*Fuzzy Systems and Data Mining VIII*
*A.J. Tallón-Ballesteros (Ed.)*

*doi:10.3233/FAIA220378*

# Automated Log Analysis and Anomaly Detection Using Machine Learning

Ali Hussain SHAH[a], Daem PASHA [a], Esmaeil Habib ZADEH [a] and Savas KONUR [a, 1]
[a] *Department of Computer Science, University of Bradford, Bradford, BD7 1DP, UK*

**Abstract.** Reducing the number of alerts and anomalies has been the focus of several studies, but an automated anomaly detection using log files is still an ongoing challenge. One of the pertinent challenges in the detection of anomalies using log files is dealing with 'unlabelled' data. In the existing approaches, there is a lack of anomalous examples and that log anomalies can have many different patterns. One solution is to label the data manually, but this can be a tedious task as the data size could be very large and log files are not easily understandable. In this paper, we have presented an automated anomaly detection model that combines supervised and unsupervised machine learning with domain knowledge. Our method reduces the number of alerts by accurately predicting anomalous log events based on domain expertise, which is used to create automated rules that allow generating a labelled dataset from unlabelled log records, which are unstructured and present in many different formats. This labelled dataset is then used to train a classification model that will help predict anomalous log events. Our results show that we can accurately predict anomalous and non-anomalous events with an average accuracy of 98%. Our approach offers a practical solution for systems where logs are collected without any labelling, making it difficult to create an accurate model to identify anomalous log records. The methodology presented is very fast and efficient, which can provide real-time anomaly detection for time critical environments.

**Keywords.** Anomaly detection, logs, machine learning, clustering, labelling

## 1. Introduction

Software systems store important events taking place in a system in the form of log files, which can be used for troubleshooting purposes via the use of log messages. Log events can also be used for anomaly detection. For example, timestamps or severity of the log can be used for the identification of anomalies, but these provide limited information, hindering the recognition of all anomalous log events. Effective approaches therefore require specific domain knowledge to detect abnormalities in the log data.

Although anomaly detection can be done by checking the log files manually, it is not feasible because of the complexity and the sheer size of data available. Therefore, an automated process is required to analyse logs and identify anomalies. Automated anomaly detection can allow trouble-shooters to spend less time in finding the issues, hence more time can be spent to fix the problem.

There have been several studies that focus on reducing the number of alerts and anomalies [1-10], but an automated anomaly detection using log data is still an ongoing

[1] Corresponding Author: Savas KONUR, Department of Computer Science, University of Bradford, Bradford, BD7 1DP, UK; Email: s.konur@bradford.ac.uk

challenge. Anomaly detection is, in fact, a binary classification, deciding between normal and anomalous classes; however, there are a number of challenges, which can be summarised as follows:

- the sheer rates at which log events are generated,
- the lack of a unified structure, as log data usually comes in a large variety with many different patterns,
- the absence of labelled records (a.k.a. anomalous examples), which are crucial for any classification problems.

While one solution could be manually labelling and analysing log events, this has proven least cost-efficient, requiring a lot of engineers' time and effort as well as computational resources.

We offer a practical solution where logs are collected with no labels. Our approach in this paper combines unsupervised and supervised machine learning techniques with insights from domain knowledge to provide more effective and better performing anomaly detection models. More specifically, our contributions can be summarised as follows:

- incorporating domain knowledge to create automated rules and generate labelled dataset from unlabelled log records,
- significant reduction in the number of alerts by eliminating and suppressing duplicate records,
- very fast and efficient algorithms, allowing for real-time anomaly detection in time critical environments,
- combining domain expertise with ML algorithms, providing more accurate models,
- not relying on the assumption that "only rare log events are anomalous" or "change in the log patterns indicates abnormal behaviour".

The rest of the paper is organised as follows: Section 2 briefly overviews the existing related work; Section 3 describes our methodology; Section 4 presents experimental results and summarises the findings; Section 5 discusses some relevant issues; and Section 6 concludes with future work.

## 2. Related work

Several studies have been carried out to reduce number of alerts and anomalies. In [1], SOM + K-means algorithms are used to aggregate alerts and Naïve Bayes is used to remove false alerts, thus only keeping true alerts, resulting in a 94% reduction in number of alerts. In [2], K-means clustering algorithm is used to create an alert clustering model for a Network-based Intrusion Detection Systems (NIDS) with an accuracy of 99.67%. [3] proposes a two-stage classification system using a SOM neural network and K-means algorithm for an Intrusion detection system (IDS) to correlate the relevant alerts and to further categorise the alerts into classes of true and false alarms, which led to a reduction of all excessive and noisy alerts, which often contribute to more than 50% of false alarms generated by a common IDS. [4] uses event correlation analysis to group correlated events on log data based on the degree of similarity to reduce alarms compared to analysing individual alarms. [5] applies clustering technique and supporting evidence to significantly reduce alerts. [6] uses semantic relationships between logs to find those that are anomalous. [7] proposes a deep neural network model utilising Long Short-Term

Memory (LSTM) to learn log patterns different from the normal to allow detection of anomalies. [8] uses semantic relationships between logs to generate word vectors and weighted log sequence feature vectors with Term Frequency-Inverse Document Frequency (TF-IDF), which is then used with K-Means clustering to detect anomalous logs. [9] uses isolation forest and deep autoencoder neural networks to detect anomalous logs. Instead of finding anomalous logs, this approach predicts if logs are positive or negative, which reduces false alerts. [10] proposes a two-part deep autoencoder model that uses LSTM units which do not require hand-crafted features; this is useful in identifying rare/unseen log events by assigning each log event an anomaly score. [11] provides an automated anomaly detection solution by using a time series clustering approach, where K-means clustering is used to find time spans where the system in question does not behave as expected. However, this method offers limited capabilities as in this approach small clusters are used to identify anomalous log events with the help of domain knowledge. [12] proposes an approach where logs are sampled at regular intervals to compute scores which are used to train a semi-supervised deep autoencoder. In this approach, normal log events are used to train the model and to find anomalies using the trained model. [13] proposes an anomaly detection approach that uses thresholds to distinguish between normal and abnormal log events. In this study, Robust Random Cut Forest (RRCF) is used to find the outliers and threshold setting is determined by various threshold values such as maximum deviation, three-sigma, and median absolute deviation to determine the best threshold to identify anomalies.

In literature, there are many different machine learning techniques that can be used to find anomalies in the system. Those that are based on supervised learning have higher accuracy rates; however, more often than not, the labelled datasets required by these approaches do not exist and preparing such datasets is not a straightforward task either in real world scenarios. As for unsupervised learning, log events that deviate from the norm are flagged to provide early detection to prevent down-time and reduce Mean Time To Repair (MTTR). However, these approaches do not provide any context to help trouble-shooters understand what the problem is, leaving them to manually inspect log files to find the root causes.

Our approach is different from the studies reported above because we use a combination of unsupervised, supervised machine learning techniques with domain knowledge. The aim of unsupervised K-means clustering in our approach is to allow to understand data better and to provide a meaningful representation that would allow automation rules to be defined using domain expertise. This would then allow creation of a labelled dataset to train the classification model for automated anomaly detection. Unlike other studies, with this approach we do not rely on the assumption that outliers are anomalies. Moreover, because we use domain knowledge to create automated rules to label logs, we get a more accurate anomaly detection model as we are not using thresholds or semantic meanings of the logs, or rare/unseen logs to label log records as anomalies. In addition, our approach does not require the need to create log parsers, as all information is kept such that it can be used to identify anomalous log records which removes log parser related errors. Furthermore, this approach can be applied to any system as it removes the need to have a labelled dataset for anomaly detection.

## 3. Methodology

In this section, we present our automated anomaly detection method that combines supervised and unsupervised machine learning and domain knowledge to reduce the number of alerts by accurately predicting anomalous log events.

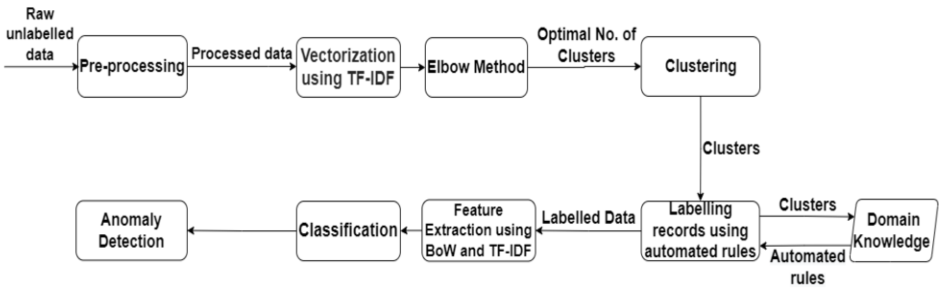

**Figure 1.** Methodology.

Our methodology is presented in Figure 1. The Pre-processing step involves the removal of digits and punctuation, keeping only textual data. The Term Frequency Inverse Document Frequency (TF-IDF) is then used to convert textual data into meaningful numerical representation of text data. Both processes are important for clustering purposes. In our model, a combination of pre-processing steps, the TF-IDF technique, Elbow method and K-means clustering are used to create meaningful clusters. Subsequently, these clusters are analysed to create rules for automation that would allow the creation of a labelled dataset. The labelled dataset is used to extract features using Bag-of-Words (BoW) and TF-IDF models. Naïve Bayes classification model is then used to predict log events as anomaly or normal. This is critical for the detection of log events which can be addressed promptly to reduce major performance issues.

### *3.1. Logs cleaning and unification*

The log events contain different types of information such as timestamp, severity, message, source of the message, dynamic changes of software and hardware, etc. Data loggers do not necessarily follow a standard, unified structural format, i.e., different devices log records in different formats. In order to use log records from a dataset to cluster them using K-means clustering algorithm and to train the Naïve Bayes model, we transform logs from multiple sources into a unified form.

To pre-process the log data, we only keep the message part of each log record, and remove all numbers, special characters, and punctuations. This process is necessary because log parsers are not efficient and can cause semantic misunderstanding, which lead to inaccurate results; so, it is more effective to use raw logs directly to create meaningful representation of textual data. The cleaning of logs involves 4 regex steps: *i.* removing tabs and line breaks; *ii.* removing punctuation and digits; *iii.* removing words with less than three characters; *iv.* transforming logs into lowercase.

## *3.2. Conversion of log events into numerical representation*

### *3.2.1. Natural language processing*

Models in machine learning only understand numerical values, therefore, textual data must be converted into numerical data. The process of converting text data to numerical vectors is called vectorisation, which can be used to build various machine learning models. The words which have similar meanings will be assigned to a vector closer to each other to reduce the dimensionality of the vector space. This is very important because meaningful representation of log data will allow us to provide meaningful clusters.

### *3.2.2. Term Frequency Inverse Document Frequency (TF-IDF) vectoriser*

TF-IDF is used to convert text into meaningful numerical representation by converting text to feature vectors, which are used as input to the K-means clustering. For each log record a vector is created, where each value in the vector corresponds to each word in the log record. TF-IDF algorithm allows us to represent each log record in the dataset as a vector. In this way, relevant log records will have similar vectors, therefore they can be clustered into similar type of groups.

## *3.3. Meaningful clusters and labelling of records using automated rules*

After pre-processing and vectorisation, TF-IDF vectors are used with the K-means algorithm to create clusters, where the Elbow method is used to find the optimal number of clusters. Given the insights from the clustering, we use domain knowledge to help write automation rules; so, each rule from the list of rules will specify and label each single individual log record to create a labelled dataset.

## *3.4. Feature extraction using bag of words and TF-IDF*

The aim of this step is to transform data into feature vectors, which are used as an input to the classification model. Feature extraction is an important step because the quality of the features will help improve the model performance. Bag of Words and TF-IDF models are used for feature extraction as they are useful techniques to extract features from the data. Bag of Words model is a representation of text that counts unique occurrences of words in a dataset by creating a vocabulary of words; whereas TF-IDF denotes how important a word is in a record in the dataset. A higher TF-IDF score means that a word is more important, whereas a low score indicates it is less importance.

Term frequency (TF) is a measure of how frequently a word appears in the log event. To calculate this, the vocabulary created using Bag of Words model is used. Term frequency is calculated using the following formula:

*TF(w,d) = (number of times word 'w' appears in the log event) / (total number of words in the log event 'd')*

Inverse Document Frequency (IDF) is a measure to find how important a word is in the dataset, which is calculated as follows:

*IDF(w) = log (number of log records / number of log records with word 'w')*

TF-IDF score for each word is calculated as:

*TF-IDF(w, d) = TF(w, d) x IDF(w)*

For example, if we consider a log record containing 100 words wherein the word ‘fatal’ appears 5 times, the term frequency (TF) for the word ‘fatal’ is (5 / 100) = 0.05. If we assume that there are one million log records and the word ‘fatal’ appears in 1,000 log records, then the inverse document frequency (IDF) is calculated as log (1,000,000 / 1,000) = 3. Since the TF-IDF score is the product of these two statistics, we get 0.05 x 3 = 0.15.

Using Bag of Words and TF-IDF models, we convert feature vectors into vectors of TF-IDF scores with most important contextual words having higher values.

### *3.5. Classification*

After the pre-processing step, log events are converted into vectors, features are generated, which are then passed to a Naive Bayes classifier to train a classification model. Naive Bayes classifier is a supervised machine learning algorithm based on ‘Bayes’ Theorem which defines the probability of occurrence of an event related to any condition.

Using this classification model, pre-processed log records will be modelled as an unordered collection of words from one of two probability distributions: one representing anomalous log events and the other one representing normal log events. For example, there will be two bags of words, one is filled with words found in anomalous log records and the other one is filled with words from normal log events. This means that anomalous bag of words will contain anomalous-related words while normal bag of words will contain more words that are related to normal log events. To classify log events, the Bayesian filter assumes that the log record is from one of the bags and uses Bayesian probability to determine which bag the log record is in. After the model is trained it can calculate the probability to classify logs as anomalous or normal.

The probability of log record being anomalous is calculated using the formula:

$$P(A|W) = \frac{P(W|A) * P(A)}{P(W|A) * P(A) + P(W|N) * P(N)}$$

where:
P(A|W) is the probability that a log record is anomalous ‘A’ given the word ‘W’ is in it.
P(A) is the overall probability that any given message is anomalous ‘A’.
P(W|A) is the probability that the word ‘W’ appears in a log record given that the log record is anomalous ‘A’.
P(N) is the overall probability that any given log record is normal ‘N’.
P(W|N) is the probability that the word ‘W’ appears in a log record given that the log record is normal ‘N’.

## 4. Experimental results

### *4.1. Dataset*

To demonstrate the effectiveness of our methodology, we have used a public log dataset, BGL(BlueGene/L) [14], collected from a BlueGene/L supercomputer system with

4,747,963 log events, manually labelled as normal or anomalous by Lawrence Livermore National Labs (LLNL). We have split the dataset into train (80%) and test (20%) datasets. The train dataset contains 3,798,370 log events, of which 278,586 are anomalies. The test dataset contains 949,593 log events, of which 69,874 are anomalies. The K-means clustering algorithm is applied to the train dataset in order to find a meaningful set of clusters. Domain knowledge together with the insights from the clustering are then used to create automation rules that help generate a labelled dataset. We use this labelled dataset to construct a classification model.

## 4.2. Pre-processing

In this step, data is processed to only keep textual data. Figure 2 shows an example of an unprocessed log record and Figure 3 shows an example of a processed log record, where only the textual data is kept. This step ensures that we keep all the information that can be used to identify anomalous or normal log events. At the same time, it reduces the size of logs, which allows the model construction to be done more efficiently.

```
KERNSOCK 1136390405 2006.01.04 R00-M0-NC-I:J18-U11 2006-01-04-08.00.05.167045
R00-M0-NC-I:J18-U11 RAS KERNEL FATAL idoproxy communication failure: socket closed\n
```

**Figure 2.** Example of unprocessed log record.

```
kernsock ras kernel fatal idoproxy communication failure socket closed
```

**Figure 3.** Example of pre-processed log record.

## 4.3. Clustering

After the pre-processing step, TF-IDF vectorisation has been used to convert the log events into numerical vectors. The numerical vectors have then been used with the Elbow method to find the optimal number of clusters.

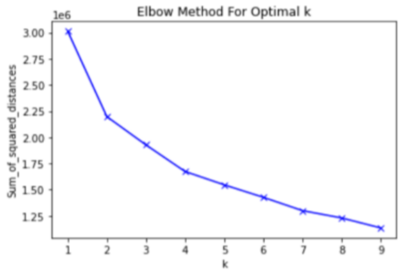

**Figure 4.** Elbow method for k=10.

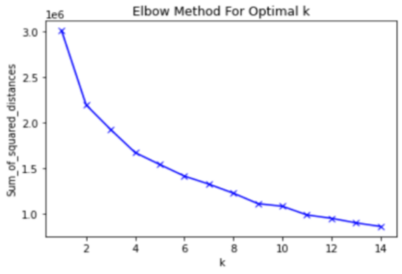

**Figure 5.** Elbow method for k=15.

Figures 4 and 5 represent two Elbow method results. Figure 4 shows the result of running the K-means clustering for a range of 1 to 10 clusters and Figure 5 shows the result of running the K-means clustering for a range of 1 to 15 clusters. Based on these figures, we can say there are two obvious choices, 2 or 4, for optimal number of clusters. We have decided to choose 4 as it provides more granularity than 2. The other clusters are not good candidates. For example, based on Figure 4 we can say that 7 can be a weak candidate; however, it disappears in Figure 5.

The K-means clustering results based on *k*=4 are presented in Table 1. We can see that all anomalous samples are part of the same cluster (Cluster 3). However, this cluster contains a mixture of normal and anomalous events.

**Table 1.** Clustering results.

| Cluster | Total log events in cluster | Anomalous log events in cluster |
|---|---|---|
| 1 | 1,365,476 | 0 |
| 2 | 350,687 | 0 |
| 3 | 1,631,279 | 278,586 |
| 4 | 450,928 | 0 |
| Total No. of Records | 3,798,370 | 278,586 |

## 4.4. Automated rules

Since we are using a public dataset, we do not have any domain knowledge about how the data is labelled, but based on the log data analysis, we have been able to write some rules to label records as anomolous or normal. However, to create more sophisticated or complicated rules we need more in-depth understanding/investigation of these anomalous data to ensure that the dataset is accuretly labelled. This automation process has been created using the rules:

```
begin
clusters = kmeansPrediction
rulesForAnomalousLogs = ['socket link has been severed',
'data tlb error interrupt', 'data storage interrupt']
for logRecord in clusters
      logRecordStatus = 'normal'
      for text in rulesForAnomalousLogs
            if text in logRecord
                  logRecordStatus = 'anomalous'
end
```

Using this algorithm, we can incorporte domain knowledge to automatically label datasets; hence we are able to check log events against these rules to label records as anomalous or normal. After applying these rules, we have created a labelled dataset containing 3,798,370 records in total, of which 212,458 are anomalous.

## 4.5. Feature extraction

To extract features from this data, we have first used the Bag of Words model, followed by the TF-IDF algorithm to convert words into numerical vectors. Using these models, 2,873 words have been extracted as features from a total of 38,705,984 words.

The feature words extracted signify the important words which describe the dataset. These words provide the contextual information and are best used to characterise all log records in the dataset.

## 4.6. Anomaly detection

To create a classification model, we have used Multinomial Naive Bayes. After the model was trained, 20% data from the original log dataset has been used for validation.

**Table 2.** Classification results.

| | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| **0** | 1.00 | 0.76 | 0.87 | 69874 |
| **1** | 0.98 | 1.00 | 0.99 | 879719 |
| **Accuracy** | | | 0.98 | 949593 |
| **Macro Avg** | 0.99 | 0.88 | 0.93 | 949593 |
| **Weighted Avg** | 0.98 | 0.98 | 0.98 | 949593 |

In Table 2, we present our results. Here, 'support' denotes the number of occurrences of the given group in the dataset. As can be seen in the table, 69,874 log events are anomalous and 879,719 are non-anomalous. The classification model can accurately predict 87% of anomalous log events as anomalies and 99% of non-anomalous log events as non-anomalies with a total accuracy of 98%.

Since we create a labelled dataset using some rules generated based on domain knowledge (required to understand the underlying structure of the logs), we only need to label records that are anomalous, which reduces the number of false alerts, as shown in the Table 2.

Furthermore, our approach does not require log parsers because we use raw logs and only keep textual information. Hence, using this approach we can avoid log parser related errors, and can achieve a better performance. Moreover, this allows using different types of logs from different sources without making any changes in our approach as we do not need to create any templates for different types of logs.

The high accuracy rate achieved shows that we have created an effective automated anomaly detection model. Our approach offers a practical solution to predict anomaly detection using unlabelled log data without any manual labelling, which is a very tedious and challenging task. Also, since our machine learning models work very fast, this approach can effectively be implemented in time critical environments.

However, our approach has some limitation. Since we keep all textual information that can be used to label log records as normal or anomaly, we do not have any data-related parameters, which can lead to infeasible solutions. On the other hand, the clustering results are very important because automation-based rules will be defined using the insights from the clustering results, therefore, it is important to accurately pre-process data using steps provided in Section 3.1.

## 5. Discussion

Today monitoring and logging operational status of IT systems is a ubiquitous task, leading to continuous generation and stream of events. These log events are invaluable and the ability to effectively maintain the performance of an IT system heavily relies on careful analysis of these logs. However, as mentioned above, the absence of labelled records, sheer rate of log events and lack of a unified structure altogether make it almost impossible for even enterprise companies to manually analyse and act on log data. Therefore, utilising ML algorithms, our approach here provides a practical solution to automatically deal with log data and help identify anomalous events with high accuracy, enabling much smaller teams to consume and manage logs in a timely fashion.

On the other hand, there is no one-size-fits-all solution and when it comes to performance data, logs have little, if any, to offer to help. While static threshold alerting and logging might be in place, e.g., based on a percentage of interface bandwidth, it would be ill-suited for detection of seasonal anomalies. The common practice, therefore, involves using historical time series data to learn seasonal patterns and construct forecast models; stream of observations is then compared against these baseline forecasts to detect anomalies [15]. Time series, however, have their challenges, one of which being the fact that they usually require a lot of low granular past data, making them computationally expensive. Therefore, depending on the use cases, either or a combination of these complementary solutions can be considered to achieve the best performing and cost-effective solution for detecting and proactively acting on anomalous events.

## 6. Conclusion

In this paper, we have presented an automated anomaly detection model combining unsupervised and supervised machine learning with domain knowledge. The method is composed of the following stages: parsing and analysing logs to create meaningful clusters, creating a labelled dataset using automation rules based on the domain knowledge, and creating an effective classification model to detect anomalous log events.

Our framework, after generating a labelled dataset, uses supervised machine learning models to extract useful features using Bag of Words and TF-IDF algorithms to create an effective anomaly detection model for log events. Although, this approach could require different set of rules for different systems, as different type of domain expertise is needed to create contextual automation rules, it offers an effective anomaly detection model that can be used to significantly reduce the number of alerts.

Automated detection of anomalous log events still faces three main challenges. First, there is no concrete way of distinguishing anomalous log events. Secondly, every software-production environment is different, therefore, identification of anomalous log events requires a different type of domain expertise. Thirdly, the automated analysis process should be speedy enough to allow early detection of anomalies and reduce downtime and improve business operational function.

Our future work will focus on addressing these challenges. We will also consider integrating anomaly detection methods based on time series data in conjunction with log files to improve the efficiency of anomaly detection. Finally, we will apply our approach to use cases requiring high volume of real-time data traffic such as autonomous vehicles [16], smart manufacturing [17], and ubiquitous systems [18,19] and as well as scientific software systems [20, 21].

## Acknowledgements

The authors acknowledge the Innovate UK support [grant number KTP012139].

## References

[1] Afolabi-B OOK, Siraj M. Intrusion alert reduction based on unsupervised and supervised learning algorithms. Int. J. Innov. Comput. 2021; 11(2): 25-34.
[2] Hua HHW, Siraj M, Din MM. Integration of PSO and K-Means clustering algorithm for structural-based alert correlation model. Int. J. Innov. Comput. 2017, 7(2).

[3] Tjhai GC, Furnell SM, Papadaki, M, Clarke NL. A preliminary two-stage alarm correlation and filtering system using SOM neural network and K-means algorithm. Comput. Secur 2010; 29(6):712-723.
[4] Noda M, Higuchi F, Takai T, Nishitani H. Event correlation analysis for alarm system rationalisation. Special Issue: Process Systems Engineering (PSE) Asia 2010; 6(3):497-502.
[5] Njogu HW, Jiawei L. Using alert cluster to reduce IDS alerts. International Conference on Computer Science and Information Technology, 2010; p. 467-471.
[6] Le VH, Zhang H. Log-based anomaly detection without log parsing. 36th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2021; p. 492-504.
[7] Du M, Li F, Zheng G, Srikumar V. DeepLog: Anomaly detection and diagnosis from system logs through deep learning. Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, p. 1285–1298.
[8] Wang J, Zhao C, He S, Gu Y, Alfarraj O. LogUAD: log unsupervised anomaly detection based on word2vec. Comput. Syst. Sci. Eng. 2022; 41(3): 1207-1222.
[9] Farzad A, Gulliver TA. Unsupervised log message anomaly detection. ICT Express 2020; 6(3): 229-237.
[10] Bursic S, Vittorio C, D'Amelio A. Anomaly detection from log files using unsupervised deep learning. FM 2019 International Workshops: Porto, Portugal, October 7–11, 2019; p. 200–207.
[11] Schmidt T, Hauer F, Pretschner A. Automated anomaly detection in CPS log files. computer safety, reliability, and security. SAFECOMP 2020. Lecture Notes in Computer Sci. 12234. p. 179–194.
[12] Marta C, Antonio P, Umberto V. AutoLog: Anomaly detection by deep autoencoding of system logs. Expert Syst Appl. 2022; 3: 116-263.
[13] Toluwalope David A, Barjinder K, Sajjad D, Ali A. Ghorbani. Threshold based technique to detect anomalies using log files. In 7th International Conference on Machine Learning Technologies, 2022; p. 191–198.
[14] GitHub. A large collection of system log datasets for AI-powered log analytics. https://github.com/logpai/loghub Accessed June 14, 2022.
[15] Zadeh E, Amstutz S, Collins J, Ingham C, Gheorghe M, Konur S. Automated contextual anomaly detection for network interface bandwidth utilisation: a case study in network capacity management, Proceedings of 11th Int. Conf. CECNet, Front. Artif. Intell. Appl. 2022; 345: 659-666.
[16] Badue et. al. Self-driving cars: A survey. Expert Sys. Appl. 2021; 165: 113816.
[17] Konur S et. al. Towards design and implementation of Industry 4.0 for food manufacturing. Neural Comput. Appl. 2021, p. 1-13.
[18] Arapinis M et. al. Towards the verification of pervasive systems. Electronic Communications of the EASST, 2010, 22.
[19] Konur S, Fisher M, Dobson S, Knox S. Formal verification of a pervasive messaging system. Form. Asp. Comput. 2014; 26(4): 677-694.
[20] Blakes J et. al. Infobiotics workbench: A P systems-based tool for systems and synthetic biology. Applications of Membrane Computing in Systems and Synthetic Biology, Emergence, Complexity and Computation, 2014; 7: 1-41.
[21] Bakir ME et. al. Extended simulation and verification platform for kernel P systems. International Conference on Membrane Computing, LNCS, 2014; 8961: 158-178.