Log Mining using Natural Language Processing for Anomaly Detection

Ankita Chikodi
MS Software Engineering
Department of Computer Science
San Jose State University
E-mail:ankita.chikodi@sju.edu

Arkil Thakkar
MS Software Engineering
Department of Computer Science
San Jose State University
E-mail: arkil.thakkar@sjsu.edu

Nehal Sharma
MS Software Engineering
Department of Computer Science
San Jose State University
E-mail:nehal.sharma@sju.edu

Shravani Pande
MS Software Engineering
Department of Computer Science
San Jose State University
E-mail:shravani.pande@sjsu.edu

***Abstract*- Logs store runtime information of systems, which is critical for developers and support engineers to monitor their systems to figure out anomalous behavior. Due to ever-expanding volume of logs, data mining models are used to extract system behavior information. However, most of the logs data is unstructured therefore it needs to be parsed. In our project, we have designed a pipeline to analyze anomalous behavior in logs, which after being traced, are notified through an e-mail to system administrators.**

Keywords: Anomaly detection, Log file, Natural Language Processing, Kafka, Spark streaming

I. INTRODUCTION

Logs contain crucial information about the state of a system such as the timestamp, level of the logs and the state of a task execution. Analyzing server logs provides deep intuitive understanding about its activity and aids in examining the correct state and diagnosing errors. However, interpreting logs is a daunting task. Firstly, application deployed on the production servers generate gigabytes of unstructured data which makes it unfeasible to manually analyze logs for key information. Secondly, log messages have unstructured format which further elevates the difficulty in analysis of log data.

In our project, we aim to reduce human intervention for log file analyzing and debugging. Our methodology addresses log mining as a NLP domain problem and makes use of sophisticated techniques from natural language processing to extract key features from the logs. We are applying machine learning Random Forest Classifier to build the model. The log data that is being generated is being processed in real-time through Kafka Spark streaming pipeline. If the anomaly is detected, the system administrators are notifying about the anomalous behavior traced in the log files. Our model demonstrates an accurate predictive performance (93% accuracy).

II. SYSTEM ARCHITECTURE

The log data is transferred through Kafka producer which is received as a Spark streaming object. The log data in the Spark streaming object is pre-processed and applied to the pre-trained model which is build using the older logs. After pre-processing, the model will predict whether it is an anomaly and then it will send a mail to the user.

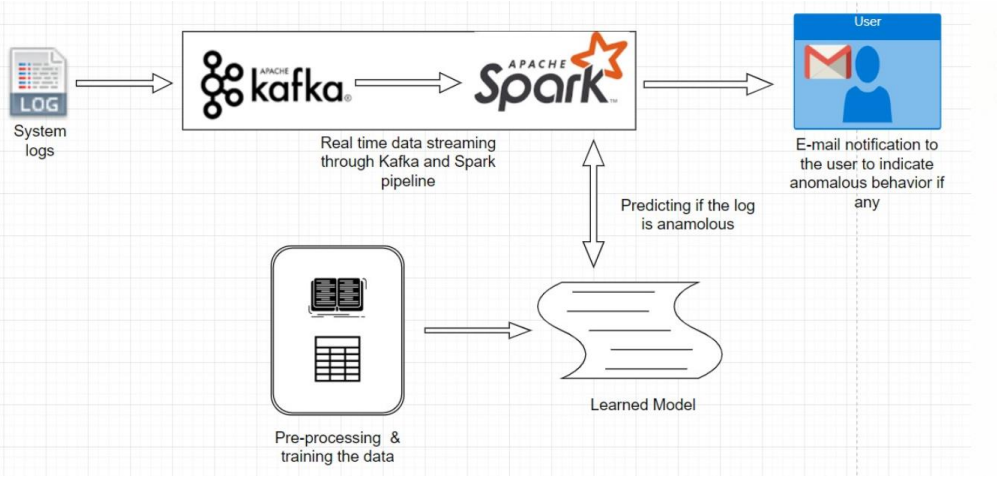

Fig 1: System Architecture

III. PROCESS FLOW

As our project is based on analyzing log data, it is important to use high-quality dataset for conducting experiments. However, procuring live application data can be difficult owing to privacy and security concerns. By conducting a literature review, our team was able to identify publicly available log datasets.

For our project, we have used HDFS labelled log data which has been procured from LogPAI by requesting them for the dataset for our project purpose. It consists of approximately 11,175,629 log messages with 5,75,062 labels.

A. LOG PRE-PROCESSING

A log record consists of a specific system event with the following fields: timestamp, verbosity level and message content. The objective of log parsing is to extract the pattern from the log contents. Each different type of event are associated with an event id which are generated using a hash function. A dictionary is generated for mapping each block id with the event sequences. Using the label file, event sequences are mapped to their corresponding labels using the block id. Categorical variable (anomaly, normal) is converted to binary values 1 and 0 respectively.

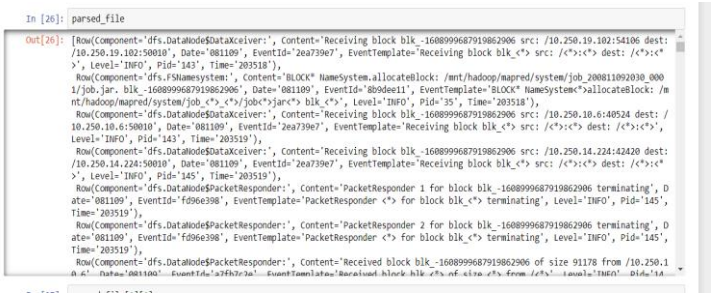

Fig 2: Raw Data

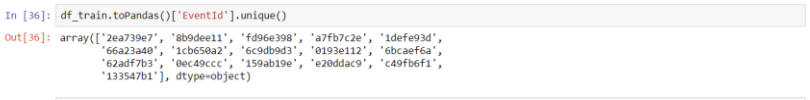

Fig 3: Event Sequence

```
In [63]: df_event.show(10)
```

```
+--------------------+--------------------+-----+
|             BlockId|       EventSequence|Label|
+--------------------+--------------------+-----+
|blk_-160899968791...|[2ea739e7, 8b9dee...|    0|
|blk_7503483334202...|[2ea739e7, 2ea739...|    0|
|blk_-354458337728...|[2ea739e7, 62adf7...|    1|
|blk_-907399258668...|[2ea739e7, 159ab1...|    0|
|blk_7854771516489...|[2ea739e7, 2ea739e7]|    0|
+--------------------+--------------------+-----+
```

Fig 4: Label Mapping

B. NATURAL LANGUAGE PROCESSING

The event sequence values are converted into TF-IDF matrix after which correlation among the different event sequences of the matrix is found. The matrix is normalized.

C. BUILDING A MODEL:

The dataset is split into train(80%) and test(20%).Random Forest Classifier is applied to the normalized matrix and anomaly prediction labels are generated. After which the model's accuracy is calculated and then the model is dumped into the pickle file.

```
====== Transformed test data summary ======
[[ 0.  1.  0.  0.  0. 10.  1.  0.  1.  0.  0.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0. 11.  0.  0.  0.  1.  0.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0. 11.  0.  1.  0.  0.  0.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0. 11.  0.  0.  0.  1.  0.  0.  0.  0.  0.]
 [ 0.  1.  0.  0.  0. 10.  1.  1.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  1.  0.  0. 10.  0.  0.  1.  0.  0.  0.  0.  1.  0.]
 [ 1.  0.  0.  0.  0. 11.  0.  0.  1.  0.  0.  0.  0.  0.  0.]
 [ 0.  1.  0.  0.  0. 10.  1.  0.  1.  0.  0.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0. 11.  0.  1.  0.  0.  0.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0. 11.  0.  1.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  1.  0.  0.  0. 10.  1.  1.  0.  0.  0.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0. 11.  0.  1.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  1.  0.  0.  0. 10.  1.  0.  1.  0.  0.  0.  0.  0.  0.]
 [ 0.  1.  0.  0.  0. 10.  1.  1.  0.  0.  0.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0. 11.  0.  1.  0.  0.  0.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0. 11.  0.  0.  0.  1.  0.  0.  0.  0.  0.]]
Test data shape: 16-by-15
```

Fig 5: TF-IDF

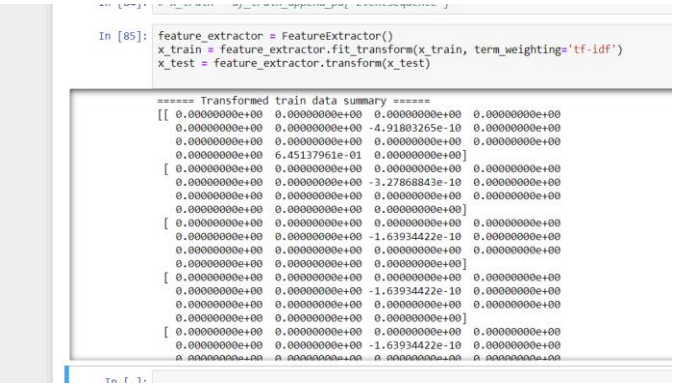

Fig 6:Normalized Vector

D. PREDICTIONS ON NEW DATA:

The new data is sent through the Kafka producer in chunks which is received by Spark streaming object in real time. On this object, the pre-processing is performed and given to the pre-trained model which computes the label of the log. If anomaly is found, then the whole data associated with it is sent to the system administrator through e-mail.

IV. CONCLUSION AND FUTURE ENHANCEMENTS

The accuracy of our model is 93% and it successfully reduces human intervention in anomaly detection in log data. For future enhancement, we plan to classify logs in the extremely critical conditions and normal anomalies and sent the e-mail notifications accordingly.

V. ACKNOWLEDGEMENTS

We would like to extend our gratitude to Prof Rakesh Ranjan for his continued support and providing us the guidance necessary to work on this project. We would like to thank our Teaching Assistant Ms. Kavya Mohan Sahai for timely communication about the project requirements.

VI. PROJECT REPOSITORY

https://github.com/SJSU272Spring2019/Project-Group-2

VII. REFERENCES

[1] H. Mi, H. Wang, Y. Zhou, R. Lyu, and H. Cai, "Toward fine-grained, unsupervised, scalable performance diagnosis for production cloud computing systems," IEEE Transactions on Parallel and Distributed Systems, vol. 24, pp. 1245–1255, 2013.

[2] W. Xu, L. Huang, A. Fox, D. Patterson, and M. Jordon, "Detecting largescale system problems by mining console logs," in SOSP'09: Proc. of the ACM Symposium on Operating Systems Principles, 2009.

[3] Q. Fu, J. Lou, Y. Wang, and J. Li, "Execution anomaly detection in distributed systems through unstructured log analysis," in ICDM'09: Proc. of International Conference on Data Mining, 2009.

[4] I. Beschastnikh, Y. Brun, S. Schneider, M. Sloan, and M. Ernst, "Leveraging existing instrumentation to automatically infer invariantconstrained models," in ESEC/FSE'11: Proc. of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering, 2011.

[5] W. Shang, Z. Jiang, H. Hemmati, B. Adams, A. Hassan, and P. Martin, "Assisting developers of big data analytics applications when deploying on hadoop clouds," in ICSE'13: Proc. of the 35th International Conference on Software Engineering, 2013, pp. 402–411.

[6] D. Yuan, S. Park, P. Huang, Y. Liu, M. Lee, X. Tang, Y. Zhou, and S. Savage, "Be conservative: enhancing failure diagnosis with proactive logging," in OSDI'12: Proc. of the 10th USENIX Conference on Operating Systems Design and Implementation, 2012, pp. 293–306.

[7] K. Nagaraj, C. Killian, and J. Neville, "structured comparative analysis of systems logs to diagnose performance problems," in NSDI'12: Proc. of the 9th USENIX conference on Networked Systems Design and Implementation, 2012.

[8] A. Oprea, Z. Li, T. Yen, S. Chin, and S. Alrwais, "Dectection of earlystage enterprise infection by mining large-scale log data," in DSN'15, 2015.