes. Figure 14 shows that the progression of our metrics exhibits similar patterns for BGL and Thunderbird, where each set of observations is captured during a single evaluation run. Event-based detection yields TPR of almost 100% independent from the time-window size; this is due to the fact that almost all anomalous sequences are easily identified through anomalous events that do not occur in normal sequences. For small time-windows, many sequences comprising only few events are formed and detected with high precision. As time-window sizes increase, precision first declines since false positives increase relative to true positives. However, this effect reverses when sequences become long enough so that most of them contain at least one anomalous event, rendering a majority of the available sequences anomalous. At this point, true positives again start to outnumber false positives, leading to an increase of precision and F1 score, while TNR approaches 0%. We conclude that small time-windows work best for event-based detection as larger time-windows lead to many false positives or cause problematic data imbalance.

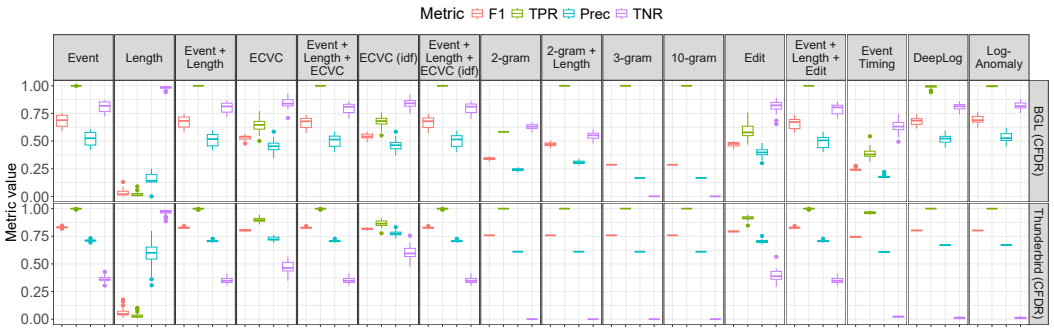

Fig. 13. Evaluation results for detection based on time-windows on BGL and Thunderbird.

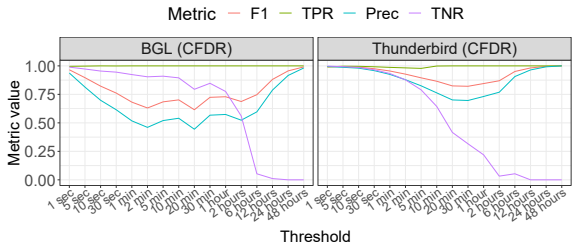

Fig. 14. Influence of time-window size on the metrics.

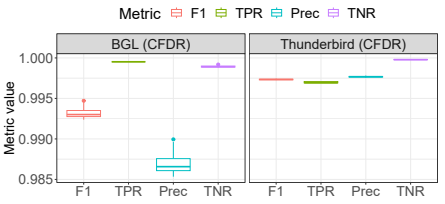

Fig. 15. Evaluation results for event-based detection on BGL and Thunderbird data sets.

4.3.3 Event-based Detection. The results from the previous section suggest that event-based detection performs better for shorter sequences in the BGL and Thunderbird data sets. We thus consider the shortest possible sequences: single events. Note that this evaluation is only feasible on these two data sets, because labels are available for events rather than pre-defined sequences.

Since almost all anomalous events belong to different types than normal events (cf. Sect. 3.2 and Sect. 3.3), it is generally easy to detect anomalies. However, false positives may be problematic in case that the training data set is not large enough to cover all normal events, which are subsequently reported in the detection phase. However, Fig. 15 shows that very high detection scores and TNR of around 99.9% are achieved when training data is randomly sampled in both data sets, indicating that 1% is a sufficiently large training sample for this type of detection. When the first 1% of chronologically sorted events are used, F1 scores drop to 29.1% and 40.9% for BGL and Thunderbird respectively due to many false positives.

5 DISCUSSION

This section contains the discussion of our analysis and evaluation results. We first focus on the data sets themselves before moving to the way the data is used in evaluations. We end this section with some recommendations for future work.

5.1 Appropriateness of Data Sets

To be suitable for anomaly detection evaluations, data sets need to meet characteristics that fit the type of detection. Given that the data sets analyzed in this study are widely used for the evaluation of deep learning anomaly detection techniques that ingest the data as sequences of event types and possibly event parameters, one would expect that changes in sequential patterns is exactly how the anomalies manifest in these data sets. Specifically, most approaches rely on inherent sequential dependencies of event types as expected event occurrences are predicted using a set of preceding

Table 3. Anomaly manifestations and identified issues in commonly used data sets

Data set	Anomaly types	Version	Issues
HDFS	- New event types - Sequence lengths - Event counts	Xu et al. [38]	- The majority of anomalies is straightforward to detect. - Some sub-sequences are randomly ordered, increasing sequence complexity.
		LogDeep [1]	- Same as for the original data set published by Xu et al. [38] - Training sample is not representative and yields excessively high detection rates. - Timestamps for events cannot be retrieved as sequence identifiers are missing.
		Loghub [12]	- Same as for the original data set published by Xu et al. [38] - Some lines are missing for unknown reasons.
BGL	- New event types	CFDR [27]	- Anomalies only affect certain event types and can be detected without sequences. - Sometimes analyzed with sliding windows despite availability of sequence identifiers.
		Loghub [12]	- Same as for the original data set provided by CFDR [27]. - Some lines have been relabeled for unknown reasons.
Thunderbird	- New event types	CFDR [27]	- Anomalies only affect certain event types and can be detected without sequences.
OpenStack	- None	Loghub [12]	- Anomalies do not manifest as changes of event sequences or event inter-arrival times.
Hadoop	- None (majority) - New event types	Loghub [12]	- The majority of anomalies do not manifest as changes of event sequences.

event types [7, 19]. However, the results of our study suggest that the link between anomalies and sequential patterns is less pronounced than expected.

We thus answer our research questions as follows. *How do anomalies manifest themselves in common log data sets?* Anomalies primarily manifest through new event types, changes of event frequencies, and sometimes sequence lengths; sequential patterns are less relevant than expected. *What are drawbacks that render these data sets inadequate for evaluation of sequence-based anomaly detection techniques?* Drawbacks include high fractions of anomalies that are straightforward to detect, randomly ordered events that complicate sequential event analysis, identical event sequences that occur in both normal and anomalous classes, and class imbalance. A detailed mapping of anomaly manifestations as well as drawbacks for each data set is stated in Table 3.

The HDFS data set, which is the most popular data set in this research area [19], involves anomalies that manifest as new event types that do not occur within normal instances, unusually short sequence lengths, and event occurrence frequencies that do not require ordered sequences. In fact, random permutations of events that are generated simultaneously and not related to anomalies complicate sequence-based anomaly detection compared to other analysis techniques that are robust against permutations, such as count vector clustering. Another indicator that sequential patterns are less relevant in the HDFS data set than expected is that the ground truth of that data set was generated by clustering unordered count vectors [38].

Our study further shows that a majority of the anomalies are straightforward to identify even for very simple detection approaches, which are capable of achieving competitive detection rates on the HDFS data set. While approaches such as Deeplog [7] and LogAnomaly [26] apply advanced detection models, a major part of their correctly detected instances thus implicitly relies on simple detection of new events [23] or detection of short sequences as a result of padding, i.e., augmenting short sequences with additional event types that make them more likely to be detected [14]. Unfortunately, this means that evaluation metrics reported on the HDFS data set give the misleading impression that a majority of anomalies are disclosed by the complex sequence-based detection techniques, even though they make up a much smaller share.

Anomalies in the BGL and Thunderbird data sets primarily manifest as events corresponding to types that never occur in normal data. Accordingly, for the simple task of identifying single events as anomalies, it is not required to analyze logs as sequences. In addition, grouping log data into sequences is often carried out using sliding windows [5], which is problematic when different processes interleave [19]. However, due to the length and complexity of these data sets, they appear well suited for many types of log analysis, such as automatic parser generation [21] or word embedding to compensate evolution of log messages [41].

Both Hadoop and OpenStack data sets involve a high fraction of identical event sequences in normal and anomalous classes. No other artifacts suitable for anomaly detection in these data sets were identified in course of this study. We therefore advise against using these data sets for evaluation of anomaly detection techniques.

Our study suggests that the ADFA data set is a promising alternative to aforementioned data sets, because anomalies are not primarily detectable by simple techniques such as new events or sequence lengths. In course of our evaluation study we were able to determine that some of the anomaly classes are easier to detect than others as suggested by related work [35]. Moreover, system call logs form ordered sequences that only involve discrete event types, which means that the composition of parsing templates has no influence on detection performance. However, we point out that our experiments are not suitable to confirm that the anomalies in the ADFA data set indeed manifest as changes of sequential patterns, which is a task we leave for future work.

Just because a data set is widely used in scientific publications does not necessarily mean that it is automatically a good choice. When Creech et al. [6] published the ADFA data set in 2013, they attempted to replace data sets such as KDD99 that were already criticized and considered outdated at that time [31]. Nonetheless, KDD99 was still widely used by the research community many years later [17]. This shows that researchers are drawn towards data sets that are convenient to use (e.g., because they are labeled, sufficiently large, or pre-processed) and accepted as a benchmark data set by the community, even though they are not ideal for evaluations. We thus expect that despite our findings, data sets such as HDFS will continue to be used until superior alternatives are proposed.

5.2 Appropriateness of Evaluations

A major issue with most published evaluations is that the results are hardly reproducible. Since many authors do not publish code, parameters settings, and data in a way that enables others to recreate the presented results, it is difficult to obtain an accurate overview of the detection capabilities achieved by state of the art approaches. Even though some authors re-implement well-known models [5, 23], the evaluation results hardly ever align across multiple papers even though the same detection techniques are applied on the same data sets.

One of the contributing factors for this issue is that authors usually fine-tune model parameters and repeat evaluation runs multiple times only to report the best results [5]. Unfortunately, this makes it difficult to comprehend the variance of the detection scores and the influence of selected thresholds. In addition, we observed that splits between training and test data vary strongly across different evaluations, including 1% [7], 20% [3], 50% [10], and 80% [5, 23].

Another issue is that some papers only measure precision and recall to compute the F1 score [9], but omit false positive rate or true negative rate. Given that anomaly detection data sets are highly imbalanced, it is important to consider either of these metrics to avoid misinterpretation of results as we demonstrate on the Hadoop data set in Sect. 4.3.1.

Anomaly detection techniques that leverage deep learning and neural networks generally have a lower explainability than conventional machine learning methods [34]. Unfortunately, this also means that the reasons why instances are reported as anomalies or not can often go unnoticed. Accordingly, it is important to come up with evaluation methodologies and fitting data sets that are capable of demonstrating the advantages of advanced models in a clear way, in particular, to justify the higher runtime and computational effort in contrast to conventional machine learning [5].

5.3 Recommendations & Future Work

Based on the problems we identified in the previous sections, we formulate a set of recommendations to be addressed by future works. *(i) Create new data sets that specifically support evaluation of sequence-based detectors.* System call logs such as the ADFA log data set appear beneficial as they

are ordered, easy to group into sequences, avoid the need for parsing, and enable anomaly injection by executing adverse functions on the host where logs are collected. *(ii) Repeat evaluation runs multiple times and report scores with variances.* Presenting only the results from the best run with fine-tuned parameters causes that detection capabilities appear better than they are. *(iii) Ensure that data sets are suitable for evaluation.* In particular, the way anomalies manifest in the data should be verified and explained beforehand. *(iv) Use simple detection techniques fitting to the anomaly manifestations as baselines. (v) Ensure reproducibility of reported results.* This involves publishing all code and data that is necessary to repeat the conducted experiments and confirm the presented results. Moreover, relevant settings such as sampling strategies, splits between training and test data sets, and model parameters should be analyzed for their respective influence.

Our simple detection methods used in this paper focus on event occurrences, i.e., event timestamps are the only contextual information derived from the logs other than sequences of events. However, anomalies often manifest in event parameters, and incorporating them in the detection procedure is thus a reasonable approach. For example, simple parameter-based detection could leverage new value occurrences in categorical event parameters similar to our detection of new event types.

Finally, our study focuses on semi-supervised anomaly detection, where only normal data is used for training. However, supervised classification of anomalies is also an actively researched field that leverages the data sets used in this paper [19]. We therefore plan to adopt our simple detection techniques for supervised anomaly detection and classification of anomalies into their respective classes. As this is considered out of scope for this paper, we leave this task for future work.

6 CONCLUSION

Quality and appropriateness of data sets are crucial for sound and representative evaluations of anomaly detection techniques. In this paper, we analyzed five log data sets that are commonly used in state of the art and one additional data set from security research to determine whether they are suitable for evaluation of sequence-based detection algorithms. While these algorithms are primarily designed to recognize changes of sequential patterns, such as log event types generated in different order than during normal operation, our analysis suggests that these artifacts hardly occur as part of anomalies. In the HDFS data set, for example, shuffled sub-sequences in event executions result from simultaneously generated events rather than anomalies. Moreover, anomalous sequences sometimes do not differ from normal behavior at all, rendering some data sets unusable. We tested the most suitable data sets with simple detectors for new events, sequence lengths, count vector similarity, sub-sequence similarity, and event timing. Our results suggest that these techniques achieve competitive detection rates compared to advanced state of the art approaches, further indicating that high detection rates are easy to achieve in some data sets. To counteract these issues, we recommend to work on new data sets that are specifically designed to include sequential anomalies and improve evaluation methodologies to avoid that misleading results are obtained.

DATA AVAILABILITY

Data and code are available at https://github.com/ait-aecid/anomaly-detection-log-datasets or [18].

ACKNOWLEDGMENTS

The work in this paper has received funding from the European Union - European Defence Fund under GA no. 101103385 (AInception) and GA no. 101121403 (NEWSROOM), and from the Austrian Research Promotion Agency (FFG) under GA no. FO999899544 (PRESENT). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. The European Union cannot be held responsible for them.

REFERENCES

[1] Donglee Afar. 2020. LogDeep: Open-source Log Anomaly Detection Toolkit. https://github.com/d0ng1ee/logdeep.

[2] Charu C Aggarwal, Alexander Hinneburg, and Daniel A Keim. 2001. On the surprising behavior of distance metrics in high dimensional space. In *Proceedings of the Database 8th International Conference on Database Theory*. Springer, 420–434. https://doi.org/10.1007/3-540-44503-X_27

[3] Marta Catillo, Antonio Pecchia, and Umberto Villano. 2022. AutoLog: Anomaly detection by deep autoencoding of system logs. *Expert Systems with Applications* 191 (2022), 116263. https://doi.org/10.1016/j.eswa.2021.116263

[4] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM computing surveys* 41, 3 (2009), 1–58. https://doi.org/10.1145/1541880.1541882

[5] Zhuangbin Chen, Jinyang Liu, Wenwei Gu, Yuxin Su, and Michael R Lyu. 2021. Experience report: Deep learning-based system log analysis for anomaly detection. *arXiv preprint arXiv:2107.05908* (2021). https://doi.org/10.48550/arXiv.2107.05908

[6] Gideon Creech and Jiankun Hu. 2013. Generation of a new IDS test dataset: Time to retire the KDD collection. In *Proceedings of the IEEE Wireless Communications and Networking Conference*. IEEE, 4487–4492. https://doi.org/10.1109/WCNC.2013.6555301

[7] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. 2017. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. 1285–1298. https://doi.org/10.1145/3133956.3134015

[8] Stephanie Forrest, Steven A Hofmeyr, Anil Somayaji, and Thomas A Longstaff. 1996. A sense of self for unix processes. In *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE, 120–128. https://doi.org/10.1109/SECPRI.1996.502675

[9] Haixuan Guo, Shuhan Yuan, and Xintao Wu. 2021. Logbert: Log anomaly detection via bert. In *Proceedings of the International Joint Conference on Neural Networks*. IEEE, 1–8. https://doi.org/10.1109/IJCNN52387.2021.9534113

[10] Shayan Hashemi and Mika Mäntylä. 2021. OneLog: Towards end-to-end training in software log anomaly detection. *arXiv preprint arXiv:2104.07324* (2021). https://doi.org/10.48550/arXiv.2104.07324

[11] Pinjia He, Jieming Zhu, Zibin Zheng, and Michael R Lyu. 2017. Drain: An online log parsing approach with fixed depth tree. In *Proceedings of the IEEE international conference on web services*. IEEE, 33–40. https://doi.org/10.1109/ICWS.2017.13

[12] Shilin He, Jieming Zhu, Pinjia He, and Michael R Lyu. 2020. Loghub: a large collection of system log datasets towards automated log analytics. *arXiv preprint arXiv:2008.06448* (2020). https://doi.org/10.48550/arXiv.2008.06448

[13] Yingying He, Xiaobing Pei, and Lihong Shen. 2024. Semi-supervised learning via DQN for log anomaly detection. *arXiv preprint arXiv:2401.03151* (2024). https://doi.org/10.48550/arXiv.2401.03151

[14] Patrick Himler, Max Landauer, Florian Skopik, and Markus Wurzenberger. 2023. Towards Detecting Anomalies in Log-Event Sequences with Deep Learning: Open Research Challenges. In *European Interdisciplinary Cybersecurity Conference*. 71–77. https://doi.org/10.1145/3590777.3590789

[15] Parisa Sadat Kalaki, Alireza Shameli-Sendi, and Behzad Khalaji Emamzadeh Abbasi. 2023. Anomaly detection on OpenStack logs based on an improved robust principal component analysis model and its projection onto column space. *Software: Practice and Experience* 53, 3 (2023), 665–681. https://doi.org/10.1002/spe.3164

[16] Anthony Kenyon, Lipika Deka, and David Elizondo. 2020. Are public intrusion datasets fit for purpose characterising the state of the art in intrusion event datasets. *Computers & Security* 99 (2020), 102022. https://doi.org/10.1016/j.cose.2020.102022

[17] Ansam Khraisat, Iqbal Gondal, Peter Vamplew, and Joarder Kamruzzaman. 2019. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity* 2, 1 (2019), 1–22. https://doi.org/10.1186/s42400-019-0038-7

[18] Max Landauer. 2024. *ait-aecid/anomaly-detection-log-datasets: Zenodo DOI*. https://doi.org/10.5281/zenodo.11528723

[19] Max Landauer, Sebastian Onder, Florian Skopik, and Markus Wurzenberger. 2023. Deep learning for anomaly detection in log data: A survey. *Machine Learning with Applications* 12 (2023), 100470. https://doi.org/10.1016/j.mlwa.2023.100470

[20] Max Landauer, Florian Skopik, Georg Höld, and Markus Wurzenberger. 2022. A User and Entity Behavior Analytics Log Data Set for Anomaly Detection in Cloud Computing. In *Proceedings of the IEEE International Conference on Big Data*. IEEE, 4285–4294. https://doi.org/10.1109/BigData55660.2022.10020672

[21] Max Landauer, Florian Skopik, Markus Wurzenberger, and Andreas Rauber. 2020. System log clustering approaches for cyber security applications: A survey. *Computers & Security* 92 (2020), 101739. https://doi.org/10.1016/j.cose.2020.101739

[22] Max Landauer, Markus Wurzenberger, Florian Skopik, Wolfgang Hotwagner, and Georg Höld. 2023. Aminer: A modular log data analysis pipeline for anomaly-based intrusion detection. *Digital Threats: Research and Practice* 4, 1 (2023), 1–16. https://doi.org/10.1145/3567675

[23] Van-Hoang Le and Hongyu Zhang. 2022. Log-based anomaly detection with deep learning: How far are we?. In *Proceedings of the 44th International Conference on Software Engineering*. 1356–1367. https://doi.org/10.1145/3510003.3510155

[24] Abraham Lempel and Jacob Ziv. 1976. On the complexity of finite sequences. *IEEE Transactions on Information Theory* 22, 1 (1976), 75–81. https://doi.org/10.1109/TIT.1976.1055501

[25] Qingwei Lin, Hongyu Zhang, Jian-Guang Lou, Yu Zhang, and Xuewei Chen. 2016. Log clustering based problem identification for online service systems. In *Proceedings of the 38th International Conference on Software Engineering Companion*. 102–111. https://doi.org/10.1145/2889160.2889232

[26] Weibin Meng, Ying Liu, Yichen Zhu, Shenglin Zhang, Dan Pei, Yuqing Liu, Yihao Chen, Ruizhi Zhang, Shimin Tao, Pei Sun, et al. 2019. LogAnomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. In *Proceedings of the International Joint Conference on Artificial Intelligence*, Vol. 19. 4739–4745.

[27] Adam Oliner and Jon Stearley. 2007. What supercomputers say: A study of five system logs. In *Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 575–584. https://doi.org/10.1109/DSN.2007.103

[28] Bianca Schroeder and Garth Gibson. 2024. The Computer Failure Data Repository (CFDR). https://www.usenix.org/cfdr.

[29] Hinrich Schutze, Christopher D Manning, and Prabhakar Raghavan. 2008. *Introduction to information retrieval*. Cambridge University Press.

[30] Weiyi Shang, Zhen Ming Jiang, Hadi Hemmati, Brain Adams, Ahmed E Hassan, and Patrick Martin. 2013. Assisting developers of big data analytics applications when deploying on hadoop clouds. In *Proceedings of the 35th International Conference on Software Engineering*. IEEE, 402–411. https://doi.org/10.1109/ICSE.2013.6606586

[31] Kamran Siddique, Zahid Akhtar, Farrukh Aslam Khan, and Yangwoo Kim. 2019. KDD cup 99 data sets: A perspective on the role of data sets in network intrusion detection research. *Computer* 52, 2 (2019), 41–51. https://doi.org/10.1109/MC.2018.2888764

[32] Belal Sudqi Khater, Ainuddin Wahid Bin Abdul Wahab, Mohd Yamani Idna Bin Idris, Mohammed Abdulla Hussain, and Ashraf Ahmed Ibrahim. 2019. A lightweight perceptron-based intrusion detection system for fog computing. *Applied Sciences* 9, 1 (2019), 178. https://doi.org/10.3390/app9010178

[33] Narate Taerat, Nichamon Naksinehaboon, Clayton Chandler, James Elliott, Chokchai Leangsuksun, George Ostrouchov, Stephen L Scott, and Christian Engelmann. 2009. Blue gene/l log analysis and time to interrupt estimation. In *Proceedings of the International Conference on Availability, Reliability and Security*. IEEE, 173–180. https://doi.org/10.1109/ARES.2009.105

[34] Konrad Wolsing, Lea Thiemt, Christian van Sloun, Eric Wagner, Klaus Wehrle, and Martin Henze. 2022. Can Industrial Intrusion Detection Be SIMPLE?. In *Proceedings of the 27th European Symposium on Research in Computer Security*. Springer, 574–594. https://doi.org/10.1007/978-3-031-17143-7_28

[35] Miao Xie, Jiankun Hu, Xinghuo Yu, and Elizabeth Chang. 2014. Evaluating host-based anomaly detection systems: Application of the frequency-based algorithms to ADFA-LD. In *Proceedings of the 8th International Conference on Network and System Security*. Springer, 542–549. https://doi.org/10.1007/978-3-319-11698-3_44

[36] Wei Xu, Ling Huang, Armando Fox, David Patterson, and Michael Jordan. 2009. Online system problem detection by mining patterns of console logs. In *Proceedings of the 9th IEEE International Conference on Data Mining*. IEEE, 588–597. https://doi.org/10.1109/ICDM.2009.19

[37] Wei Xu, Ling Huang, Armando Fox, David Patterson, and Michael I Jordan. 2009. Detecting large-scale system problems by mining console logs. In *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles*. 117–132. https://doi.org/10.1145/1629575.1629587

[38] Wei Xu, Ling Huang, Armando Fox, David A Patterson, and Michael I Jordan. 2008. Mining Console Logs for Large-Scale System Problem Detection. *SysML* 8 (2008), 4–4.

[39] Rakesh Bahadur Yadav, P Santosh Kumar, and Sunita Vikrant Dhavale. 2020. A survey on log anomaly detection using deep learning. In *Proceedings of the 8th International Conference on Reliability, Infocom Technologies and Optimization*. IEEE, 1215–1220. https://doi.org/10.1109/ICRITO48877.2020.9197818

[40] Lin Yang, Junjie Chen, Zan Wang, Weijing Wang, Jiajun Jiang, Xuyuan Dong, and Wenbin Zhang. 2021. Semi-supervised log-based anomaly detection via probabilistic label estimation. In *Proceedings of the IEEE/ACM 43rd International Conference on Software Engineering*. IEEE, 1448–1460. https://doi.org/10.1109/ICSE43902.2021.00130

[41] Xu Zhang, Yong Xu, Qingwei Lin, Bo Qiao, Hongyu Zhang, Yingnong Dang, Chunyu Xie, Xinsheng Yang, Qian Cheng, Ze Li, et al. 2019. Robust log-based anomaly detection on unstable log data. In *Proceedings of the 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 807–817. https://doi.org/10.1145/3338906.3338931

Received 2023-09-28; accepted 2024-04-16