Anomaly Detection and Root Cause Analysis on Log Data

Daem Pasha, Ali Hussain Shah, Esmaeil Habib Zadeh, and Savas Konur(✉)

Department of Computer Science, University of Bradford, Bradford BD7 1DP, UK
s.konur@bradford.ac.uk

Abstract. In this paper, we perform anomaly detection and root cause analysis on log data (system logs). Firstly, we employ a log parsing solution known as Drain (an online log parsing approach with fixed depth tree). We then present an anomaly detection approach that utilizes a decision tree model. This will be used to determine the anomalous devices in the log files. One benefit of decision tree models is that they are easily traceable, providing a contrast to most "black-box" solutions currently available in the industry. Finally, a sequential model using Keras is built to predict the root cause of a given issue.

Keywords: Root cause analysis · Anomaly detection · Log files · Machine learning

1 Introduction

Logging is a common practice in the IT industry and can provide useful information for anomaly detection. Logs keep the historical records of everything that happens in a system and can cover many types of events, e.g., transactional, error, and intrusion-based events.

Anomaly detection can be used to detect anomalous patterns in the logs. The problem with the current commercial anomaly detection solutions is that they are essentially a black box which makes it difficult to see what the algorithm is doing behind the scenes [3]. Another limitation is that they only focus on new/unique log events.

Root cause analysis (RCA) is useful considering that a single issue can cause a snowball effect leading to an influx of log records. RCA can be performed and used to determine the source of the issue without the need to trawl through thousands of log entries. A problem with the providers of these solutions, e.g., Zebrium, requires all your log data to be on their platform. This process can be costly as frequent cloud storage and data insertion can quickly increase the price.

In this paper, we propose a transparent anomaly detection method that utilizes Drain as log parsing (an online log parsing approach with fixed depth tree, which has a 51.85–81.47% improvement in run-time when compared with state-of-the-art online parsers) and a decision tree model for anomaly detection. One benefit of using decision tree models is that they are easily traceable, providing a contrast to most "black-box" solutions currently available in the industry. We will also look at other important issues such

https://doi.org/10.1007/978-3-031-21441-7_27

as the frequency of the events to determine an anomalous appliance. We also use a sequential model using Keras to predict the root cause of a given anomaly. Our proposed solution is independent from any storage platform making it much more cost-effective and integrable with existing architecture.

2 Methodology

2.1 Data

The datasets used for this solution mainly contain log files and some network traffic metric data (see Table 1) [4]. The root cause analysis data file is used to train the RCA model. All these datasets have been obtained from a network traffic management company.

Table 1. Dataset description.

Name	Type	Train/test	Size	Anomaly:normal ratio
Syslogs (Linux system logs)	Log file	50/50	944	148:796
Root cause analysis data	Log file	50/50	1000	N/A

2.2 Methodology Outline

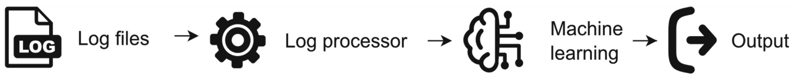

Fig. 1. Overall method diagram.

In the first stage, the *log processor* (Fig. 1) takes raw, unstructured logs and converts them into a consistent and structured format. For this process, we use Drain [1] which takes the log files and a template string (e.g., <Date> <Appliance> <Message>) as input and returns a CSV file with the data split into individual columns as output.

The output from the log parser is then fed into the *machine learning* module (Fig. 1) which utilises a decision tree model to determine anomalous patterns in the log files. We then output a list of anomalous appliances by utilising the TF-IDF measure (Sect. 2.4). Finally, we take into consideration multiple log files and metrics and pass them into a sequential model to determine the root cause of the failure.

2.3 Log Parsing

Drain is an open-source log parser that can parse logs with high accuracy and speed [2]. It categorizes logs by assigning them to a template and then extracts the parameters from the logs, making it very useful for machine learning. Drain also streams logs as it parses them. This means it does not require an offline training step and it is not limited by memory.

When a new log arrives, Drain uses regular expressions to pre-process the log. These regular expressions are provided by the system experts in terms of domain knowledge. We can then construct a tree and decide on the best group, based on the length of the logs. If the group does not exist, a new group is created [2].

Next, Drain further groups these logs by picking the first word from the log, for example, all words beginning with “Receive” are placed into the same group [1]. Finally, at the leaf node we see a list of different log groups and the log is placed in the most closely related group.

Table 2. MySQL log parser results.

Metric	Value
Precision	1.0
Recall	1.0
F-measure	1.0

Table 3. Syslogs log parser results.

Metric	Value
Precision	0.984
Recall	0.755
F-measure	0.992

Tables 2 and 3 show the F-measure values obtained from these log files, both of which come from live environments. This shows a good potential for our approach.

2.4 Anomaly Detection

For anomaly detection, we use the decision tree algorithm. This is because the decision tree model is much more interpretable as the thought process in the previous nodes leading up to the anomaly [3]. This will provide much more detail and insight which will be useful when diagnosing potential issues with the output.

To determine the best possible split for the tree, the model makes use of the Gini index. The next step is to split the data into the train and test datasets. His is a way to evaluate the performance of a model by using them to make predictions on data not involved in the training process. For the syslogs dataset, a train-test split of 50/50 has been selected. The train dataset is used to fit the model whilst the test dataset is used to test the model. This prevents bias as the same data is not being used twice, which could cause overfitting.

Next, we make use of a technique called *fit transform* to convert our dataset into something usable by the model. In this case for the decision tree, we will need to convert them into their numerical representation [5]. This process is called *text vectorization.*

To determine whether an appliance is anomalous we make use of a numerical statistic called TF-IDF (Term frequency-inverse document frequency). TF will increase proportionally, whereas IDF offsets that number. Events that are common for every block/appliance do not rank as high, however, an event that occurs multiple times in one block/appliance but does not occur in others would rank higher [6].

Finally, we can obtain the F-measure score for the model which can be used to determine the accuracy and reliability of the model.

Table 4. Syslog anomaly detection accuracy results.

Metric	Value
Precision	0.6
Recall	1.0
F-measure	**0.75**

Table 5. Decision tree parameters.

	Value
Max depth	∞
Criterion	Gini
Min sample split	**2**

Table 4 shows the results of the anomaly detection when applied to network traffic monitoring data. Table 5 shows the decision tree parameters. An F-measure score of 0.75 is desirable in this scenario as the dataset is only 1000 lines. In a real-world scenario, the dataset would be consuming a lot more data, which is expected to increase the accuracy as there is more train data.

2.5 Root Cause Analysis

For root cause analysis, we take our anomalous appliances from the anomaly detection and combine them with other sources of information (i.e., other log files or metric data). The more datasets there are, the more accurate the prediction is.

To classify the data, we need to convert our list of root cause outcomes (e.g., "MEMORY_ISSUE") into numerical values. This is a crucial step to obtain good results from our data [7]. A label encoder is used to fit and transform the root cause outcomes into numerical values. For this implementation, the Sequential model is used from the Keras framework. This is a machine learning model that is built up of many neural network layers.

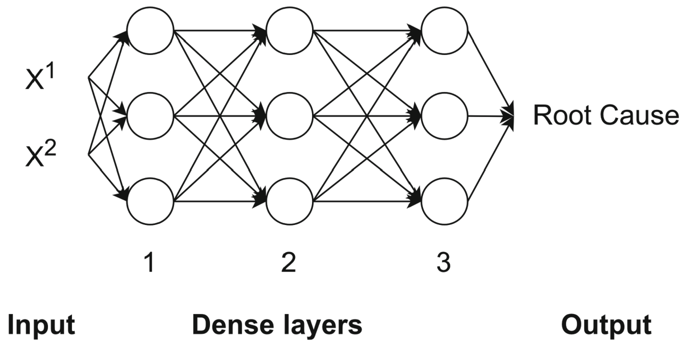

Fig. 2. Root cause analysis process.

Figure 2 shows the RCA process. Firstly, the input layer takes the values from the dataset (either 1 or 0). We then apply two dense layers with a ReLU (Rectified Linear

Unit) activation function; these layers have 128 neurons and take each of the previous inputs in.

Finally, we apply a dense layer with the SoftMax activation; this converts the inputs from the previous dense layer and makes a categorical prediction based on the number of root cause classes.

Table 6. Input data/parameters for RCA model.

Parameters	Value
Amount of data	148 lines
Number of categorisation classes	4
Epoch	20
Batch size	100
Validation split	50%
Optimizer	adam
Learning rate	0.01

Table 7. Results of RCA model after training on network dataset.

	Train	Test
Epoch	Acc.	Acc.
1	0.40	0.63
2	0.66	0.65
…	…	…
12	0.79	0.78
13	**0.80**	**0.78**

Table 6 shows the input parameters fed to the RCA model and Table 7 shows the results of the RCA solution after being trained with 13 epochs on 1000 lines of data with a 50% split ratio. In the future, this will be trained on a larger dataset which is likely to improve the accuracy. Figure 3 shows the class distribution for the four potential categories (root causes).

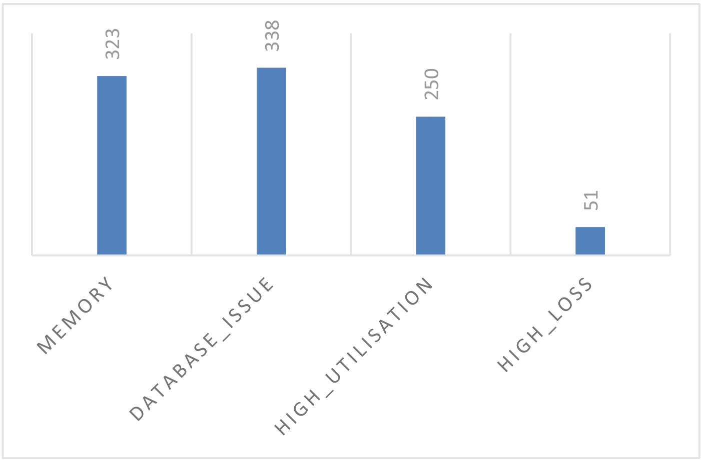

Fig. 3. Class distribution.

3 Conclusion and Future Work

This paper outlines a process that can be used to transform a number of logs into a much smaller and more useful output using three main components: log parser, anomaly detection, and root cause analysis.

The log parser is very effective and accurate with an F-measure score of 0.99 when applied to the syslogs dataset (Table 3). It also has some capabilities to detect and automatically parse log templates by checking the similarity threshold between them.

The anomaly detection algorithm is also effective in discovering anomalies as well with an F-measure score of 0.75 when applied to the syslogs dataset (Table 4). It is also easily viewable due to the transparent nature of the decision tree model. This helps when understanding the logic behind the decision. The use of a rule-based labeling system also greatly decreases the initial time taken to collect and prepare the data allowing for the use of a supervised learning algorithm without the initial delay that comes with data processing.

Root cause analysis has an accuracy score of 0.78 for the train dataset (Table 7), this is a good result for the current stage. Overall, there is a strong foundation for root cause analysis. The log parsing, anomaly detection, and root cause models show good results when applied to real-world datasets.

In future, we will consider the feasibility of our approach in very large datasets. In particular, we will apply our approach to use cases requiring high volume of real-time data traffic and logs such as autonomous vehicles [8], smart manufacturing [9], and ubiquitous systems [10, 11] and as well as scientific software systems [12, 13].

Acknowledgements. The authors acknowledge Innovate UK [KTP012139] and EPSRC (EP/R043787/1).

References

1. 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), pp. 33–40
2. He, S., Zhu, J., He, P., Lyu, M.R.: Experience report: system log analysis for anomaly detection. In: 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE), pp. 207–218
3. He, P., Zhu, J., He, S., Li, J., Lyu, M.R.: An evaluation study on log parsing and its use in log mining. In: 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 654–661
4. 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. In: Proceedings of 11th Int. Conference on CECNet, Frontiers in Artificial Intelligence and Applications, vol. 345,pp. 659–666 (2022)
5. Kumari, A., Shashi, M.: Vectorization of text documents for identifying unifiable news articles. Int. J. Adv. Comput. Sci. Appl. **10**, 305 (2019)
6. Salton, G., Buckley, C.: Termweighting approaches in automatic text retrieval. Inf. Process. Manage. **24**(5), 513–523 (1988)

7. Sola, J., Sevilla, J.: Importance of input data normalization for the application of neural networks to complex industrial problems. IEEE Trans. Nucl. Sci. **44**, 1464–1468 (1997)
8. Badue, C., et al.: Self-driving cars: a survey. Expert Syst. Appl. **165**, 113816 (2021)
9. Konur, S., et al.: Towards design and implementation of Industry 4.0 for food manufacturing. Neural Comput. Appl. 1–13 (2021)
10. Arapinis, M., et al.: Towards the verification of pervasive systems. Electron. Commun. EASST 22 (2010)
11. Konur, S., Fisher, M., Dobson, S., Knox, S.: Formal verification of a pervasive messaging system. Formal Aspects Comput. **26**(4), 677–694 (2013)
12. Blakes, J., et al.: Infobiotics workbench: A P systems based tool for systems and synthetic biology. Appl. Membr. Comput. Syst. Synth. Biol. Emerg. Complex. Comput. **7**, 1–41 (2014)
13. Bakir, M., et al.: Extended simulation and verification platform for kernel P systems. Int. Conf. Membr. Comput. LNCS **8961**, 158–178 (2014)