RESEARCH Open Access

An online log template extraction method based on hierarchical clustering

Ruipeng Yang[1,2*], Dan Qu[1], Yekui Qian[2*], Yusheng Dai[3] and Shaowei Zhu[2]

Abstract

The raw log messages record extremely rich system, network, and application running dynamic information that is a good data source for abnormal detection. Log template extraction is an important prerequisite for log sequence anomaly detection. The problems of the existing log template extraction methods are mostly offline, and the few online methods have insufficient F1-score in multi-source log data. In view of the shortcomings of the existing methods, an online log template extraction method called LogOHC is proposed. Firstly, the raw log messages are preprocessed, and the word distributed representation (word2vec) is used to vectorize the log messages online. Then, the online hierarchical clustering algorithm is applied, and finally, log templates are generated. The experimental analysis shows that LogOHC has a higher F1-score than the existing log template extraction methods, is suitable for multi-source log data sets, and has a shorter single-step execution time, which can meet the requirements of online real-time processing.

Keywords: Log template extraction, Log anomaly detection, Online vectorization, Online hierarchical clustering

1 Introduction

The network environment is increasingly complex, and attacks against network applications and different systems are constantly emerging and are often combined with multiple attack methods. Once the attack succeeds or the network application itself is abnormal, it will bring immeasurable losses to the owners and users of the application. The earlier the attack and error are discovered, the less damage it will cause. Therefore, anomaly detection has caused extensive attention from the academia. Current anomaly detection data sources include malware and traffic, but they all have their shortcomings [1–3].

Networks, systems, and applications generate various types of log data during running that are used to record the status of networks, systems, and applications, as well as important events. Therefore, log data contain extremely rich network operational dynamic information that can be used for anomaly detection [4–9], discover and diagnose performance problems [10], and find software bugs [11]. Because the log-based anomaly detection method has the characteristics of an accurate analysis of attack problems and reconfigurability of attack chains, it is increasingly becoming the mainstream method for detecting abnormal behavior of network or system. On the other hand, the log data has the characteristics of large data volume, heterogeneity, and unstructured, which poses great challenges for the analysis.

The log is sequence data, generated by the corresponding template of log print statements in the source code. A template reflects a type of event, but we do not know the code beforehand. The log sequence records the events that occur in the system. Inspired by natural language processing, each log message can be viewed as a sentence. The log-based anomaly detection can be regarded as a text prediction to some extent. However, the log data has a large volume. Generally, templates are extracted from the log messages firstly. The log-based anomaly detection can be simplified to the abnormal detection for the log sequence, that is, the abnormal detection for the log template sequence. So log template extraction is an important prerequisite for log-based anomaly detection, and it is highly valued by the academia. Traditionally, log template extraction mainly relied on regular expressions, which are designed by analysts. But it is too much subject to the log format and so manually. In recent years, log template extraction

* Correspondence: yangruipeng-@163.com; qyk1129@163.com
[1]PLA Strategic Support Force Information Engineering University, Zhengzhou, China
[2]No. 24, East Jianshe Road, Zhengzhou, China
Full list of author information is available at the end of the article

is widely studied to parse the raw log messages automatically [12–20]. However, these methods relied on the log format and most of them were offline, which did not meet the real-time requirements for log analysis. In response to this problem, Du and Li [13] and He et al. [14] proposed Spell and Drain for online log template extraction. Spell extracted the log templates online based on the idea of the longest common subsequence matching and solved the online extraction problem of the log templates. Drain parsed log messages in a streaming mode and further improved the accuracy and running time. However, the existing online log template extraction methods were not accurate and efficient enough. More importantly, they did not analyze their applicability to multi-source logs, which had a different log format, while this requirement was often present in practical applications.

In view of the shortcomings of the existing methods, this paper proposes an online log template extraction method, namely, LogOHC. Firstly, the raw log messages are preprocessed, and the word distributed representation (word2vec) is used to vectorize the log messages online. Then, the online hierarchical clustering algorithm is applied, and finally, log templates are generated. The process of vectorization is word embedding, which is inspired by natural language processing. It is not limited by the log format, and suitable for multi-source heterogeneous log data. We evaluate LogOHC on real-world log data sets. It demonstrates a higher F1-score and a shorter single-step execution time, which can fully meet the needs of online log analysis.

The main contributions and innovations of this paper are summarized as the following:

(1) On the basis of data preprocessing, this paper vectorizes the log message online using the mean value word2vec algorithm to provide a high-quality data source for online hierarchical clustering. It applies the idea of natural language processing to log processing, which is not subject to the log format;

(2) This paper proposes an online log template extraction method based on online hierarchical clustering, meeting the needs of online processing of log data;

(3) Sufficient experiments have been conducted on three real-world log data sets. Three state-of-the-art methods are used to automatically extract log templates as the competing baseline. Our study shows that compared with state-of-the-art methods, LogOHC outperforms them in terms of effectiveness and also has the superiority in efficiency and the sensitivity of the parameters.

The remainder of this paper is organized as follows: Section 2 provides preliminary and background, Section 3 presents the LogOHC log template extraction method, Section 4 conducts the experimental evaluation, and Section 5 summarizes the full paper and proposes further research directions.

2 Preliminary and background

2.1 Problem description

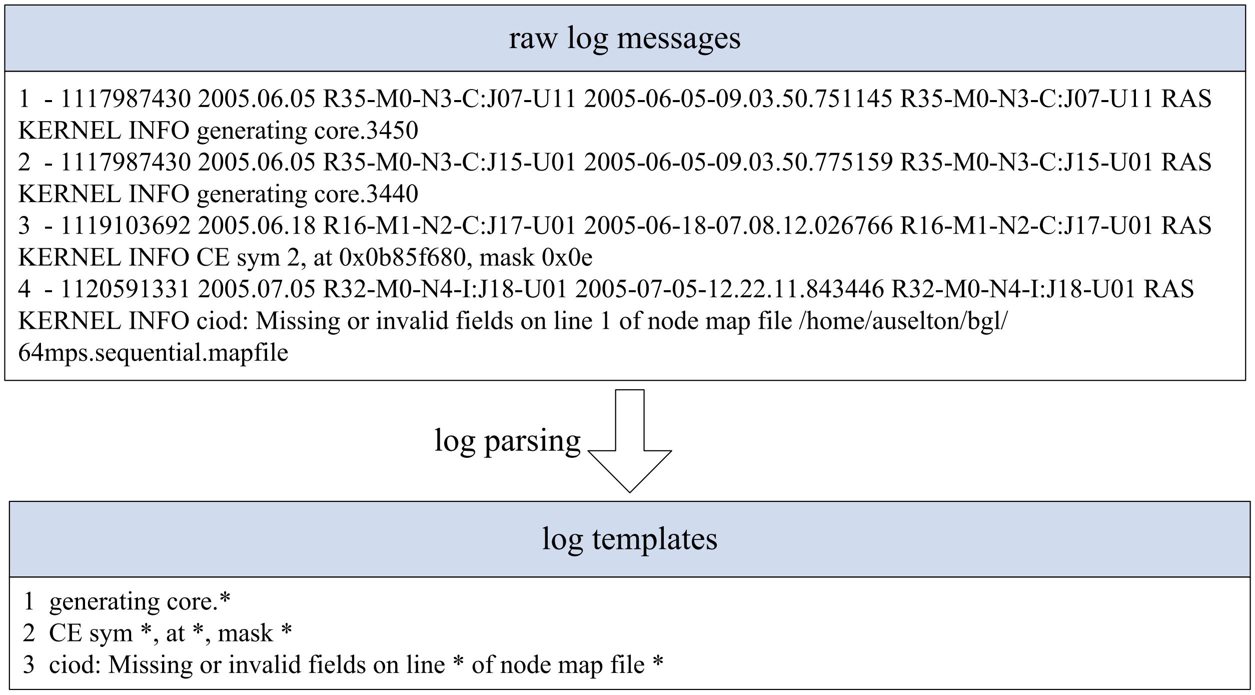
For the convenience of the expression of the following chapters, we define a few terms: the raw log messages refer to the event records that are generated by the network device, system, and service program during operation. A log message (or event) refers to a single line in the raw log messages after preprocessing. A log template is defined as the description for events in the same log group with a similar structure.

The raw log message structure contains two parts: the log template and the parameter value. The log template extraction is a process of continuously clustering the raw log messages. Figure 1 lists four raw log messages, INFO is an event type in the system log, and the content after INFO describes the events that occur in the system. The raw log messages are generated by a certain template format, although we do not know the specific template in advance. For example, the first and second raw log messages have a common template: generating core.*. "3450" and "3440" are parameter values. The four raw log messages in Fig. 1 generate three log templates. Common parameter values include a port number, IP address, file path, number, etc. The format is not the same as those used by different systems and different devices to express the same parameter. So an accurate template cannot be obtained only through regular expression matching. The method LogOHC in this paper is to use the idea of online hierarchical clustering for log parsing and get the log template in real-time accurately.

2.2 Related work

At present, log template extraction has two main methods: clustering-based log template extraction method [14–18] and heuristic-based log template extraction method [12, 13].

The clustering-based log template extraction method firstly calculates the similarity among logs, uses the corresponding clustering method to classify the logs into different categories according to the similarity, and finally extracts the log templates. SLCT proposed by Vaarandi [16] was the earliest method of log template extraction. This method classified logs with common frequent words into one category; however, this method did not consider the position information of words. Tang and Li proposed LogTree [15] to convert the raw log messages into tree-type semi-structured information. It classified the words in the log messages into five categories including word, number, IP address, etc., and then established similarity function based on whether the words in the same position in the two



```
1 - 1117987430 2005.06.05 R35-M0-N3-C:J07-U11 2005-06-05-09.03.50.751145 R35-M0-N3-C:J07-U11 RAS
KERNEL INFO generating core.3450
2 - 1117987430 2005.06.05 R35-M0-N3-C:J15-U01 2005-06-05-09.03.50.775159 R35-M0-N3-C:J15-U01 RAS
KERNEL INFO generating core.3440
3 - 1119103692 2005.06.18 R16-M1-N2-C:J17-U01 2005-06-18-07.08.12.026766 R16-M1-N2-C:J17-U01 RAS
KERNEL INFO CE sym 2, at 0x0b85f680, mask 0x0e
4 - 1120591331 2005.07.05 R32-M0-N4-I:J18-U01 2005-07-05-12.22.11.843446 R32-M0-N4-I:J18-U01 RAS
KERNEL INFO ciod: Missing or invalid fields on line 1 of node map file /home/auselton/bgl/
64mps.sequential.mapfile
```

```
1 generating core.*
2 CE sym *, at *, mask *
3 ciod: Missing or invalid fields on line * of node map file *
```

Fig. 1 Description of the log templates extraction problem

log messages were of the same type, clustered and extracted the log templates. Tang et al. proposed LogSig [18] to generate system events from textual log messages. By searching the most representative message signatures, LogSig categorizes log messages into a set of event types. It was able to incorporate human's domain knowledge to achieve high performance. It used the domain knowledge to establish regular expressions, converted each log message into sets of word pairs, and clustered based on word pairs, then extracted log templates for each category. This method was not limited by the log format. Ning et al. proposed HLAer [17], a heterogeneous log analysis system, which adopted a hierarchical clustering method. He et al. proposed Drain [14]. It did not require source code or any information other than raw log messages. It converted the log messages into a fixed-length analysis tree according to certain rules, first grouped the log messages by the length of the log messages, then grouped the log messages by tokens, and finally compared whether the words in the same position were the same word to establish a similarity function, clustered, and extracted the log templates.

The heuristic-based log template extraction method extracts the templates according to the format information of the raw log messages or the word information in the raw log messages. Makanju et al. [12] proposed IPLoM to add location information on the basis of SLCT. It made full use of the format characteristics of the raw log messages, carried out three-step hierarchical partition on the raw log messages, classified the log messages into different categories, and finally generated a log template for each category. However, this method was only applicable to the logs with an extremely strict format. Du et al. proposed Spell [13] to extract the log templates based on the idea of the longest common subsequence matching. With streaming, real-time message template and parameter extraction produced by Spell not only provides a concise, intuitive summary for end users, but the logs are also represented by clean structured data to be processed and analyzed further using advanced data analytics methods by down-stream analysts. It solved the online extraction problem of the log templates for the first time. The experimental results showed that this method had significantly improved the F1-score and efficiency compared with the previous offline algorithm.

There are other studies on the log template extraction. For example, Beschastnikh et al. [19] used pre-defined rules in the specialized domain for expression. Fu et al. [20] extracted log templates using log print statements in the source code. The method using the rule for representation requires domain knowledge. The method of using the source code for log parsing, although highly accurate, is generally not available, because ordinary engineers cannot get the source code generating logs. In the field which researches of log analysis, a great deal of researchers did a lot of related works about detecting anomaly [21–31].

The most important feature of the abovementioned clustering-based log template extraction method is that when the similarity function is established. Different from the general text data, it fully explores the structural information of the log data and establishes a similarity function suitable for the log data characteristics. However, different types of log data have different characteristics. It is difficult to find a general similarity function that can be applied to all types of log data. Additionally, most of these algorithms are offline algorithms, which occupy high system memory and cannot meet the online requirement of log analysis. Although some heuristic-based log template extraction methods achieve online extraction of log templates, they do not analyze their applicability to multi-source log data.

3 LogOHC method

We now present LogOHC, an online log template extraction method for log messages. In this section, the distributed word representation (word2vec) method is used to online vectorize the log messages, and the online hierarchical clustering algorithm is used to cluster the log messages, and finally log templates are generated.

The basic steps of the LogOHC method are shown in Fig. 2.

3.1 Log preprocessing and online vectorization

The raw log messages cannot be directly used as an input to the clustering algorithm, and a mathematical method is needed to convert the text into digital information in vector form. If the raw log messages are directly vectorized, there will be problems such as excessive log vocabulary, long training time, and excessive interference words affecting the training effect. For example, each raw log message will contain an accurate log generation time. In a data set containing millions or even hundreds of millions of raw log messages, only the log message generating time will make the vocabulary very large, which greatly affects the training speed. The role of the timestamp is negligible for us to study log template extraction. Therefore, we need to perform data cleaning for the raw log message data set first and use regular expression matching to remove parameter values including the time and IP address. Then, we need to segment the words and remove the stop words for the log messages. The experiment has proved that the 2000 raw log messages randomly selected in the BLG data set were preprocessed, and the number of words in the vocabulary generated after preprocessing was reduced from more than 10,000 to more than 2000 compared with the number of words in the vocabulary directly generated without preprocessing, which greatly improved operational efficiency.

The preprocessed log messages are represented using the word2vec. The basic idea is to map each word into a K-dimensional real number vector by training. This paper uses “CBOW + negative sampling” training model and optimization method, as shown in Fig. 3. A piece of text in the log data set $w_{-c}, \cdots w_{-2}, w_{-1}, w, w_1, w_2, \cdots w_c$ is marked as a sample $(\text{context}(w), w)$. CBOW used a three-layer neural network: the input layer, the projection layer, and the output layer. It uses the context of the word to predict the current word. The input layer is $2c$ context word vectors $V(\text{context}_i(w))$ of w, the projection layer X_w is the sum of all $V(\text{context}_i(w))$, and a negative sampling optimization algorithm is used for the output layer to find the word embedding that makes $g(w)$ the largest. It is considered that w is a positive sample, and different negative sampling algorithms can be used to select the negative sample. After a non-empty negative sample set $\text{NEG}(w)$ about $\text{context}(w)$ is determined, the corresponding words in the log vocabulary are determined to have a positive sample label of 1 and a negative sample label of 0. In the objective function of the output layer,

$$p(u|\text{context}(w)) = \begin{cases} \sigma\left(X_w^{T}\theta^u\right), & u = w \\ 1-\sigma\left(X_w^{T}\theta^u\right), & u \neq w \end{cases}, \sigma \text{ is sigmod function} \tag{1}$$

What obtained from word2vec is a vector representation of one word, and the vector representation of the entire log message cannot be directly obtained. This paper averages the word embedding of each word in the log message, as the vector representation of the entire log message $V(l_j)$.

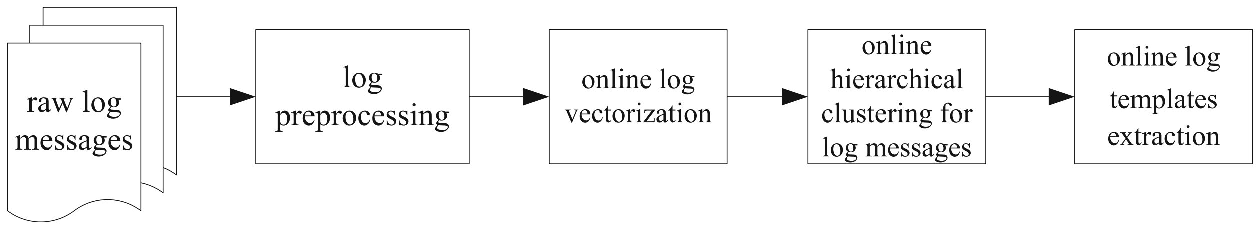

raw log messages
log preprocessing
online log vectorization
online hierarchical clustering for log messages
online log templates extraction

Fig. 2 Basic steps of the LogOHC method

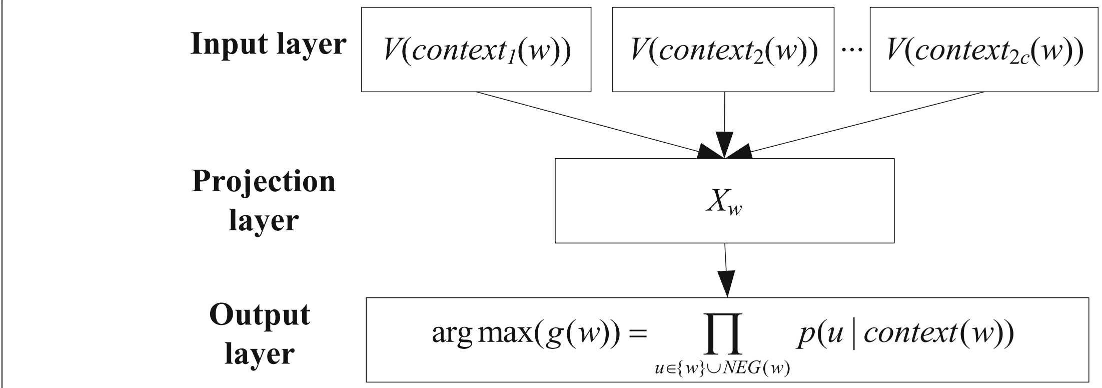


Fig. 3 CBOW + negative sampling training and optimization process

$$V(l_j) = \frac{1}{|l_j|}\sum_{i=1}^{|l_j|} V(w_i) \tag{2}$$

$V(w_i)$ represents the word embedding of w_i, l_j represents jth log message, and $|l_j|$ represents the length of the jth log message (the number of words).

The word2vec model can easily learn the vector representation of new words from the new log message. The specific process is to judge whether the words in the newly added log message are in the log vocabulary; if not, add new words to the log vocabulary and load the trained model to incrementally train new words. That is, we can load new words into the existing model and online get the word embedding without re-learning all.

3.2 Online hierarchical clustering for log messages

Hierarchical clustering is to create a hierarchical nested clustering tree by calculating the similarity between different types of data points. In the clustering tree, the original data points of different categories are the lowest layer of the tree. The top layer of the tree is the root node of a cluster, and the root node cluster covers all the data points. LogOHC applies an online hierarchical clustering algorithm [32] to aggregate and group log messages with similar structures. Each leaf node corresponds to a log message, and any internal node corresponds to a class cluster. The elements in the class cluster are all leaf nodes using this internal node as the ancestor node. The online hierarchical clustering algorithm for log messages includes the following three basic steps:

Step 1: A new log message is inserted, as shown in Fig. 4, traverse the current clustering tree, compares the distance between the newly inserted log message node l' and all leaf nodes (all log messages that have been inserted), and the nearest log message l is found. There are two cases: for the first case in Fig. 4, l' is inserted directly, so that l and l' have a common parent node; for the second case in Fig. 4, the structure of the tree is adjusted so that l and l' have a common parent node. This is done by cutting the connection between l and its parent node and inserting the new internal node p as the parent node of l and l'. However, the disadvantage of this is that the algorithm complexity is too high, and each time a new node is inserted. It has to traverse the entire clustering tree. So the idea of a bounding box is

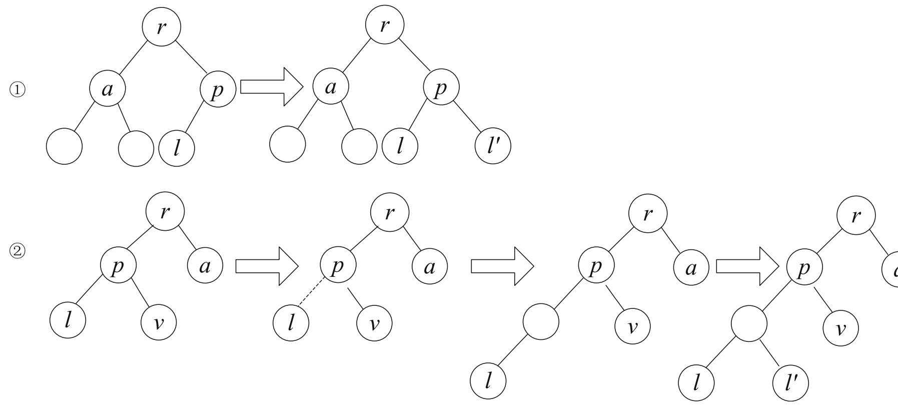


Fig. 4 Steps to insert a new node

introduced; here, each internal node maintains a bounding box that contains all of its leaves (log messages). When a new log message $l^{'}$ is inserted, it is not necessary to compare the distances with each of the inserted log vectors, only to compare the boundary distance with the internal nodes, as so to logarithmically reduce the algorithm complexity. $d_{\min}(l^{'},v)^2$ and $d_{\max}(l^{'},v)^2$ are respectively used to represent the squares of the minimum and maximum distances between the newly inserted log message $l^{'}$ and the internal node v. j in Formulas (3) and (4) represents the jth dimension. v_{j-} and v_{j+} represent the minimum and maximum coordinates of the internal node v in the jth dimension coordinate. $V_j(l^{'})$ represents the value of the newly inserted log message $l^{'}$ in the jth dimension coordinate. Taking 2D as an example, as shown in Fig. 5, due to $d_{\max}(l^{'},l) < d_{\min}(l^{'},v)$, l and $l^{'}$ belong to the same cluster.

$$d_{\min}(l',v)^2 = \sum_{j=1}^{d} \begin{cases} \left(V_j(l')-v_{j-}\right)^2 & \text{if } V_j(l') \le v_{j-} \\ \left(V_j(l')-v_{j+}\right)^2 & \text{if } V_j(l') \ge v_{j+} \\ 0 & \text{others} \end{cases} \tag{3}$$

$$d_{\max}(l',v)^2 = \sum_{j=1}^{d} \max\left\{ \left(V_j(l')-v_{j-}\right)^2, \left(V_j(l')-v_{j+}\right)^2 \right\} \tag{4}$$

Step 2: Determining if any nodes are masked. The basis is Formula (5):

$$\max_{y \in lvs(v^{'})} \|x-y\| > \min_{z \in lvs(a)} \|x-z\| \tag{5}$$

where $v^{'}$ is the sibling node of v, $lvs(v^{'})$ represents the set of all log messages contained in the internal node $v^{'}$, and a is the sibling node of the $v^{'}s$ parent, $x \in lvs(v^{'})$.

As in the first case of Fig. 4, if $\|l^{'}-l\| > \|a-l\|$, the node l is masked. The tree structure is adjusted by the following steps to insert a new log message, as shown in the blue part of Fig. 6.

Step 3: Balanced rotations of the clustering tree.

Both of the adjustments of the tree after the node being masked (Fig. 6) and the adjustment of the tree by directly inserting the nodes (the case 2 in Fig. 4) cause the structure of the tree to change. The depth of the adjusted tree is too deep, and the difference in depth between the left subtree and the right subtree exceeds 1, resulting in a tree unbalance. It is required to further rotate the unbalanced tree based on the balanced binary tree.

The pseudo codes of the inserted new log online are as follows:

```
The pseudo codes of the insert a new log online are as follows:
Algorithm 1 online hierarchical clustering
Function Insert (T, $l'$)
 t=NearNeighbor($l'$)
 if t has no sibling:
   T.addPnode($l'$)
 else
   x=Split(t)
   for a in Ancestors(x) do
     a.addPnode($l'$)
   end for
 T=T.Rotate(x.Sibling)(),CheckMasked)
 T=T.Rotate(x.Sibling)(),CheckBalanced)
 return T
```

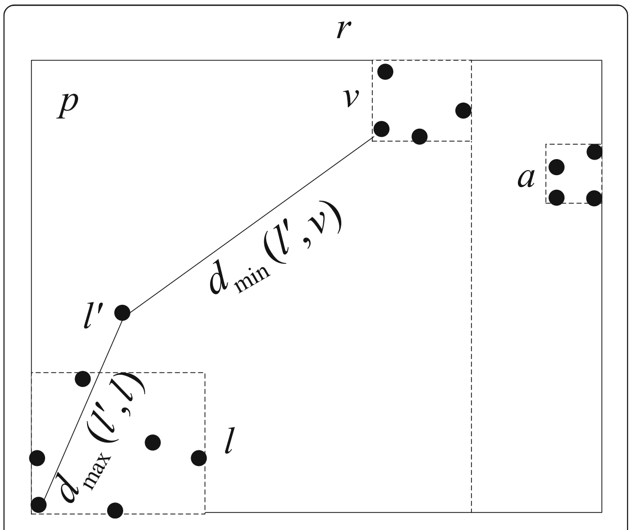


Fig. 5 Optimized method of inserting new nodes

3.3 Online log template extraction

Found from actual experiments, log messages with the same template have the same length. Do the following for the new log added to the cluster:

Step 1. Determine whether the new log message added to the current cluster is the same as the length of the existing log template in the current cluster. If yes, go to step 2. If no, go to step 3.

Step 2. Compare the new log message added to the cluster and the template of the same length in the current cluster word by word to determine whether they are the same word. If yes, the word in this position in the template keeps the word unchanged; if not, the word in the position in the template is replaced by *. Go to step 1.

Step 3. Use the newly added log message as a new template in the current cluster. Go to step 1.

The pseudo codes of the online log template extraction algorithm are as follows:

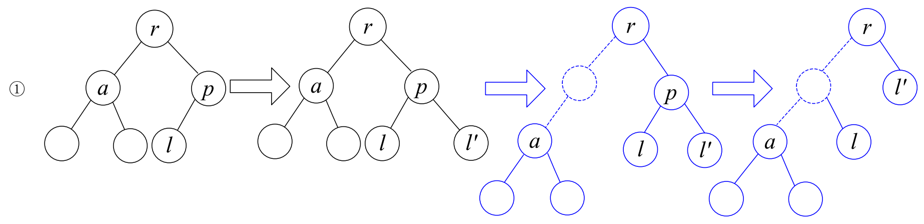


Fig. 6 Processing method for node being masked

The pseudo codes of the online log templates extraction algorithm are as follows:

```
for currentcluster do
  if len(newlog)==len(cluster)
    for columnIdx in range(len(event)):
      if newlog[columnIdx]==event[i][columnIdx]
        continue
      else
        event[i][columnIdx] = '*'
    endfor
  else
    event[i+1]= event.append(newlog)
endfor
return event
```

3.4 Algorithm complexity analysis

In the LogOHC algorithm, the main computation overhead is log online vectorization and online hierarchical clustering. In the log vectorization phase, the time complexity is $O(\log V)$ (V is the size of the vocabulary). During the online hierarchical clustering process, the process of adding a new log message uses A^* search. This paper introduces the idea of the bounding box to optimize its search strategy. The optimized time complexity is $O(\log N)$. During the online log template extraction phase, the complexity is $O(E^*L)$ (E is the count of templates, L is the largest length of the template). So the total time complexity is $O(\log V+\log N+ E^*L)$.

4 Experimental evaluation

4.1 Experimental environment and data sets

The experimental environment uses Windows 7 system, 32-GB dual-channel DDR4 memory, Intel(R) Core(TM) i7-6800 K CPU @ 3.40 GHz processor. The log data set BlueGene/L (BGL) is a public, partially tagged data set from IBM's renowned high-performance computing lab (Lawrence Livermore National Labs, LLNL). The BGL data set [33, 34] contains 4,747,963 raw log messages of 215 days, with a size of 708 M. HPC data set [33, 34] is a high-performance cluster log collected by Los Alamos National Labs, containing 433,490 raw log messages. HDFS data set [33, 34] is a log data set collected from the 203 node cluster of the Amazon EC2 platform. He et al. [35] randomly sampled 2000 raw log messages from BGL, HDFS, and HPC called 2kBGL, 2kHDFS, and 2kHPC, and performed manual classification and template extraction for 2kBGL, 2kHDFS, and 2kHPC, and obtained 112, 14, and 44 log templates respectively. To verify that the LogOHC method is applicable to multi-source log data, 1500, 3000, and 4500 raw log messages were randomly sampled from the combined data set from 2kBGL, 2kHDFS, and 2kHPC (Totaldata, 6000 log messages in total) to obtain Sampledata1, Sampledata2, and Sampledata3. The data sets used in this experiment are shown in Table 1.

4.2 Assessment methods

In this paper, the experiment measures the effectiveness of online clustering based on purity. Defining the purity of the cluster i as:

$$p_i = \max\left(p_{ij}\right) \tag{6}$$

where $p_{ij} = \frac{m_{ij}}{m_i}$ refers to the probability that the members in the clustering i belonging to the true class j, m_{ij} refers to the number of members in the clustering i belonging to the real class j, and m_i is the number of all members in the clustering i. The purity of the entire clustering is:

$$\text{purity} = \sum_{i=1}^{K} \frac{m_i}{m} p_i \tag{7}$$

For the measurement of the clustering effect of the clustering tree, this experiment was based on the normally defined purity, combined with the structural characteristics of the clustering tree, and referred to the evaluation criteria of the clustering tree called *dendrogram purity* in the literature [36] (the dendrogram purity of clustering tree T is recorded as DP(T)). The known tagged data has a

Table 1 Data set information

Data set	Number of raw log messages	Number of log templates
2kBGL	2000	112
2kHDFS	2000	14
2kHPC	2000	44
Totaldata	6000	170
Sampledata1	1500	112
Sampledata2	3000	138
Sampledata3	4500	155

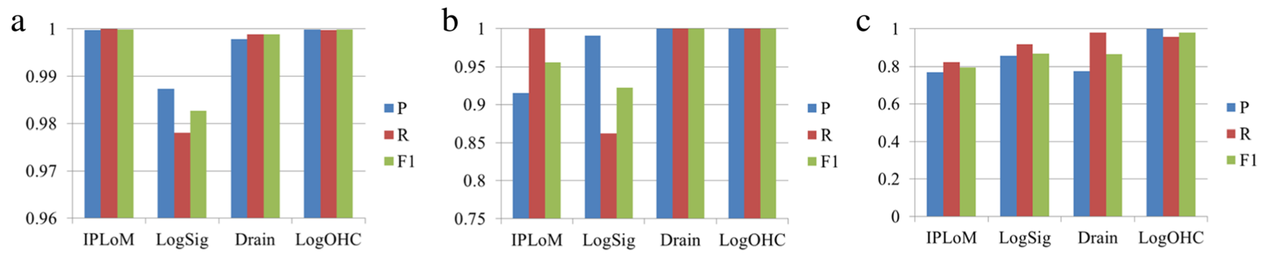


Fig. 7 Comparison of the LogOHC and the other methods on 2kBGL (**a**), 2kHDFS (**b**), and 2kHPC (**c**)

total of K clusters, which are recorded as $C^* = \{C_k^*\}_{i=1}^K$ and

$$\mathrm{DP}(T) = \frac{1}{|P^*|}\sum_{k=1}^{K}\sum_{x_i,x_j\in C_k^*} pur\left(lvs\left(LCA\left(x_i,x_j\right)\right), C_k^*\right) \tag{8}$$

where $P^* = \{(x_i, x_j) | C^*(x_i) = C^*(x_j)\}$ refers to all the data pairs in the tagged data belonging to the same clustering, $LCA(x_i, x_j)$ refers to the least common ancestor of x_i and x_j, and $lvs(z)$ refers to all the leaves belonging to the internal node z in $T{\cdot}pur(S_1, S_2) = |S_1 \cap S_2|/S_1$.

Meanwhile, we adopt three indicators of Precision, Recall, and F1-score to measure the LogOHC method proposed in this paper. As the Precision and Recall of the classification process are a pair of contradictory indicator, the use of F1-score can effectively balance the Precision and Recall, and the closer to 1 of F1-score numerical value means better classifier's performance. Where TP, FP, TN, and FN are used to respectively represent true positive cases, false positive cases, true negative cases, and false negative cases, the calculation of precision as well as recall rate along with F1-score formula is as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{9}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{10}$$

$$\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{11}$$

4.3 Experimental analysis

The LogOHC is used to perform an experiment on data sets described in Table 1, and compared with the classic offline algorithms IPLoM [12], LogSig [18], and the online algorithm Drain [14]. For the convenience of comparing Precision, Recall, and F1-score with other three algorithms, the clustering number of LogOHC is set as the number of manually tagged categories in the experiment. The comparison results are shown in Fig. 7.

Results show that the LogOHC exceeds the best results of the classic offline algorithm IPLoM, LogSig, and the online algorithm Drain on the three data sets in Precision, Recall, and F1-score.

To further measure the effect of LogOHC on multi-source log data sets, experiments are performed to

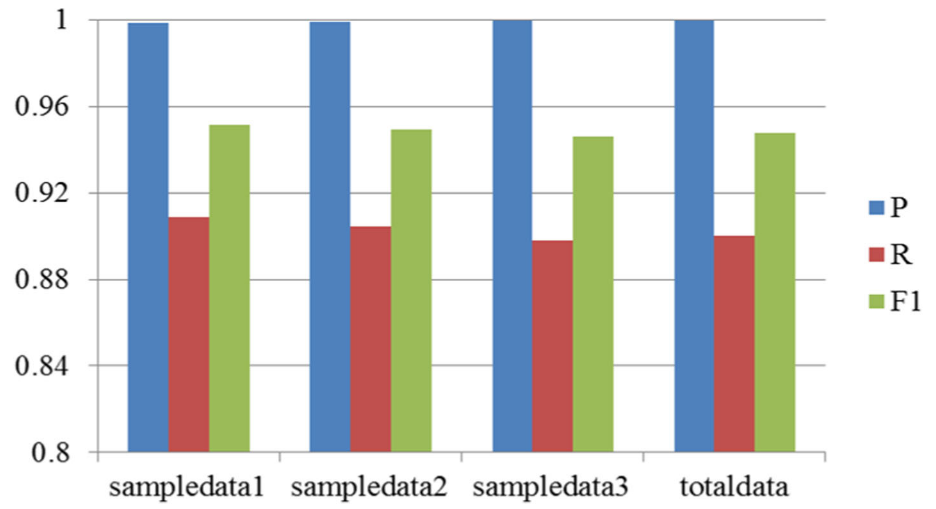


Fig. 8 Comparison of P, R, F1-score of LogOHC on multi-source log data sets

Table 2 F1-score on multi-source log data sets with four methods

Data set	Methods			
	IPLoM	LogSig	Drain	LogOHC
Totaldata	0.9334	0.6727	0.9446	0.9474
Sampledata1	0.5358	0.4074	0.9312	0.9514
Sampledata2	0.2995	0.2475	0.9368	0.9495
Sampledata3	0.1976	0.165	0.942	0.946

compare F1-score on the multi-source log data sets Totaldata, Sampledata1, Sampledata2, and Sampledata3. The results of LogOHC performed on the above four data sets are shown in Fig. 8. And the results of comparison with the other three methods are shown in Table 2.

As can be seen from Fig. 8, the LogOHC has a lower average F1-score on multi-source log data set than that on single-log data set. However, it is not the larger the multi-source log data set, the worse the algorithm performance, for example, the experimental results with the algorithm on Totaldata are better than those on Sampledata3. As can be seen from Table 2, the LogOHC has a higher F1-score than other three methods on all multi-source log data sets. Both results indicate that the LogOHC has good effect on multi-source log data set. This is very significant in practice, because the log data is always multi-source, and the algorithm can be used for clustering analysis of multi-source log data effectively as a general algorithm, whose results provide a good research data source and research method for subsequent researches on anomaly detection.

To further measure the clustering effect of LogOHC on multi-source data sets, this experiment calculates and compares the dendrogram purity values on four multi-source data sets and three single-log data sets described as Table 1. As shown in Fig. 9, it can be seen that the dendrogram purity value for multi-source log data sets is lower than the maximum value of dendrogram purity for all single-log data sets, but is equivalent to the average value of dendrogram purity for single-log data sets, which is consistent with expectations. Even so, the minimum value of dendrogram purity clustered for the four multi-source log data sets is still more than 98.5%, indicating good clustering results.

4.4 Analysis on algorithm execution time

The experimental environment is as described in Section 4.1, and LogOHC is applied to the data sets in Table 1. The single-step execution time is shown in Fig. 10. It can be seen that the single-step execution time of LogOHC method does not exceed 0.025 s, which is able to well meet the requirements of real-time log analysis.

4.5 Parameter selection

During the online clustering process, the similarity measure between nodes and class clusters is an important parameter of the algorithm, and its value has an important influence on the effect of the algorithm. Since the online clustering algorithm does not select the similarity measure method with best results after fully acquiring all the data and takes it as an appropriate parameter, the log data is acquired in an incremental mode and clustered online in accordance with the parameters set in advance. Therefore, it is

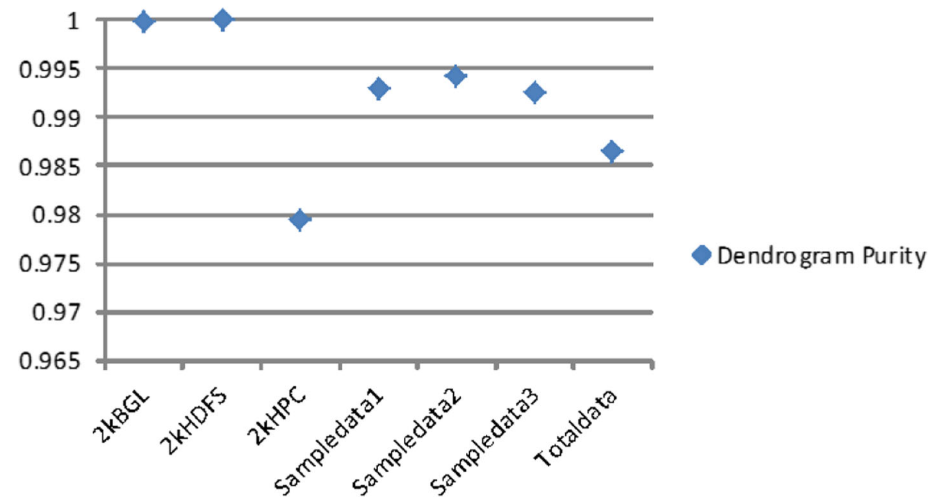


Fig. 9 Comparison of the dendrogram purity of the LogOHC on different data sets

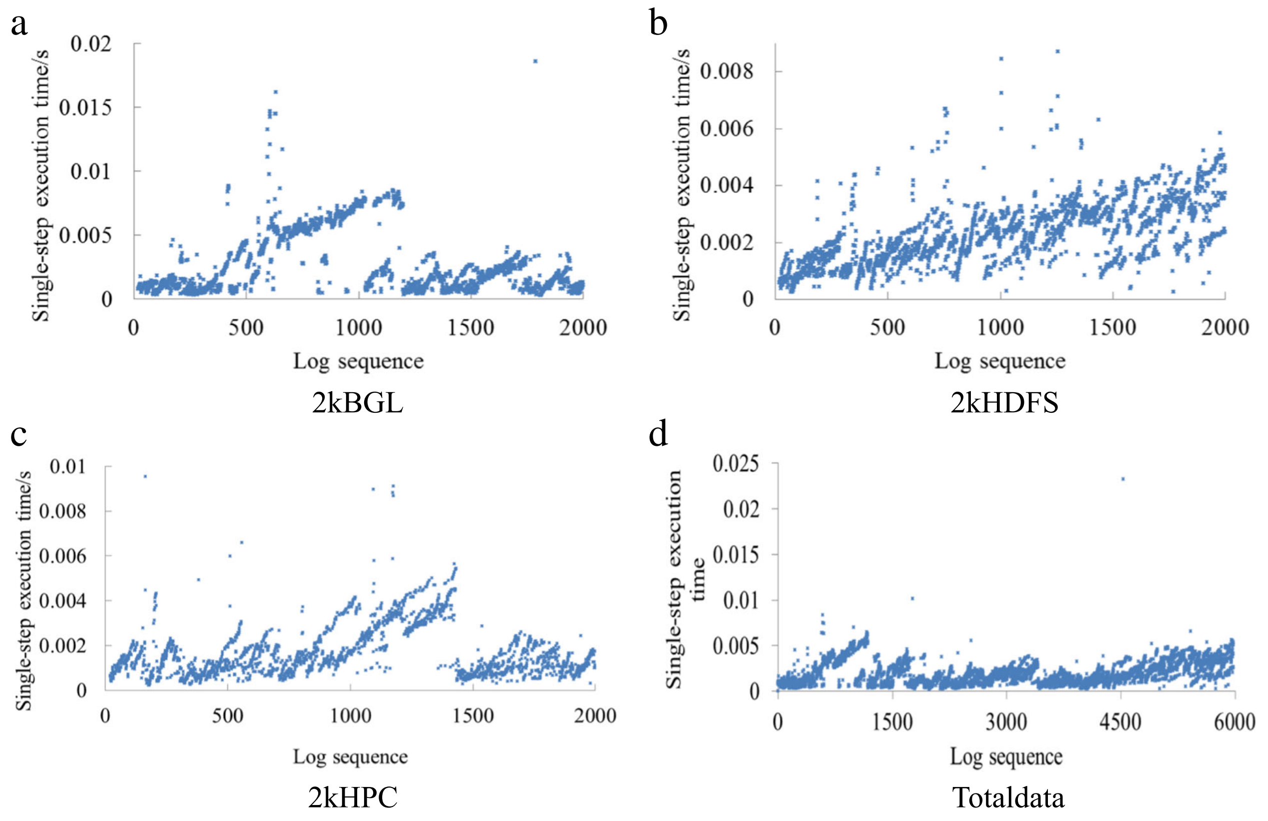


Fig. 10 Single-step execution time of LogOHC on data sets. **a** 2kBGL. **c** 2kHPC. **b** 2kHDFS. **d** Totaldata

necessary to select similarity measure parameters that can be stable on different data sets and can acquire relatively good results. In order to select the parameter, six methods for measuring similarity are selected on four data sets in this paper: 2kBGL, 2kHDFS, 2kHPC, and Totaldata. The dendrogram purity of the corresponding clustering tree and the F1-score of the clustering result are calculated and compared. The results are shown in Figs. 11 and 12. The same parameters perform differently on different data sets, for example, on the 2kBGL data set and the 2kHDFS data set, both maxD and minD parameters can be selected to acquire good clustering result, whereas the clustering result of the same parameter on 2kHPC data set, compared to other parameters, reduces the dendrogram purity by nearly 20% and reduces the F1-score by 27%. As can be seen from Figs. 11 and 12,

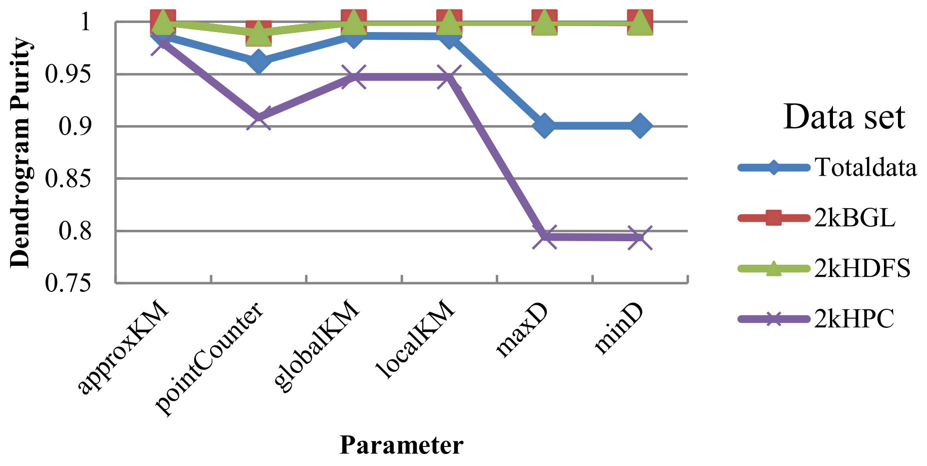


Fig. 11 Dendrogram purity of clustering tree corresponding to different parameters clustering on different data sets

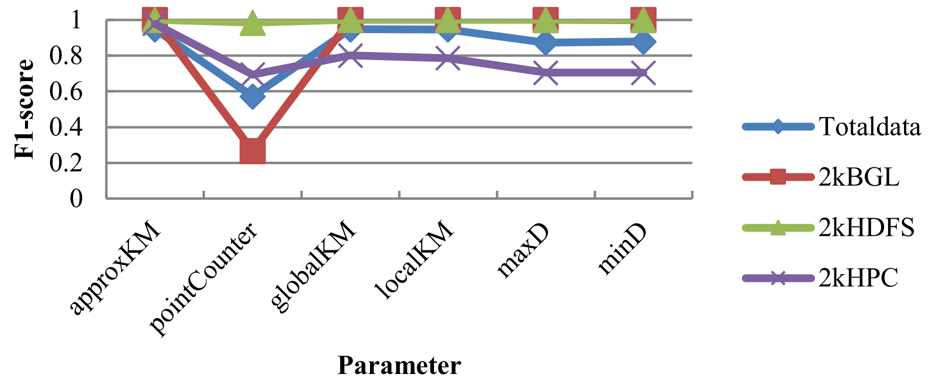


Fig. 12 F1-score corresponding to different parameters clustering on different data sets

the parameters approxKM, gloabalKM, and localKM, compared to other parameters, make the algorithm have a better clustering result; however, the results of gloabalKM and localKM on 2kHPC data set reduce the dendrogram purity by 3% and reduce the F1-score by 17% and 19% respectively than those of approxKM. Through the above experimental analysis, the approxKM parameter is selected by the LogOHC method to measure simiarity. The meanings of the six parameters are respectively

pointCounter = the number of nodes in class cluster
maxD = the maximum distance between nodes in class cluster
minD = the minimum distance between nodes in class cluster

$$\text{approxKM} = 0.5^{*}\text{pointCounter}^{*}\text{maxD}$$

$$\text{localKM(node)} = \sqrt{(x_1 - \overline{x})^2 + (x_2 - \overline{x})^2 + \cdots + (x_n - \overline{x})^2},$$

where $\overline{x}$ is the clustering center

globalKM(node) = localKM(node) – (localKM(child1) + localKM(child2)), where child1 and child2 are the two children nodes

5 Conclusion

In view of the shortcomings of the existing log template extraction methods, this paper proposes an online hierarchical clustering for log template extraction: LogOHC. With the ideas of text vectorization and online hierarchical clustering, firstly, the raw log messages are preprocessed, and the word distributed representation (word2vec) is used to vectorize the log messages online. Then, the online hierarchical clustering algorithm is applied, and finally, log templates are generated. The experimental analysis shows that, compared to the existing classic offline methods and the newly proposed online method, the LogOHC has higher F1-score and is suitable for multi-source log data. The template extraction method is not limited by the log format, and the single-step execution time is shorter, which is able to meet the requirements of real-time online processing. Next, we will continue to carry out research in two aspects: one is to explore the heuristic method for determining the cluster number in log online hierarchical clustering; the other is to further research the online log anomaly detection method based on the online log template extraction.

Abbreviations
CBOW: Continuous Bag-of-Words; DP: Dendrogram purity; FN: False negative cases; FP: False positive cases; LLNL: Lawrence Livermore National Labs; TN: True negative cases; TP: True positive cases

Acknowledgements
The research presented in this paper was supported by WS-DREAMwsdream which towards open data sets and source code for web service recommendation powered by The Chinese University of Hong Kong, Hong Kong.

Funding
The authors acknowledge the Science and Technology on Information Transmission and Dissemination in Communication Networks Laboratory Research fund.

Availability of data and materials
The data set can be downloaded at https://github.com/logpai/

Authors' contributions
RY is the main writer of this paper. She proposed the main idea, completed the evaluation, and analyzed the result. DQ gave good suggestions on the integrity and innovation of the paper. YQ , YD and SZ put forward some suggestions for the design of the experimental part. All authors read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Publisher's Note

Author details
[1]PLA Strategic Support Force Information Engineering University, Zhengzhou, China. [2]No. 24, East Jianshe Road, Zhengzhou, China. [3]Northwestern Polytechnical University, Xi'an, China.

Received: 19 February 2019 Accepted: 10 April 2019
Published online: 28 May 2019

References

1. Y. Dai, H. Li, Y. Qian, X. Lu, A malware classification method based on memory dump grayscale image. Digit. Investig. **27**, 30–37 (2018)
2. L. Yuchong, L. Xingguo, Q. Yekui, Z. Xin, Network-wide traffic anomaly detection and localization based on robust multivariate probabilistic calibration model. Math. Probl. Eng. **2015**, 1–26 (2015)
3. C. Wang, Q. Xu, X. Lin, S. Liu, Research on data mining of permissions mode for Android malware detection. Clust. Comput. **2018**, 1–14 (2018)
4. A. Nandi, A. Mandal, S. Atreja, G.B. Dasgupta, S. Bhattacharya, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Anomaly detection using program control flow graph mining from execution logs (ACM, 2016), pp. 215–224
5. M. Du, F. Li, G. Zheng, V. Srikumar, in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. DeepLog: anomaly detection and diagnosis from system logs through deep learning, vol 2017 (ACM, 2017), pp. 1285–1298
6. A. Rahman, Y. Xu, K. Radke, E. Foo, in *International Conference on Network and System Security*. Finding anomalies in SCADA logs using rare sequential pattern mining (Springer, Cham, 2016), pp. 499–506
7. A. Oprea, Z. Li, T.F. Yen, S.H. Chin, S. Alrwais, *Detection of early-stage enterprise infection by mining large-scale log data, Dependable Systems and Networks (DSN), 2015 45th Annual IEEE/IFIP International Conference on* (IEEE, 2015), pp. 45–56
8. N. Nagashree, R. Tejasvi, K.C. Swathi, An early risk detection and management system for the cloud with log parser. Comput. Ind. **97**, 24–33 (2018)
9. D. Kim, D. Shin, D. Shin, Y.H. Kim, Attack detection application with attack tree for mobile system using log analysis. Mob Netw App. **2018**, 1–9 (2018)
10. K. Nagaraj, C. Killian, J. Neville, in *Proc. USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. Structured comparative analysis of systems logs to diagnose performance problems (2012)
11. W. Xu, L. Huang, A. Fox, D. Patterson, M.I. Jordan, in *Proc. ACM Symposium on Operating Systems Principles (SOSP)*. Detecting large-scale system problems by mining console logs **2009**, 117–132 (2009)
12. A. Makanju, A.N. Zincir-Heywood, E.E. Milios, A lightweight algorithm for message type extraction in system application logs. IEEE Trans. Knowl. Data Eng. **24**(11), 1921–1936 (2012)
13. M. Du, F. Li, Spell: streaming parsing of system event logs. Data Mining (ICDM), 2016 IEEE 16th International Conference on IEEE, 859–864 (2016)
14. P. He, J. Zhu, Z. Zheng, M.R. Lyu, Drain: an online log parsing approach with fixed depth tree. Web Services (ICWS), 2017 IEEE International Conference on IEEE, 33–40 (2017)
15. L. Tang, T. Li, in *2010 IEEE International Conference on Data Mining*. LogTree: a framework for generating system events from raw textual logs (IEEE, 2010), pp. 491–500
16. R. Vaarandi, A Data Clustering Algorithm for Mining Patterns From Event Logs, IP Operations & Management, 2003.(IPOM 2003). 3rd IEEE Workshop on. IEEE, 2003: 119–126
17. X. Ning, G. Jiang, H. Chen, K. Yoshihira, *HLAer: a System for Heterogeneous Log Analysis* (Sdm, 2014), pp. 1–22
18. L. Tang, T. Li, C.-S. Perng, LogSig: generating system events from raw textual logs. Proceedings of the 20th ACM International Conference on Information and Knowledge Management ACM, 785–794 (2011)
19. I. Beschastnikh, Y. Brun, M.D. Ernst, A. Krishnamurthy, *Inferring models of concurrent systems from logs of their behavior with CSight*, Proceedings of the 36th International Conference on Software Engineering (ACM, 2014), pp. 468–479
20. Q. Fu, J.-G. Lou, Y. Wang, J. Li, *Execution Anomaly Detection in Distributed Systems through Unstructured Log Analysis, Data Mining, 2009. ICDM'09*, Ninth IEEE International Conference on (IEEE, 2009), pp. 149–158
21. C. Wickramage, C. Fidge, T. Sahama, A. Daly, R. Wong, *Preserving Privacy through Log Analysis in Health Information Systems* (2017)
22. Q. Xu, M. Li, M. Yu, Learning to rank with relational graph and pointwise constraint for cross-modal retrieval. Soft. Comput., 1–15 (2018)
23. S. Liu, M. Li, M. Li, Q. Xu, in *Concurrency and Computation: Practice and Experience*. Research of animals image semantic segmentation based on deep learning (2015), p. e4892
24. Moh, M., Pininti, S., Doddapaneni, S., & Moh, T. S. Detecting web attacks using multi-stage log analysis[C]//Advanced Computing (IACC), 2016 IEEE 6th International Conference on. IEEE, 2016: 733–738
25. Q. Xu, M. Li, A new cluster computing technique for social media data analysis. Clust. Comput., 1–8 (2017)
26. W. Peihe, Z. Dekai, Convexity of level sets of minimal graph on space form with nonnegative curvature. J Diff Equ. **262**, 5534–5564 (2017)
27. D. Sun, M. Fu, L. Zhu, G. Li, Q. Lu, Non-intrusive anomaly detection with streaming performance metrics and logs for DevOps in public clouds: a case study in AWS. IEEE Trans. Emerg. Top. Comput. **4**(2), 278–289 (2016)
28. Q. Xu, Z. Wang, F. Wang, J. Li, Thermal comfort research on human CT data modeling. Multimed. Tools Appl. **77**(5), 6311–6326 (2018)
29. J. Breier, J. Branišová, in *Information Science and Applications*. Anomaly detection from log files using data mining techniques (Springer, Berlin, Heidelberg, 2015), pp. 449–457
30. Tuor, A., Kaplan, S., Hutchinson, B., Nichols, N., & Robinson, S. Deep learning for unsupervised insider threat detection in structured cybersecurity data streams. arXiv preprint arXiv:1710.00811, (2017), pp. 224–231
31. Q. Xu, M. Li, M. Li, S. Liu, Energy spectrum CT image detection based dimensionality reduction with phase congruency. J Med Syst. **42**(3), 49 (2018)
32. Kobren, A., Monath, N., Krishnamurthy, A., & McCallum, A. (2017). An Online Hierarchical Algorithm for Extreme Clustering. arXiv preprint arXiv:1704.01858
33. Zhu, J., He, S., Liu, J., He, P., Xie, Q., Zheng, Z., & Lyu, M. R. (2018). Tools and Benchmarks for Automated Log Parsing. arXiv preprint arXiv:1811.03509
34. P. He, J. Zhu, S. He, J. Li, M.R. Lyu, Towards automated log parsing for large-scale log data analysis. IEEE Trans Dependable Secure Comput. **15**(6), 931–944 (2018)
35. P. He, J. Zhu, S. He, J. Li, M.R. Lyu, *An Evaluation Study on Log Parsing and Its Use in Log Mining*, 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) (IEEE, 2016), pp. 654–661
36. K.a. Heller, Z. Ghahramani, in *Proceedings of the 22nd International Conference on Machine Learning*. Bayesian hierarchical clustering (ACM, 2005), pp. 297–304