Identifying Anomaly Detection Patterns from Log Files: A Dynamic Approach

Claudia Cavallaro and Elisabetta Ronchieri(✉)

INFN-CNAF, Bologna, Italy
{claudia.cavallaro,elisabetta.ronchieri}@cnaf.infn.it

Abstract. Context: Services that run in a data center can be configured to store information about their behaviour in specific logs. They contain huge amount of data that makes difficult to manually understand run-time properties of services. Therefore, to facilitate their analysis it is important to use dynamic solutions to quickly parse log files, detect anomaly and diagnose problems in order to react promptly.

Objectives: We want to build a model for determining anomaly detection in a certain period of time. Furthermore, we wish to identify machine learning techniques that support us determining problems patterns according to the messages available in logs.

Method: We have selected machine learning techniques, such as Invariant Mining model, natural language process and autoencoder, able to work with messages and identify anomaly patterns. According to the data available we have decided to study monthly data and detect samples with a higher frequency of problems. We have scanned the various logs to search the services with a wrong behaviour in the same period of time to recognize past anomalies in the data center and code these behaviours.

Results: The results are promising. We have obtained an average of F-measure metric over 86%.

Conclusion: Our model aims at quickly recognizing problems and solving them. It helps site administrators to better understand the run services, code anomalies and crosscheck different messages in the same time slot.

Keywords: Log analysis · Log mining · Log parsing · Anomaly detection · Machine learning

1 Introduction

A data center has to gather information about computer systems states. It is a common practice to have programs logging events that provide insights on the service activity and report their internal state, allowing to detect anomalies. Logs are semi-structured texts, usually appended to a file, with the '.log' extension, that grows in size and becomes very large, therefore the fact that system administrators analyze systems' health according to log files does not scale. This leads to develop solutions that automate the processing of logs, reducing human intervention. Software components and applications produce heterogeneous log files. There are services that use extremely flexible logging methods in their syntax, like syslog [22]. Log files usually do not contain the same type of information: ‘syslog’ event logs a system activity, while ‘crond’ event logs the CRON entries that show up in ‘syslog’. Each file tends to describe a partial view of the whole data center. The stored information can contain the time and date of specific event to log exactly what happened and does use a simple formatting. Furthermore, the services can use different key words to express normal or erroneous behaviour. Logging analysis involves processing large amount of data that are not easy to read and understand manually. This can make difficult to perform log summarizing. Non-automatic processes for trouble shooting [40] are discouraged in large computing system. Several studies have proposed approaches to scale up log analysis [20] , moving the error problems from manual operation to automated operation. However, they still may require log processing and conversion of log files into a format that can be understood by analysis tools. Some methods are used to compact the information into more readable formats, such as ‘.csv’ and ‘.json’, and to also automatically extract any suspicious information, like the cause of an ‘error’ message. Quite often it is not feasible just selecting the file and retyping the file extension as ‘.csv’ or ‘.json’, because the transformed file could be wrongly reformatted. Furthermore, the file usually contains daily service information, so the number of data can be over $60 - 120\ K$ rows, penalizing the usage of some analysis tools, because they will likely become unresponsive when they perform, e.g., selecting and adding new variables in the data sets. Before starting any analysis it is essential to decide which variables have to be included in the resultant data sets. Spreading the data across multiple columns is another aspect to consider to organise your data set into a manageable format. The resultant files can be used to identify trends and unusual activity that is beneficial for both short- and long-term data center management. Machine Learning techniques are a solution to identify anomaly detection patterns automatically, predicting the failure of the machine.

In this paper, we focus on determining problems patterns by combining the results obtained by different machine learning (ML) techniques. Log files contain a considerable amount of texts, therefore we have used modern Natural Language Processing (NLP) methods [1,33] to map the log files’ words to a high dimensional metric space in order to define site administrators’ actions. The aim is to cluster and- if possible- classify the various system events. Furthermore, having to work with unsupervised data, we have used a machine (deep) learning technique called autoencoder [56]: the decision taken by and information available to site experts are not stored in any local framework. We have also used an invariant mining technique [37] that supports, for example, detection of failures and anomalies. During the log analysis, we have adopted the invariant mining model, because it is a general approach that does not rely on the nature of the data and it does not require any meaningful knowledge of the domain or constant rules. Furthermore, it does not require training labels and messages are not grouped with respect to distance only. The proposed approach is repeatable in other contexts and domains. With minimum setup effort and the usage of machine learning tools, it is possible to automatically extract relevant information about system state. Through experiments, we illustrate the potential benefits of our approach by answering our research questions.

The reminder of this paper is as follows. Section 2 introduces the background of log analysis and provides related works. Section 3 introduces the rational of our approach. Section 4 provides some promising experimental results. Section 5 concludes the paper with future work.

2 Log Mining and Related Works

Existing commercial tools, such as Elk [18], Splunk [45], Loggly [36] or Over Ops [41], are mainly used for Big Data visualization, but are not appropriate for anomaly detection. Moreover, utilities such as Salsa [30] or Log Enhancer [55] for diagnosability are not good on large volume data, because they are heavy. Log mining is an approach that uses statistical and ML algorithms in structured event list to extract knowledge from logs, discovering a model inference and detecting outliers in a system.

Log Compression. Preliminary phases to log mining are log compression, which uses, for example, dictionary-based and statistic-based techniques or Jaccard similarity [28], and log parsing, which collects clustering, heuristics, frequent pattern mining and evolutionary algorithms. Frequent pattern mining identifies frequencies, correlations and causalities between sets of items in transactional databases: among the most used algorithms, Apriori [46] and FP-growth [23] can be mentioned. The Apriori algorithm is based on a level structure and it is useful to search for the most frequent schemes [5] from a large dataset, to quickly exclude non-recurring patterns. FP-growth bases its operation on the construction of a tree structure that models the considered dataset. In particular, the frequent item sets are extracted by going through the tree, without therefore having to read the database several times.

Log Parsing. In the preprocessing techniques category, also called log abstraction, there is the log parsing activity used to abstract and simplify the initial data. Among the clustering methods for log parsing, we can mention Simple Logfile Clustering Tool (SLCT) [49], Hierarchical Event Log Organized (HELO) [21] and Iterative Partitioning Log Mining (IPLoM) [38]. In addition, POP [25], a parallel log analyzer, is suitable for large-scale data processing. During the log parsing process, metrics are sometimes used, for example LevenShtein distance [34] or Longest Common Subsequence [48] in Spell [14].

Data Transformation. In the preprocessing phase, there is also data transformation. Through this process the log set is transformed into an appropriate form, such as an histogram matrix of events or a binary transformation, which becomes the input in the next data mining step. El-Masri et al. [17] compare some aspects, such as scalability and efficiency, of the main log abstraction techniques.

Log Analysis. Logs can be analyzed in a static way (off-line method) or dynamically (on-line method), but the latter method is time consuming. The former is less precise but also faster. Hybrid analysis is a mixture of the two. The most popular on-line log parsing approaches are SHISO [39], LenMa (which are based on clustering), Drain [26] (based on heuristics) and Logram [9] (based on frequent pattern mining). A more detailed description of clustering and log parsing methods can be found in the survey [27]. Log analysis techniques can be divided into the following branches: failure detection techniques that include, for instance, decision tree and associative rule learning; anomaly detection techniques that comprises agglomerative or divisive hierarchical clustering, nearest neighbor approach, chi-squared test and Naive Bayes (NB) classifier [13], and failure diagnosis that includes the SherLog [54] tool and Decision Tree (DT) [42] approaches.

Machine Learning Techniques. Some traditional ML algorithms are applied in literature, including Principal Component Analysis (PCA) [53], LogCluster [50], Support Vector Machine (SVM) [51] and mixtures of Hidden Markov models. Some Deep Learning approaches for this scope are DeepLog (LSTM-RNN) [15], LogGAN (CNN) [52], Aarohi (a failure prediction method) [10], AirAlert (a framework based on Bayesian network) [7] and TCFG (Time Weighted Control Flow Graphs) [29]. Machine Learning techniques are divided into classification techniques (such as SVM and Bayesian networks), clustering techniques (including K-means [35], Self-Organizing Feature Map, SOFM [32] and DBSCAN [44]) and statistical techniques (for example PCA). A clustering algorithm divides data into groups, according to the criterion that the elements belonging to the same cluster are very similar. Outliers, which are those that do not belong to any cluster, often contain useful information on the anomalous characteristics of the system as they are generally different from other data. To estimate how much the results of the algorithms implemented in the log mining process correspond to reality, one can proceed in several ways. A first strategy is to measure true and false positives, true and false negatives and plotting the curve receiver operating characteristic (ROC). Other procedures, used to evaluate clustering methods, include internal evaluation (given by the Davies-Bouldin index [11] or Dunn index [16]), and external evaluation (with Rand measure [43], F-measure or Jaccard index).

Related Works. Breier and Branisova [3], employing Apache Hadoop framework, process log files in parallel. Their method, based on the generation on dynamic rules, spots anomalies through MapReduce. The parallel implementation, compared with A-priori and FP-Growth algorithms, has accelerated the process of detecting security breaches in log records.

Borghesi et al. [2], through a deep learning technique, perform anomaly detection in high performance computing systems. The Examon infrastructure (the code is available on Github [19]) dataset is explored through a neural network called autoencoder. It is trained with the Adam optimizer [31], by using an offline approach. Finally, the metric used to evaluate the accuracy of the obtained results is F-measure. Layer et al. [33] use NLP to predict the operator's action for the CMS experiment [8] workflow handling. They employ the information of the computing operators for taking the decision in case of failures, stored in the experiment framework, during the ML training phase. Bertero et al. [1] also use NLP to identity anomaly detection. They emulate different types of errors that refer to CPU consumption, misuse of memory, abnormal number of disk accesses, network packet loss and network latency.

The results presented in previous papers show that the optimal procedure for analyzing logs is the combination of, first, heuristics or filtering and, next, machine learning steps. In this article, we have chosen to use invariant mining and autoencoder techniques, because we are interested in the relationships between different messages and source hostnames. We have also applied NLP to identify anomaly categories that can be used to label the log entries and contribute to defining administrators' operations. Tools, like SLCT, have the disadvantage of being based on searching only for the most frequent types of messages in the log file, neglecting the less frequent ones. For anomaly detection, however, it may be necessary to find rare types of messages, so this option has been omitted. Furthermore, the application of clustering methods in the log core analysis phase would lead to the risk of losing significant results on the anomalies to be considered individually, if they have been aggregated with others.

3 Approach

The approach proposed as the contribution of this paper is presented in Fig. 1. Once data has been collected and the log files characteristics have been identified, we have dedicated to the following phases: data preprocessing and transformation that have allowed us to build a set of datasets to train and build our anomaly detection models.

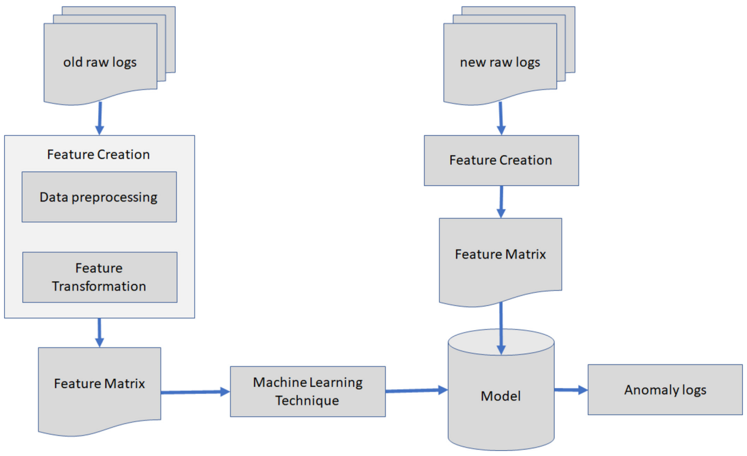


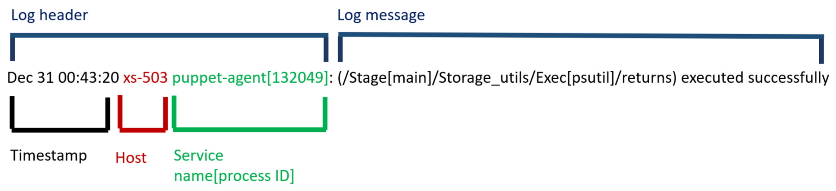
Fig. 1. General approach overview.

Source Data. The log files examined in this study are related to a set of services running at INFN Tier-1 data center [12], used by the large hadron collider experiments [4]. Low level services are shared among the highlighted groups, such as crond and sudo. The log files mainly belong to Linux system services, such as the software utility crond, the free and open-source main transfer agent postfix and the standard for message logging syslog. Table 1 summarizes the first 30 filenames according to their frequency, that is the number of times a value of an unique filename occurs. The suffix-type frequencies of the various available files are summarized as follows: .gz 10869, .log 3562759, .manifest 5, .meta 6, .pending 1, .txt 2.

Table 1. The top 30 log files per frequency.

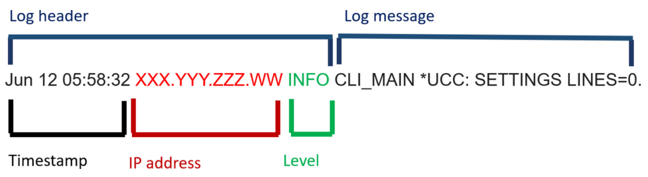
Filename	Frequency	Filename	Frequency	Filename	Frequency
sudo.log	378781	systemd.log	107700	userhelper.log	21380
puppet-agent.log	368530	mmfs.log	72620	nslcd.log	20544
run-parts.log	365734	rsyslogd.log	70210	neutron_linuxbridge.log	8572
crontab.log	348896	kernel.log	65938	runuser.log	6859
crond.log	347708	logrotate.log	62531	cvmfs_x509_validator.log	6031
sshd.log	303919	syslog.log	47330	cvmfs_x509_helper.log	5399
anacron.log	287419	yum.log	43301	srp_daemon.log	4938
postfix.log	175558	fusinv-agent.log	42125	edg-mkgridmap.log	4083
auditd.log	120473	root.log	37345	libvirtd.log	3328
smartd.log	109441	gpfs.log	31000	dbus.log	3301

Log files. Each of these log files contains a different amount of lines. They contain numerical and textual data that describe system states and run time information. Each log entry includes a message that contains a natural-language text (i.e. a list of words) describing some events. Logs are generally generated by logging statements inserted, either by software developers in source code or by system administrators in configuration files, to record particular events and software behaviour. Each log entry in the log file represents a specific event. Figures 2 and 3 show two different log entry samples composed of a log header and a log message: the former is generally composed of a timestamp, a custom-configuration information (such as the hostname in Fig. 2 where the service runs and a log level verbosity in Fig. 3) and the name of the service the message is associated to; the latter is just the message that contains information of the logged event.



```
Dec 31 00:43:20 xs-503 puppet-agent[132049]: (/Stage[main]/Storage_utils/Exec[psutil]/returns) executed successfully
```

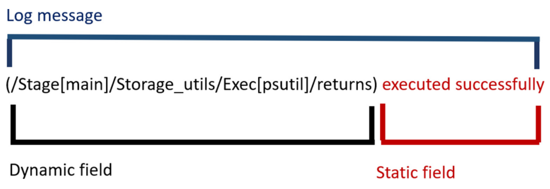
Fig. 2. A log entry sample from the puppet-agent service.



```
Jun 12 05:58:32 XXX.YYY.ZZZ.WW INFO CLI_MAIN *UCC: SETTINGS LINES=0.
```

Fig. 3. A log entry sample from the puppet-agent service.

Figures 4 and 5 show two examples of the log message of a log entry, characterized by a natural-language text which interpretation is difficult because there is not an official standard defining the message format. The text is usually composed of different fields called dynamic and static: a dynamic field (DF) is a string or a set of strings that are assigned at run time; a static field (SF) does not change during events. Such fields can be delimited by different separators, such as a comma, a white space, a parenthesis. Logging practice is scarcely well documented. This activity mainly depends on human expertise [24]. They often have to analyze a large volume of information that may be unrelated to the problematic scenarios and lead to overwhelming messages [6].



```
(/Stage[main]/Storage_utils/Exec[psutil]/returns) executed successfully
```

Fig. 4. A log message fields with just one dynamic field and static field.

Data Preprocessing. During this phase the log files change format and turn into '.csv' files. In addition the following variables are added to each file: date, time, timestamp, hostname, internet protocol (ip) address, service name, process identifier (id), component name, message (msg). The hostname and ip couple are not always both available, especially when the service runs on a virtual machine. Each file is related to a particular service that runs on a well-known machine in the data center. Its location is got by a local database and included into the resulting file. During this phase we have tackled some site administration peculiarities: the same service called in lower or capital letters; the process identifier included in the service logging file; the service name included (or not) in the log message; the process identifier included (or not) in the log message; the logging filename included typo error. Before performing any cleaning operations, we have excluded meaningless services' log files, especially those with a small number of events. In the remaining logs, the following changes have been applied in the message field: the removal of unwanted texts, such as punctuation, non-alphabets,

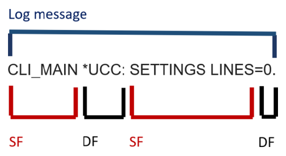


Fig. 5. A log message fields with a sequence of dynamic and static fields.

and any other kind of characters that are not part of the language by involving regular expressions; the exclusion of non-English characters; the cancellation of stopwords, that is frequent general words, like 'of, are, the, it, is', with a low meaning. According to the amount and types of services, in this phase we have started to trace the types of log events, such as assert, fail, error and debug, and to identify anomaly key terms that can be used to classify the reason of the problems in the service. This part of the study is still ongoing requiring experts' check. However, Table 2 summarizes few examples of message lines that describe a wrong service behaviour.

Table 2. Examples of message lines for the crond log file.

log event msg	log event type	anomaly key term
.. reset error counters	error	reset
.. failed create session connection time out	fail	connection

Features Transformation. Data transformation includes the creation of a new dataset that includes a matrix, whose dimension and values change according to the machine learning technique used to build the anomaly detection model. The resulting dataset can contain either binary data or numerical data. In case of binary data there is a numerical value of 1 for features that are present in the log event and a numerical value of 0 otherwise. In case of NLP usage, the tokenization phase (see Fig. 3) determines the element of the matrix: the message string is split up in single words. Once transformed the message events into a matrix of features, in case of problem with large dimension, it is possible to apply rules to reduce data, e.g., according to the uniqueness of the messages. For NLP, low frequency words or words that are not important for the meaning of the anomalies are filtered out. For a matter of time computation, the event count matrix has been created for each month, in which its elements indicate the occurrences of all messages given in input and relating to the hostnames from which they come in that time window.

Table 3. Examples of message strings split up in single words for the crond log file.

log event msg	..	connection	..	error	..	fail	..	time	..
.. reset error counters	..	0	..	1	..	0	..	0	..
.. failed create session connection time out	..	1	..	0	..	1	..	1	..

Machine Learning. Starting from the event count matrix, NLP, autoencoder and invariant mining techniques have been considered. NLP includes the word2vec [47] unsupervised algorithm, a popular embedding approach by Google to process natural language. word2vec takes a text corpus as input and produces a high-dimensional euclidean space with each unique word in the corpus. Each word of an event is mapped to nearby points in the vector space. This technique is able to produce meaningful word embeddings: similar words end up close, words that are not related to each other end up far away in the embedding space. Following the same approach to all log files, it is possible to form clusters of similar anomaly events and use traditional supervised classifiers to e.g. determine anomalous and normal behaviour of the services.

Autoencoder is a type of neural network that can be used to detect anomalies at the service level. With respect to Borghesi et al. [2], in this study we have created a set of separate autoencoder models, one for each service in the system. Each model is trained by using a loss function to ensure that the output is close to the input.

Invariant mining model is based on the fact that a number of initial operations on the files corresponds to the same number of final operations on the same or a similar type, for example the number of jobs arrived with the number of jobs started, or the number of jobs arrived and number of jobs completed. Linear program invariant is then a predicate that always contains the same values in different normal operations, such as opening and closing files. Invariants reflect the underlying correlation between variables and the properties of the execution path. Whenever an invariant is broken during a system operation, the anomaly in the log corresponding to these variables is detected because it is considered a sign of a probable malfunction. The final result is a vector of length equal to the number of unique messages given in input, and composed of values "0" and "1". Each element corresponds to a log message, in which "1" indicates that it is labeled as anomalous, "0" instead as non-anomalous.

Some messages, erroneously labeled as anomaly, could be log patterns that at the beginning of a month correctly close the operations of the end of the previous month, but are not linked to the messages that precede them if we have separated the dataset by month. These boundary cases can be assessed individually, considering a limited time window e.g. 24 h before and after, or taking into account the average closing time of execution flows. Once the logs with anomalies have been identified automatically, through this strategy we can trace the causes and files that potentially triggered the errors. This could be useful for us to know in advance what the alarm messages are in real time and to act in time to prevent problems from occurring, that is, to do predictive maintenance.

Performance Metrics. To access the performance metrics of the ML techniques we have considered the following metrics. *Precision* is the measure of the model's performance with respect to false positives (FP), which are the messages identified as anomalies, when no anomalies occurred. False negatives (FN), on the other hand, are messages mislabeled with non-anomaly, which should instead be indicated as anomalous. From the expression: $precision = \frac{TP}{TP+FP}$, where true positives (TP) are messages that correctly report an occurred anomaly, it is evident that high precision values indicate a low rate of FP. *Recall* measures model's performance with respect to false negatives, according to this formula: $recall = \frac{TP}{TP+FN}$. f-measure, expressed by: $f1 = 2 \times \frac{precision \times recall}{precision + recall}$, is the harmonic mean of precision and recall. It is less affected by extreme values; the closer is its value to 1, the better the model performs.

4 Results

For this study, we have created a set of GitLab projects to store code we have implemented so far to perform the various phases of the presented approach. The collected and preprocessed data contains reserved information, therefore at the moment of the conference we have preferred to keep private our GitLab projects. We have developed our analysis on jupyter notebooks for python3 programming language by leveraging data analysis python libraries, such as pandas, nltk, and scikit-learn.

In the following we are going to show the effectiveness of the presented approach by showing the results obtained for a subset of services (see Table 1). The logs analyzed, after their preparation with the preprocessing phase, relate to the period June–October 2020. The causes of the anomalies detected are various: we can mention, for example, bugs related to the memory or silent corruptions of the data, jobs aborted due to field validation file, software downs or errors due to some users who use the data center services. We have grouped the extracted anomalies into specific message templates, so that they can be used as warnings in future system states without waiting further data acquisition time. Our approach is off-line: however, an on-line method is simulated by comparing the anomalous results of one month logging files with both the previous and the consecutive periods, and specific ML evaluation metrics are used. Next, the success is assessed through a manual check of the most frequent messages, in order to assign labels to certain messages and compare the experimental results with the real and expected ones. The precision, recall and F-measure performance metrics of ML techniques have been calculated through the data of two consecutive months. Table 4 shows the mean values of the ML performance metrics. We can assert that these models are effectively able to describe the anomalies.

Table 4. Mean values of the ML performance metrics.

Method	Precision	Recall	F-measure
Invariant mining	0.827	0.904	0.864
NLP word2vec	0.899	0.932	0.912
Autoencoder	0.819	0.925	0.895

All the methods highlight the same period of time for the anomalies due to the presence of problem key terms in the files. Messages, quite explanatory, labeled as anomalous by the methods are for example: "failing disk", "left power supply failure", recovered: "medium error disk", "internal queue full", "Total Queue full", "process abort" and "Disk Failed: Abort".

In the following we present details of the different methods.

Invariant Mining Model. The invariant mining technique is able to detect the intrinsic linear characteristics of workflows linked to a machine, in particular used to automatically collect the frequent patterns of error messages that generate a problem in the Tier-1 data center. The message-hostname pair has been chosen, because it is the most significant in the search for anomalies, and because other variables such as the ip address could be obtained from the hostname source. The percentage of anomalous messages, calculated between the number of unique suspicious messages of each month and the total number of monthly log messages, varies from a minimum of 0.06% to a maximum of 10.3%.

Figure 6 shows some examples of anomaly messages detected from logs of Tier-1, where the histograms report the number of occurrences per day. Their frequency is high at particular dates, and therefore can be associated with some problems. These patterns come from two specific hostnames and ips, which we indicate for privacy "host_a, ip_XXX.XXX.XXX.XY" and "host_b, ip_XXX.XXX.XXX.XZ", and therefore, this information is useful for monitoring the future operations of the specific associated machines.

Natural Language Processing Method. To establish the word2vec training set, we use the concatenation of 300 log files that constitute the basis of our model training. Therefore, the message corpus is relatively small. We have use the NLP method avoiding any optimization of the computation in order to get the maximum number of information. The output of the word2vec is a file containing the coordinates of distinct words of our training corpus.

Autoencoder Method. An autoencoder is a feed-forward multi-layer neural network with the same number of input and output neurons. We have used it to learn a more efficient representation of data while minimizing the corresponding error.

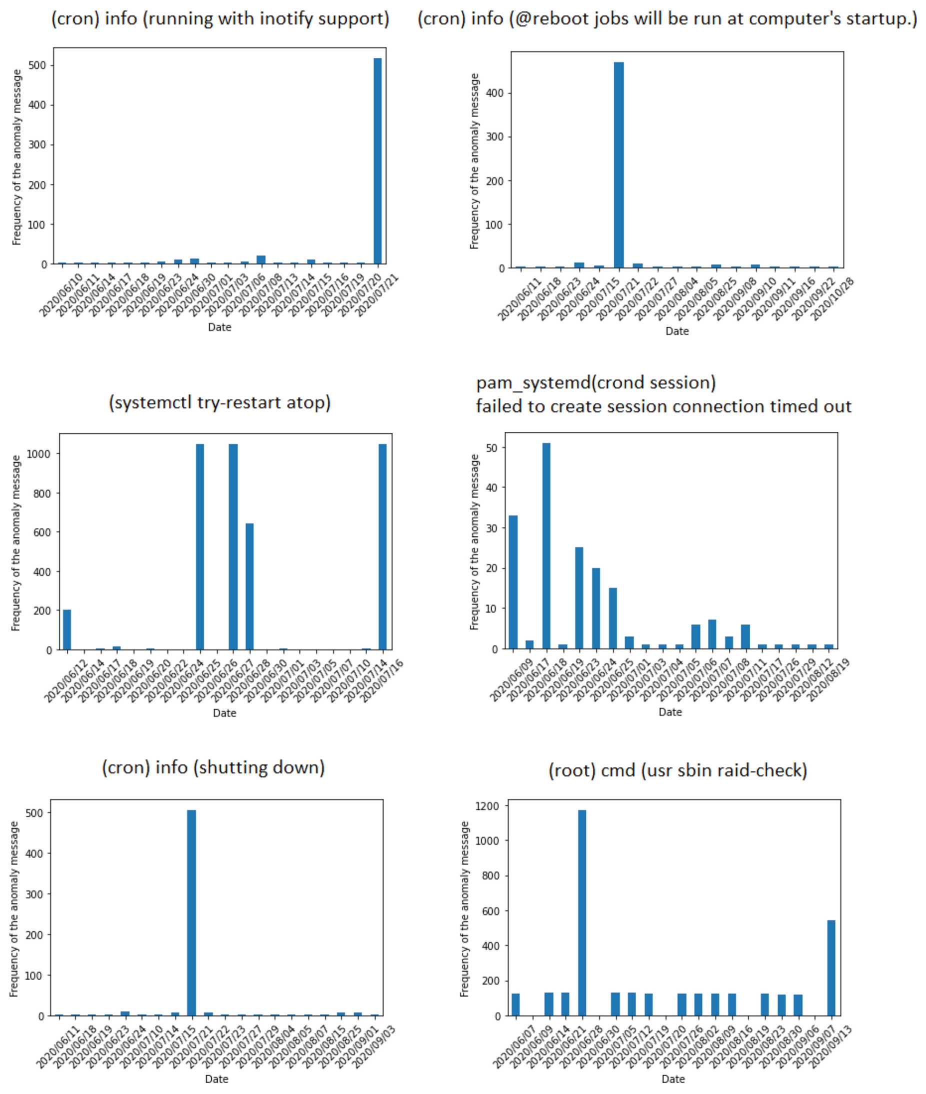


Fig. 6. Some anomaly messages detected with invariant mining.

5 Conclusions

The anomalies in logs can be related to permanent or transient errors. Performing manual diagnostics in a data center is not feasible, given the large-scale increase in unstructured information produced by machines in operation. For a human operator, even by dividing the data analysis for small time intervals, it would be very time-consuming. Most of the traditional tools on error analysis are based on verifying the standard behavior of the system by knowing a priori the behavior of specific software, but it is therefore linked to the specificity of the system and to the knowledge of the operators. If new software is used, manual analysis cannot be accurate after a short period of use, because in that case it is not possible to outline all the different normal system behaviors and tag the other unknowns as errors. If it were done for all the prototypes of unknown messages this could lead to a categorization of false positives.

In this work, the log mining is performed on monthly time slots and on the frequencies of each anomalous message, which are automatically extracted together with the hostnames registered to them. The results are promising. We have obtained an average of F-measure metric over 86%. A subset of services has been used during the study, therefore we aim at improving our study by considering all the available datasets.

Acknowledgements. This work has been supported by the IoTwins project N. 857191.

References

1. Bertero, C., Roy, M., Sauvanaud, C., Trédan, G.: Experience report: log mining using natural language processing and application to anomaly detection. In: 28th International Symposium on Software Reliability Engineering (ISSRE 2017). p. 10p. Toulouse, France (October 2017). https://hal.laas.fr/hal-01576291
2. Borghesi, A., Bartolini, A., Lombardi, M., Milano, M., Benini, L.: Anomaly detection using autoencoders in high performance computing systems. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 9428–9433 (July 2019). https://doi.org/10.1609/aaai.v33i01.33019428
3. Kim, K.J. (ed.): Information Science and Applications. LNEE, vol. 339. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46578-3
4. Breskin, R.V.A.: The CERN Large Hadron Collider: Accelerator And Experiments, vol. 2, CMS, LHCb, LHCf, And Totem. CERN (2009)
5. Cavallaro, C., Vitrià, J.: Corridor detection from large GPS trajectories datasets. Appl. Sci. **10**(14), 5003 (July 2020) https://doi.org/10.3390/app10145003
6. Chen, B., Jiang, Z.M.J.: Characterizing and detecting anti-patterns in the logging code. In: Proceedings of the IEEE/ACM 39th International Conference on Software Engineering (ICSE), pp. 71–81. IEEE Press (2017). https://doi.org/10.1109/ICSE.2017.15
7. Chen, Y., et al.: Outage prediction and diagnosis for cloud service systems. In: The World Wide Web Conference on - WWW 2019, ACM Press (2019). https://doi.org/10.1145/3308558.3313501
8. Collaboration, T.C., Chatrchyan, S., Hmayakyan, G., Khachatryan, V., Sirunyan, A.M., et al.: The CMS experiment at the CERN LHC. J. Instrum. **3**(08), S08004–S08004 (2008) https://doi.org/10.1088/1748-0221/3/08/s08004
9. Dai, H., Li, H., Chen, C.S., Shang, W., Chen, T.H.: Logram: efficient log parsing using n-gram dictionaries. IEEE Trans. Softw. Eng. 1 (2020). https://doi.org/10.1109/tse.2020.3007554
10. Das, A., Mueller, F., Rountree, B.: Aarohi: making real-time node failure prediction feasible. In: 2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS), IEEE (May 2020) https://doi.org/10.1109/ipdps47924.2020.00115

11. Davies, D.L., Bouldin, D.W.: A cluster separation measure. IEEE Trans. Pattern Anal. Mach. Intell. PAMI-1(2), 224–227 (1979). https://doi.org/10.1109/tpami.1979.4766909
12. dell'Agnello, L., et al.: Infn tier–1: a distributed site. EPJ Web Conf. **214**(08002), 01 (2019). https://doi.org/10.1051/epjconf/201921408002
13. Domingos, P., Pazzani, M.: On the optimality of the simple bayesian classifier under zero-one loss. Mach. Learn. **29**(2/3), 103–130 (1997). https://doi.org/10.1023/a:1007413511361
14. Du, M., Li, F.: Spell: Streaming parsing of system event logs. In: 2016 IEEE 16th International Conference on Data Mining (ICDM), IEEE (December 2016) https://doi.org/10.1109/icdm.2016.0103
15. Du, M., Li, F., Zheng, G., Srikumar, V.: DeepLog : anomaly detection and diagnosis from system logs through deep learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, ACM (October 2017) https://doi.org/10.1145/3133956.3134015
16. Dunn, J.C.: Well-separated clusters and optimal fuzzy partitions. J. Cybern. **4**(1), 95–104 (1974). https://doi.org/10.1080/01969727408546059
17. El-Masri, D., Petrillo, F., Guéhéneuc, Y.G., Hamou-Lhadj, A., Bouziane, A.: A systematic literature review on automated log abstraction techniques. Inf. Softw. Technol. **122**, 106276 (2020) https://doi.org/10.1016/j.infsof.2020.106276
18. ELK: Elasticsearch. https://www.elastic.co/elk-stack (2021). Accessed 11 Jun 2021
19. Examon: Examon HPC Monitoring. https://github.com/EEESlab/examon (2021). Accessed 11 Jun 2021
20. Farshchi, M., Schneider, J.G., Weber, I., Grundy, J.: Experience report: anomaly detection of cloud application operations using log and cloud metric correlation analysis. IEEE Trans. Softw. Eng. (2015). https://doi.org/10.1109/ISSRE.2015.7381796
21. Gainaru, A., Cappello, F., Trausan-Matu, S., Kramer, B.: event log mining tool for large scale HPC systems. In: Jeannot, E., Namyst, R., Roman, J. (eds.) Euro-Par 2011. LNCS, vol. 6852, pp. 52–64. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23400-2_6
22. Gerhards, R.: The syslog protocol. In: RFC. RFC Editor (2009)
23. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. ACM SIGMOD Record **29**(2), 1–12 (2000). https://doi.org/10.1145/335191.335372
24. He, P., Chen, Z., He, S., Lyu, M.R.: Characterizing the natural language descriptions in software logging statements. In: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering ASE, pp. 178–189 (2018). https://doi.org/10.1145/3238147.3238193
25. He, P., Zhu, J., He, S., Li, J., Lyu, M.R.: Towards automated log parsing for large-scale log data analysis. IEEE Trans. Dependable Secure Comput. **15**(6), 931–944 (2018). https://doi.org/10.1109/tdsc.2017.2762673
26. He, P., Zhu, J., Zheng, Z., Lyu, M.R.: Drain: An online log parsing approach with fixed depth tree. In: 2017 IEEE International Conference on Web Services (ICWS), IEEE (2017) https://doi.org/10.1109/icws.2017.13
27. He, S., He, P., Chen, Z., Yang, T., Su, Y., Lyu, M.R.: A survey on automated log analysis for reliability engineering. ArXiv (September 2020)
28. Jaccard, P.: The distribution of the flora in the alpine zone. New Phytol. **11**(2), 37–50 (1912). https://doi.org/10.1111/j.1469-8137.1912.tb05611.x

29. Jia, T., Yang, L., Chen, P., Li, Y., Meng, F., Xu, J.: LogSed: anomaly diagnosis through mining time-weighted control flow graph in logs. In: 2017 IEEE 10th International Conference on Cloud Computing (CLOUD), IEEE (2017). https://doi.org/10.1109/cloud.2017.64
30. Tan, J., Pan, X., Kavulya, S., Gandhi, R., Narasimhan, P.: Salsa: analyzing logs as state machines (cmu-pdl-08-111). In: First USENIX Workshop on the Analysis of System Logs, WASL 2008, San Diego, CA, USA, Proceedings. Carnegie Mellon University (2008). https://doi.org/10.1184/R1/6619766
31. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: International Conference on Learning Representation (ICLR) (2015)
32. Kohonen, T.: Self-organized formation of topologically correct feature maps. Biol. Cybern. **43**(1), 59–69 (1982). https://doi.org/10.1007/bf00337288
33. Layer, L., et al.: Automatic log analysis with NLP for the CMS workflow handling. In: 24th International Conference on Computing in High Energy and Nuclear Physics (CHEP 2019), p. 7 (November 2020) https://doi.org/10.1051/epjconf/202024503006
34. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions and reversals. Doklady Akademii Nauk SSSR **163**(4), 845–848 (1965)
35. Lloyd, S.: Least squares quantization in PCM. IEEE Trans. Inf. Theor. **28**(2), 129–137 (1982). https://doi.org/10.1109/tit.1982.1056489
36. Loggly: Loggly - log management by loggly. https://www.loggly.com (2021). Accessed 11 Jun 2021
37. Lou, J.G., Fu, Q., Yang, S., Xu, Y., Li, J.: Mining invariants from console logs for system problem detection. In: Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference, USENIXATC 2010, p. 24, USENIX Association, USA (2010)
38. Makanju, A., Zincir-Heywood, A.N., Milios, E.E.: A lightweight algorithm for message type extraction in system application logs. IEEE Trans. Knowl. Data Eng. **24**(11), 1921–1936 (2012). https://doi.org/10.1109/tkde.2011.138
39. Mizutani, M.: Incremental mining of system log format. In: 2013 IEEE International Conference on Services Computing, IEEE (June 2013) https://doi.org/10.1109/scc.2013.73
40. Oliver, R.: What supercomputers say: a study of 5 system logs. In: Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2007), IEEE Press (2007). https://doi.org/10.1109/DSN.2007.103
41. OverOps: OverOps Continuous Reliability Solution. https://www.overops.com/ (2021). Accessed 11 Jun 2021
42. Quinlan, J.: Simplifying decision trees. Int. J. Man-Mach. Stud. **27**(3), 221–234 (1987). https://doi.org/10.1016/s0020-7373(87)80053-6
43. Rand, W.M.: Objective criteria for the evaluation of clustering methods. J. Am. Stat. Assoc. **66**(336), 846–850 (1971). https://doi.org/10.1080/01621459.1971.10482356
44. Sander, J., Ester, M., Kriegel, H.P., Xu, X.: Density-based clustering in spatial databases: The algorithm dbscan and its applications. Data Min. Knowl. Discov. **2**(2), 169–194 (1998). https://doi.org/10.1023/a:1009745219419
45. Splunk: Splunk platform. http://www.splunk.com (2005-2021). Accessed 11 Jun 2021
46. Srikant, R., Agrawal, R.: Mining generalized association rules. Future Gener. Comput. Syst. **13**(2–3), 161–180 (1997). https://doi.org/10.1016/s0167-739x(97)00019-8

47. Tomas, M., Ilya, S., Kai, C., Greg, C., Jeffrey, D.: Distributed representations of words and phrases and their compositionality. In: Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS 2013), NIPS 2013, pp. 3111–3119. Curran Associates Inc., Red Hook, NY, USA (2013)
48. Ullman, J.D., Aho, A.V., Hirschberg, D.S.: Bounds on the complexity of the longest common subsequence problem. J. ACM **23**(1), 1–12 (1976). https://doi.org/10.1145/321921.321922
49. Vaarandi, R.: Mining event logs with SLCT and LogHound. In: NOMS 2008–2008 IEEE Network Operations and Management Symposium, IEEE (2008). https://doi.org/10.1109/noms.2008.4575281
50. Vaarandi, R., Pihelgas, M.: LogCluster - a data clustering and pattern mining algorithm for event logs. In: 2015 11th International Conference on Network and Service Management (CNSM), IEEE (November 2015) https://doi.org/10.1109/cnsm.2015.7367331
51. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, New York (1995). https://doi.org/10.1007/978-1-4757-3264-1
52. Xia, B., Bai, Y., Yin, J., Li, Y., Xu, J.: LogGAN: a log-level generative adversarial network for anomaly detection using permutation event modeling. Inf. Syst. Front. **23**(2), 285–298 (2020). https://doi.org/10.1007/s10796-020-10026-3
53. Xu, W., Huang, L., Fox, A., Patterson, D., Jordan, M.I.: Detecting large-scale system problems by mining console logs. In: Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles - SOSP 2009. ACM Press (2009). https://doi.org/10.1145/1629575.1629587
54. Yuan, D., Mai, H., Xiong, W., Tan, L., Zhou, Y., Pasupathy, S.: SherLog: error diagnosis by connecting clues from run-time logs. ACM SIGARCH Comput. Architect. News **38**(1), 143–154 (2010). https://doi.org/10.1145/1735970.1736038
55. Yuan, D., Zheng, J., Park, S., Zhou, Y., Savage, S.: Improving software diagnosability via log enhancement. In: Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems - ASPLOS 2011. ACM Press (2011). https://doi.org/10.1145/1950365.1950369
56. Zhang, C., et al.: A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. ArXiv arXiv:1811.08055 (2019)