DATA586 Project Report: Hadoop Log Data Anomaly Detection

YILIN SUN and SONG ZHANG

ACM Reference Format:
Yilin Sun and Song Zhang. 2022. DATA586 Project Report: Hadoop Log Data Anomaly Detection. 1, 1 (May 2022), 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 ABSTRACT

Our research is based on a log data file that contains the Hadoop log file generated by a Hadoop cluster. We extract necessary information and build a model to do the anomaly detection.

We extract several features and build a model to do classification on the data to find the anomaly logs. We use logistic regression, random forest classifier and Multilayer perceptrons in the research.

2 INTRODUCTION

In the research, the logs are generated from a Hadoop cluster with 46 cores across five machines. Each machine has Intel(R) Core(TM) i7-3770 CPU and 16GB RAM. We parsed the log file and extract several features including the date, time, label, function, bracket information, and so on. The label includes INFO,WARNING and ERROR information, which is used to be the classification result for our research. And the rest of the feature are used as the feature for building and training a model. There are more than 12K rows data, we use 15k rows data for our modeling.

Our research includes an EDA(Exploratory Data Analysis) that shows the visualization of the error logs. For the modeling part, first we do some feature engineering. We change the label to integer, and extract the bracketsInfo, func, hasError and hasException as the feature for the modeling. We apply one-hot-encoding to all of the features. In the modeling part, we use logistic regression, random forest classifier and multilayer perceptron as the method.

3 LITERATURE REVIEW

Logs are often generated for troubleshooting software systems. The log messages are often text strings with the information of the events or states of interest for the systems. Generally, engineers can locate the causes or gain insight about the job failures by examining the log files. Especially for large-scale online services running with multiple clusters of servers and data centers in a Big Data environment, logging is much more important as it is difficult to apply other software debugging techniques. For instance, it is impractical for an online service system to attach a debugger.

As of this, logging has been universally used in software systems. Like Hadoop[1] uses logs to record changes. Microsoft Engineers[6] use logging to mining historical issue repositories[4]. The log messages contains strings and parameters recording system status and system attributes. The first step towards the analysis is to parse the log messages[7], [15]. Then the Inverse Document Frequency of each log event can be calculated[9].

Authors' address: Yilin Sun; Song Zhang.

There are many research works analyzing logs for problem diagnosis. Lou et al.[8] mined constant linear relationships as invariant from console logs. If a new log message breaks certain invariant during the system running, a service anomaly will be detected. Xu et al.[15] used principal component analysis (PCA) to preprocess the logs and detect anomalies. However, the log-based anomaly detection algorithms cannot obtain the insights into the abnormal task, it can only check whether a service is abnormal or not. Yuan et al.[17] aims to improve software diagnosability via log enhancement. It automatically identified and inserted critical variable values into the recorded contents in existing logging statements. Fu et al.[5] records each request's runtime properties in a multi-tier Web server, and applies statistical learning techniques to identify the reasons of failures.

There are also some log-based diagnosis work which is based on the similarity among log sequences. Dickenson et al.[2] used cluster analysis of excution profilesto find failures. They classified the collected traces based on some string distance metrics by collecting execution traces and using classification techniques. Thereafter, engineers can examine the traces of each category and determine whether the category represents an anomaly or not. Yuan et al.[17] put forward a supervised classification algorithm to classify system traces based on the similarity to the traces of the problems. Mirgorodskiy et al.[10] used string distance metrics to categorize function-level traces, and to identify outlier traces or anomalies that substantially differ from the others. Ding et al.[3][4] designed a framework to correlate logs, system issues, and corresponding simple mitigation solutions when similar logs appear. Lin[14] considered weights of different events and applied hierarchical clustering to cluster similar log sequences. They compared the newly obtained log sequences with those of known failures. It facilitated problem identification for online service systems by clustering similar logs.

In addition, there are a lot of work conducted to understand the log messages and logging practices. Yuan et al.[16], Shang et al.[12], and Fu et al.[6] reported empirical studies on logging practice in open source and industrial software. Zhu et al.[18] proposed a learning to log framework, which targets to provide informative guidance on logging. On top of that, Shang et al.[11] used a sequence of logs to provide context information to examine log messages. To facilitate the understanding of log messages, Shang et al.[13] further proposed to associate the development knowledge stored in various software repositories with the log messages.

4 BACKGROUND

In the background part, we will introduce Hadoop, which is generate the log data we used. In addition, we will also include scikit-learn and PyTorch, the tools we used to build and train our model.

4.1 Hadoop

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

4.2 scikit-learn

Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting,

k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

4.3 PyTorch

PyTorch is an open source machine learning framework based on the Torch library, used for applications such as computer vision and natural language processing,primarily developed by Facebook's AI Research lab (FAIR). PyTorch defines a class called Tensor to store and operate on homogeneous multidimensional rectangular arrays of numbers. PyTorch Tensors are similar to NumPy Arrays, but can also be operated on a CUDA-capable Nvidia GPU. PyTorch supports various sub-types of Tensors.

5 METHODOLOGY

In this part, we will introduce our dataset, the data pre-processing procedure and the method we use to build the model.

5.1 Hadoop Log File

Our dataset is Hadoop log file. The logs are generated from a Hadoop cluster with 46 cores across five machines. For each row of data, it contains the date, time, label, places, functions and the messages of the log.

```
2015-10-18 18:01:49,228 INFO [main] org.apache.hadoop.mapreduce.v2.app.MRAppMaster: Using mapred newApiCommitter.
2015-10-18 18:01:50,353 INFO [main] org.apache.hadoop.mapreduce.v2.app.MRAppMaster: OutputCommitter set in config null
```

Fig. 1. An example of the log file

From Fig.1, we can see that the date of the log file is October 18, 2018, and these two logs are generated in 18:01 of that date. Both of the log are information log and both of them are in main place and the function are in hadoop mapreduce. The first log contains the information: Using mapred newApiCommitter, while the second log contains the information: OutputCommitter set in config null.

Next, as the data need to pre-processing and we think the best way is store the useful information in a dataframe, we will introduce these processes in the next data processing part.

5.2 Data Pre-Processing

First, we combine the log files together and store all the log files in a file named combined.log. We then read the log file content and store the information in a dataframe. We extract the date, hour, log type, information in the bracket, and the function(which is after the .hadoop) of the log. In addition, we determine if the log contains the key word "Error" or "Exception" and store the result in two boolean columns. Figure 2 shows the first few lines of the dataframe we generated for modeling.

For the data, there are 120519 lines with INFO label, 6164 lines with WARN label and 336 lines with ERROR label. We extract 15000 rows for modeling, which include all the WARN and ERROR label lines and the rest is filled with the INFO label lines. For the INFO lines, we just random select enough lines.

5.3 Exploratory Data Analysis

For the Exploratory Data Analysis(EDA), we focus on the logs that with ERROR label. We found that most error label appears in mapreduce function and appear at 2015-10-18. For the details information in the bracket, the error is mainly due to RMCommunicator Allocator.

	date	hour	label	bracketsInfo	hasError	hasException	func	message
0	2015-10-17	15:37:56,547	INFO	main	False	False	mapreduce	[[main], org.apache.hadoop.mapreduce.v2.app.MR...
1	2015-10-17	15:37:56,899	INFO	main	False	False	mapreduce	[[main], org.apache.hadoop.mapreduce.v2.app.MR...
2	2015-10-17	15:37:56,900	INFO	main	False	False	mapreduce	[[main], org.apache.hadoop.mapreduce.v2.app.MR...
3	2015-10-17	15:37:57,036	INFO	main	False	False	mapreduce	[[main], org.apache.hadoop.mapreduce.v2.app.MR...
4	2015-10-17	15:37:57,634	INFO	main	False	False	mapreduce	[[main], org.apache.hadoop.mapreduce.v2.app.MR...

Fig. 2. First few lines of the dataframe

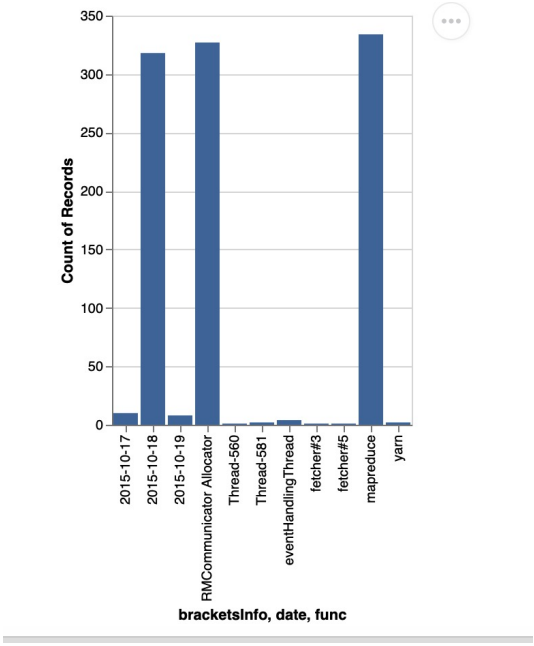

Fig. 3. Error Count result

5.4 Feature Engineering

We did some feature engineering for building and training model. Feature engineering refers to the process of using domain knowledge to select and transform the most relevant variables from raw data when creating a predictive model using machine learning or statistical modeling. We change the label from string to integer, and extract the bracketsInfo, func, hasError and hasException as the feature for the modeling. We apply one-hot-encoding to all of the features.

5.5 Logistic Regression

Logistic regression is a statistical analysis method to predict a binary outcome, such as yes or no, based on prior observations of a data set. A logistic regression model predicts a dependent data variable by analyzing the relationship between one or more existing independent variables. As the logistic regression is the basic method for modeling, so we use logistic regression first to see the accuracy of the model.

5.6 Random Forest

Random forest is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. It is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. We use random forest classifier and to see if we can improve the accuracy for the model.

5.7 Feedforward Neural Network

A feedforward neural network is an artificial neural network wherein connections between the nodes do not form a cycle. This class of networks consists of multiple layers of computational units, usually interconnected in a feed-forward way. Each neuron in one layer has directed connections to the neurons of the subsequent layer.

An MLP (Multilayer perceptron) consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.

We use Feedforward Neural Network in the last to see the accuracy of the model.

6 EXPERIMENT AND RESULTS

6.1 Data Split

We separate the dataset to training and testing set. 80 percents of data are used for training and 20 percents of data are used for testing.

6.2 Logistic Model

For the logistic model, it is a straight forward model and we use the function in scikit-learn to run the model. We iterate the training data set for maximum 10 rounds.

For the result, the MSE of logistic model is 0.01 and the accuracy on the test set is 0.994. It is a higher accuracy rate. This may due to we use the correct features that can figure out the label correctly.

And we plot the confusion matrix for the logistic model. We can see that most of the classification are correct, and 93 percents of true ERROR data has been classified by the logistic model.

6.3 Random Forest Classifier

For the random forest classifier, we focus on building better classifiers and used grid search to find an optimal combination of estimators and max depths.

We found that when the max depths are 8 and estimators are 10 is a better classifier. After running the random forest classifier, the result is quite similar to the logistic model, it is a little worse on the accuracy as the accuracy on test set is 0.990. We may need more test to get a more optimal combination of estimators and max depths in the future research.

For the confusion matrix, it is quite similar to the logistic model and 93 percent of true ERROR data has been classified by this model.

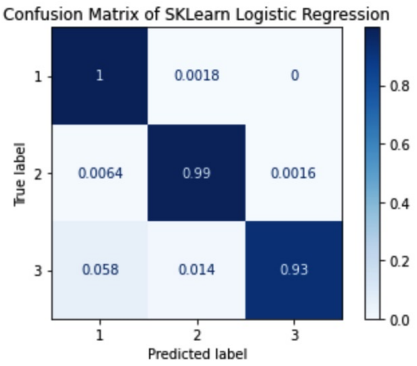

Fig. 4. Logistic Regression Matrix

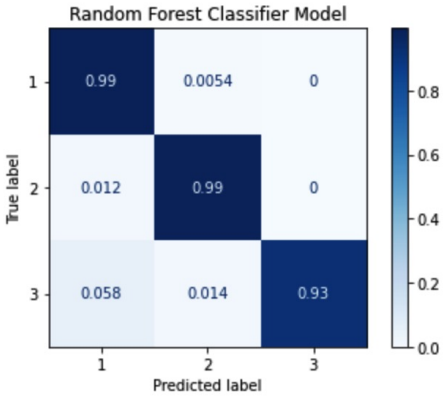

Fig. 5. Random Forest Classifier Matrix

6.4 Multilayer Perceptron Model

At last, we use Multilayer Perceptron, and we build a Feedforward neural network model. In this model, we use PyTorch to train the model. We convert the data to Tensors, create data loader, weight sampling, calculate the weight for each class and finally define the model. In the model, we use 5 layers and setting the learning rate to 0.0001. We train the model for 10 times and the training loss is around 0.016, and the training accuracy is 0.996. Below is the training Loss and training accuracy change for each epoch.

When we test the model using the test set, the testing loss is 0.0576 and the testing accuracy is 0.9963, which is higher than the logistic model and random forest classifier.

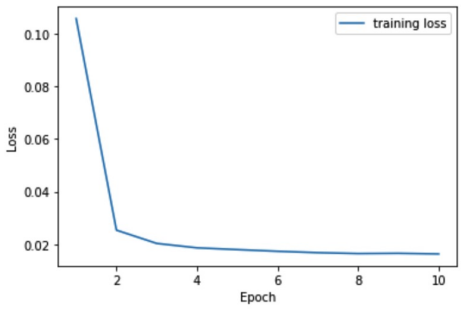

Fig. 6. Training Loss for FNN model

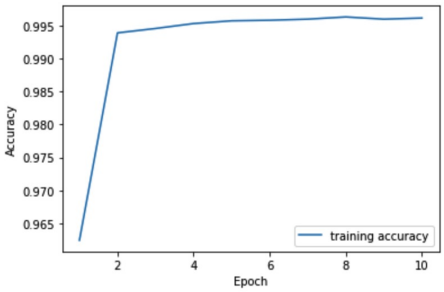

Fig. 7. Training Accuracy for FNN model

6.5 Results

We can see that all of the 3 methods have a high accuracy rate (larger than 99 percents), but the FNN model has a more accuracy rate than the other 2 models. In the future research, we may implement a model that can classify the ERROR message in a more detail way, as the ERROR message can split into different error types.

REFERENCES

[1] Changelog and Note Release. 2022. *Changelog and Note Release.* Retrieved April 29, 2022 from https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/release/

[2] W. Dickinson, David Leon, and A. Fodgurski. 2001. Finding Failures by Cluster Analysis of Execution Profiles. *The 23rd International Conference on Software Engineering* (May 2001), 10 pages. https://ieeexplore.ieee.org/author/37087809776

[3] Rui Ding, Qiang Fu, Jianguang Lou, Qingwei Lin, Dongmei Zhang, Jiajun Shen, and Tao Xie. 2012. Healing online service systems via mining historical issue repositories. *the 27th IEEE/ACM International Conference on Automated Software Engineering* (Sept. 2012), 318–321.

[4] Rui Ding, Qiang Fu, Jianguang Lou, Qingwei Lin, Dongmei Zhang, and Tao Xie. 2014. Mining Historical Issue Repositories to Heal Large-Scale Online Service Systems. *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks* (2014), 311–322.

[5] Qiang Fu, Jian Guang Lou, Yi Wang, and Jiang Li. 2009. Abstracting Execution Logs to Execution Events for Enterprise Application. *The 9th IEEE International Conference on Data Mining* (Dec. 2009), 149–158.

[6] Qiang Fu, Jieming Zhu, Wenlu Hu, Jianguang Lou, Rui Ding, Qingwei Lin, Dongmei Zhang, and Tao Xie. 2014. Where do developers log? an empirical study on logging practices in industry. *Proc. of the 36th International Conference on Software Engineering* (June 2014), 10 pages. https://doi.org/10.1145/2591062.2591175

[7] Zhen Ming Jiang, Ahmed E. Hassan, Parminder Flora, and Gilbert Hamann. 2008. Abstracting Execution Logs to Execution Events for Enterprise Application. *The 8th International Conference on Quality Software* (2008), 6 pages. https://ieeexplore.ieee.org/document/4601543/authors#authors

[8] Jianguang Lou, Qiang Fu, Shengqi Yang, Ye Xu, and Jiang Li. 2008. Mining invariants from console logs for system problem detection. *USENIX Annual Technical Conference* (2008). https://doi.org/10.5555/1855840.1855864

[9] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval.* Cambridge University Press.

[10] Alexander V. Mirgorodskiy, Naoya Maruyama, and Barton P. Miller. 2006. Problem Diagnosis in Large-Scale Computing Environments. *the ACM/IEEE SC 2006 Conference* (Nov. 2006).

[11] Weiyi Shang, Zhen Ming Jiang, Hadi Hemmati, Bram Adams, Ahmed E. Hassan, and Patrick Martin. 2013. Assisting developers of Big Data Analytics Applications when deploying on Hadoop clouds. *The 35th International Conference on Software Engineering* (May 2013), 402–411.

[12] Weiyi Shang, Meiyappan Nagappan, and Ahmed E. Hassan. 2015. Studying the relationship between logging characteristics and the code quality of platform software. *Empirical Software Engineering* (Feb. 2015), 1–27.

[13] Weiyi Shang, Meiyappan Nagappan, Ahmed E. Hassan, and Zhen Ming Jiangn. 2014. Understanding Log Lines Using Development Knowledge. *IEEE International Conference on Software Maintenance and Evolution* (Sept. 2014), 21–30.

[14] Weiyi Shang, Meiyappan Nagappan, Ahmed E. Hassan, and Zhen Ming Jiangn. 2016. Log Clustering based Problem Identification for Online Service Systems. *2016 IEEE/ACM 38th IEEE International Conference on Software Engineering Companion* (May 2016), 102–111.

[15] Wei Xu, Ling Huang, Armando Fox, David Patterson, and Michael I. Jordan. 2009. Detecting large-scale system problems by mining console logs. *The 22nd ACM Symposium on Operating Systems Principles* (Oct. 2009), 15 pages. https://people.eecs.berkeley.edu/~jordan/papers/xu-etal-icml10.pdf

[16] Ding Yuan, Soyeon Park, and Yuanyuan Zhou. 2012. Characterizing logging practices in open-source software. *The 2012 International Conference on Software Engineering* (June 2012), 102–112.

[17] Ding Yuan, Jing Zheng, Soyeon Park, Yuanyuan Zhou, and MStefan Savage. 2012. Improving software diagnosability via log enhancement. *ACM Transactions on Computer Systems* (Feb. 2012). https://doi.org/10.1145/1950365.1950369

[18] Jieming Zhu, Pinjia He, Qiang Fu, Hongyu Zhang, Michael R. Lyu, and Dongmei Zhang. 2015. Learning to Log: Helping Developers Make Informed Logging Decisions. *The 37th International Conference on Software Engineering* (May 2015).