Network Anomaly Detection Based on Logs

Longsheng Du, Yu Du and Ke Wu
Team Ember

ABSTRACT

Anomaly detection is a critical part for telecom carrier companies to build a robust system. Hundreds of thousands of devices can produce a large amount of log data. The device status and potential problems can be detected using extracted log data. However, the log data is often not structured nor uniformed, so the traditional method of keyword or regular expression matching to manually check is inefficient and costly. Thus, the competition challenges participants to analyze log data gathered from network equipment and detect anomaly using AI technique to provide inspiration to improve efficiency and reduce operation costs. This team carefully analyzes the problem and dataset given, designs and implements suitable and versatile algorithms based on fixed depth tree log parsing and LSTM unsupervised machine learning model for log-based anomaly detection problems and achieves state-of-the-art accuracy on the test dataset provided. The algorithms also execute efficiently on the competition's coding platform with great modular code organization for review and deployment.

KEYWORDS

Anomaly detection; Log analysis; LSTM; Deep learning

I. PROBLEM ANALYSIS

The data is gathered from network equipment, the data contains time stamp, log information and can be grouped into 5-minute time slice. There are over a million lines of logs and all of them are unstructured and in plain text form. The two kinds of log, *sysmonitor* and *messages,* have separate format, and each divided into training set and test set. Traditionally, the problem can be solved using keyword or regex matching, but this kind of method is not applicable for generalized problem solving which is what the challenge intended. Thus, the problem can be break up into two steps: make sense of the data using log parsing and detect the anomaly using machine learning. Furthermore, training set only contains normal data while test set contains abnormal data, so unsupervised learning is necessary here.

II. ALGRITHMS DESIGN

The algorithms can be described in two steps: log analysis and machine learning detection. First step is to make sense of the data, structuralize the plain text data for further analysis and the second step is to use unsupervised machine learning to train a detection model and use the model to classify the anomaly data.

A. Log Analysis

Traditionally, log analysis can be performed manually, which highly depended on individuals'

experience and is prone to human error. Keywords or regular expression matching is also used, this also requires creating keywords manually and has high chance to be inaccurate, while the keywords will only be applicable to one specific dataset.

Automatic template parsing is needed here in order to perform generalized log analysis. Fixed depth tree parsing is a great fir here due to its efficiency and accuracy.

Parse tree with fixed depth can guide log group search when new text data needs to be parsed, which effectively compare raw log message with log groups and decide which is the most suitable log group for it.

The parsing can be done in following steps:

Preprocess: Use regular expressions based on domain knowledge that represent commonly used variables, such as IP address and numbers to remove the tokens matched from the raw log message.

Search by length: Select a path to a 1-st layer node based on the log message length of the preprocessed log message.

Search by token: Select the next internal node by the tokens in the beginning positions of the log message.

Search by similarity: Select the most suitable log group from the log group list by calculating the similarity between the log message and the log template of each log group.

Update parse tree: Scan the tokens of the log message and the log template. If the tokens are not same, we update the token position by wildcard in the log template.

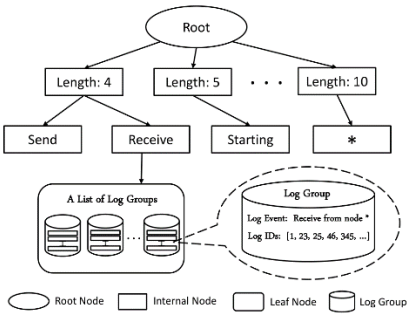

Figure 2.1 Structure of Parse Tree [1]

The structure shown in Figure 2.1 represent the parse tree. This allows the online paring for millions of log entries to be accurately parsed as its templates of certain log group.

B. Machine Learning Detection

After log analysis, we get structured data and now can use this information to represent original data and perform detection using unsupervised machine learning method.

Traditionally, anomaly detection model can be created using clustering, PCA or isolation forest methods. However these method faces some drawbacks: log clustering can be fast at predicting after training when few clusters found, but not effective if anomalies don't form significant clusters; PCA works better with numeric data and needs lot of resources to scale to very large dataset; isolation forest can be fast at predicting after training when tree is built, but branching bias may occur depending on how the tree is built and score depends on the contamination parameter which implies the percentage of the data is anomalous is known beforehand.

Model system log data as natural language sequence and utilizing Long Short-Term Memory neural network model (LSTM) can produce a great detection model, captures the potentially non-linear

and high dimensional feature among log entries from the training data in an unsupervised way that correspond to normal system execution paths.

LSTM processes sequences using three types of operational gates, forget gate (decide what information to discard from cell), input gate (decide what information to add to cell), output gate (decide what information in cell to output). This design provides the fundamental to learn which data in a sequence is important to keep or throw away and pass relevant information down the long chain of sequences to make time-series sequence predictions.

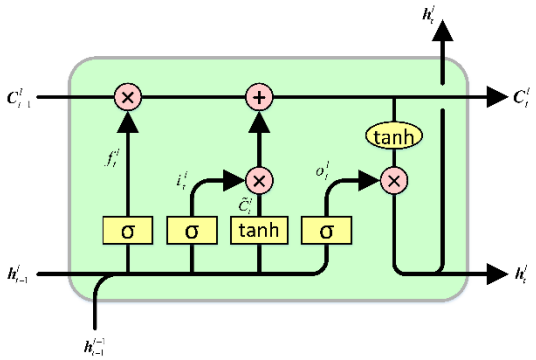

Figure 2.2 LSTM cell [2]

Training data for the model are from training dataset's each time window within each time slice. Each log entry is parsed to a log label. The log key sequence parsed from a training log data is used by model to train a log label prediction.

Detection is performed by parsing a new log entry and predict this new entry using former sequence, if new label is not as predicted, anomaly is likely happened.

III. CODE INPLEMENTATION

The code project is consisted of 6 parts:

Entry script: Platform required. Contains main function for program to call different module and get result.

System IO: Perform the file input/output according to platform interface.

Log parser: Performs fixed depth tree parsing based on Drain to extract template.

Log feature: Extract template id and assign index, divide time slice, and create sequence dataset for LSTM model.

Log trainer: Using sequence dataset as input and label at each window as true output. Train the LSTM neural network model using PyTorch so the model can predict next input label using former sequence.

Log tester: Take new sequence data and predict outcome, if label in time window is not in predicted candidates, the series is abnormal.

Parameter	Value	Parameter	Value
Window size	10	Batch size	2048
Hidden size	64	Max epoch	150
Num layers	2	Learning rate	0.001
Num candidates	0.35	Optimizer	adam

Table 3.1 The parameters used for *sysmonitor* model

Parameter	Value	Parameter	Value
Window size	10	Batch size	1024
Hidden size	64	Max epoch	6
Num layers	2	Learning rate	0.001
Num candidates	0.32	Optimizer	adam

Table 3.2 The parameters used for *messages* model

IV. RESULT

The project is deployed on designated coding platform and use the resource provided with 4-core CPU and 8G of RAM. The execution environment uses Python 3.7 and PyTorch 1.4.0. All data is fetch using platform interface and the result is auto scored inside the platform.

Score	Value
F1 Score	0.8205
Parsing speed	0.5 millisecond per log entry
Training speed	12 second per epoch
Predicting speed	65 millisecond per time slice

Table 4.1 Final score

V. CONCLUSION

In this report, we present how the problem is analyzed effectively using AI knowledge by the participated team, detailing how accurate log parsing method and LSTM detection model are designed and implemented to achieve stat-of-the-art accuracy. The algorithms and parameters are also optimized for faster execution while the code base is well-organized and designed modularly for easy deployment.

Interesting future directions would be utilized the content alongside template to provide contextual information or use multiple algorithms to create fusion or boost methods. In real production environment, network topological information can also play an important role to locate faulty devices.

REFERENCES

[1] He, Pinjia, et al. "Drain: An online log parsing approach with fixed depth tree." 2017 IEEE international conference on web services (ICWS). IEEE, 2017.

[2] Nasser, Ahmed Abdel, Magdi Z. Rashad, and Sherif E. Hussein. "A two-layer water demand prediction system in urban areas based on micro-services and LSTM Neural Networks." IEEE Access 8 (2020): 147647-147661.

[3] Du, Min, et al. "Deeplog: Anomaly detection and diagnosis from system logs through deep learning." Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017.