CONCEPT OF ANOMALY DETECTION

Sowmya Chenthi Kumar
University of British Columbia, Okanagan
British Columbia, Canada

ABSTRACT

This project deals with working on the log files. In computing, a log file is a file that records events that occur in an operating system or other software runs, messages between different users of a communication software. Essentially it is used to get information about the usage patterns, activities, and other operations. Hence these files can be used to understand system runtime problems. These problems can be identified through the concept of anomaly detection. Anomaly detection/outlier detection is the identification of rare items, events or observations which raise suspicions by differing significantly as compared to the rest of the data. It plays a major role in management of modern large-scale distributed systems. Traditionally, operators have to go through these log files manually to identify the problem. But however, in the recent years, the scale and the complexity of the systems have increased to mammoth amounts making it infeasible for manual inspection. Hence to reduce the manual effort, we use anomaly detection to automate log analysis.

In this paper we define a general methodology to mine the rich source of information to automatically detect runtime errors. The raw log messages will be in an unstructured textual format. Hence the first primary step involves parsing these log file to convert them into a much more structured format which comprises of the sequence of structured events. The analysis is then performed ion this sequence of structured events. We then use several supervised learning techniques to perform anomaly detection. The model with the most appropriate outcome is chosen to validate our analysis with another dataset from Hadoop File Systems.

INTRODUCTION

The console logs provide detailed information about the runtime information and system states, each contains a timestamp and a log message highlighting what went wring or what has essentially happened. The collection of logs is the first crucial process which we must take.[1] For a long time, these logs were manually taken care by developers when problem arises. However, in the recent times modern large-scale services usually comprises of several components authored by hundreds of developers. Hence it becomes almost impossible for an individual to go through the logs and come to useful conclusions. The modern logs are too large to examine manually and too unstructured to analyse automatically.[1]The work around which people have temporarily used is create Adhoc scripts to search for the required keywords but they are not that efficient. Unusual log messages indicate a problem hence naturally it becomes useful to use anomaly detection in machine learning to come to an informed decision.

As suggested earlier the raw logs will be unstructured texts. Though they are unstructured they will still have a specific pattern in their message which we can refer to as the template. These structured templates generated will play a key role in detecting anomalies.

This process involves four major steps which is shown below[4]:

- Log collection
- Log parsing
- Feature extraction
- Anomaly detection

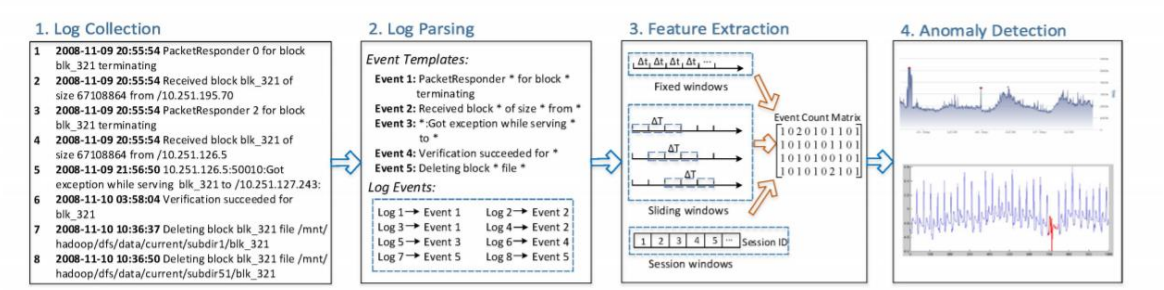


Figure 1: Framework of anomaly detection

Log collection

Understanding our Dataset:

In this paper we are working with logs from the HDFS – Hadoop file distributed system. We have two datasets:[1] HDFS_1 that is used for training the model and the HDFS_2 for validating the model. HDFS_1 is a labelled dataset, where the logs are generated from a private cloud environment and manually labelled through a set of handcrafted rules to identify the anomalies. The HDFS_2k log file is a sample which will be again used to validate our dataset.

Log Parsing

Once we have collected the datasets, we are going to perform a parsing operation as a preliminary step for our analysis.

There is mammoth information buried in millions of lines of free text console logs. A machine learning algorithm will require the data in terms of numerical values. Hence, we should extract features from the text log files which in turn can be converted to numerical which in turn can be useful for our machine learning algorithms. Though these text files look very long and highly disorganized, there is a specific underlying pattern which is present in them. They have a specific underlying structure in them as they are generated from a bunch of log printing statements in the system. Hence, we should first try to extract the hidden

schema which is present in the log files. This is precisely the concept which we are trying to execute using the parsing technique.

The logs consist of two different parts – the constant parts and the variable parts which will vary for different log entries. For example, if we have two log entries like:

- "Connection from 10.10.34.11 closed"
- "Connection from 10.10.35.12 closed"

Both these logs have the same inherent structure present among them. The words "Connection", "from" and "closed" are the constant parts in this log as they always stay the same for that specific set of operations. Whereas the other parts like the port numbers here are considered as the random parts/ variable parts as they are not constantly changing depending on ports and other conditions accordingly. The constant parts are usually predefined by the developers and the random parts are dynamically generated. The dynamically generated random parts do not play significant role in the anomaly detection.[2]

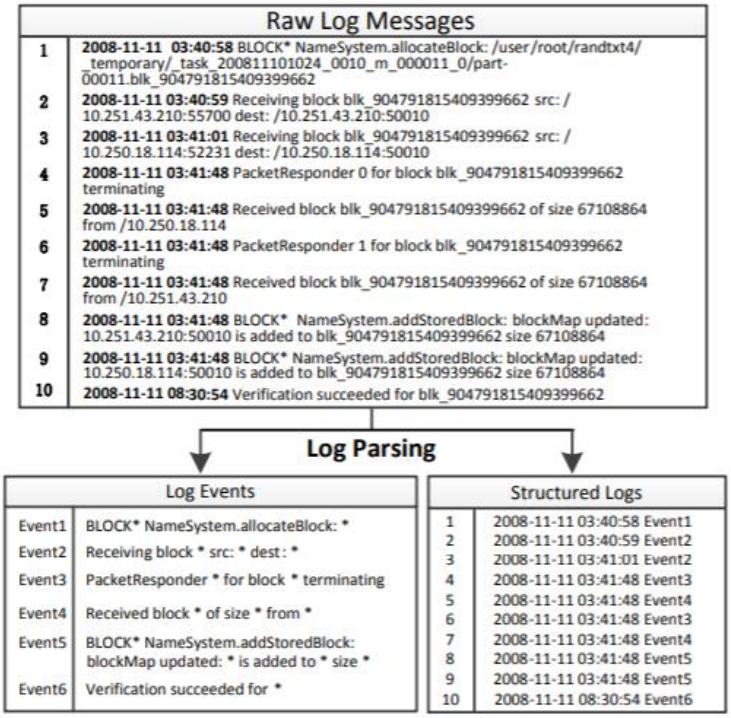


Fig. 1: Overview of Log Parsing

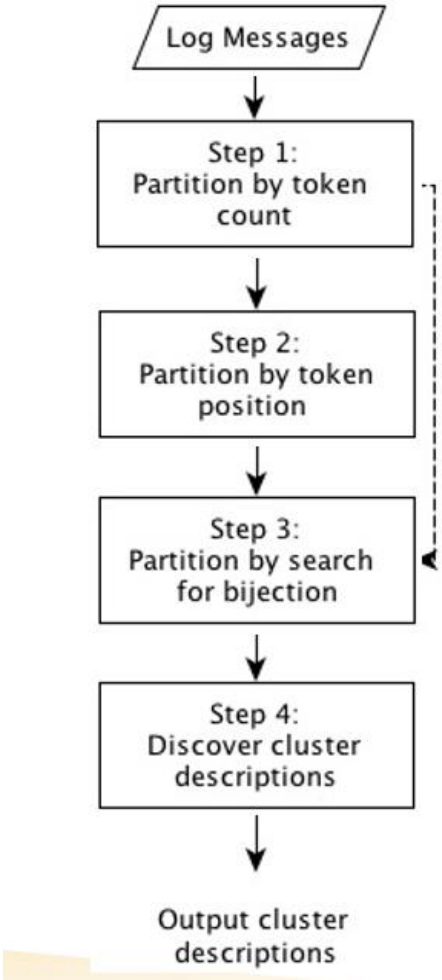
Hence log parsing essentially differentiates the constant part (schema) from the variable / random part which are deemed to be not useful for our analysis. This process is carried out by data clustering which is a technique where entities are sorted into groups called clusters, where the members of each cluster are like each other and dissimilar from the members of the other groups. Clustering for analysis and classification of large datasets, which is a little overwhelming to be done manually.

Log parsing using the Iterative Partitioning Log Mining (IPLoM):[3]

There are two different ways in which the log parsing can be done, through a clustering approach (SLCT- Simple Log File Clustering Tool) and other heuristic based approach (IPLoM- Iterative Partitioning Log Mining). In this study we have used the IPLoM log parser method owing to its high accuracy and light weight algorithm.

The IPLOM method leverages the unique characteristics of log messages and uses iterative log partitioning which in turn facilitates efficient message type extraction. The IPLOM performs parsing through three steps after which the templates are generated. [3]

1) Partitioning based on the event size: First the logs are partitioned into different clusters based on their respective lengths.
2) Based on token position: Among the partitions the words at different positions are counted. The position with the least number of unique words are the zones for the next partition. (The log messages are split at that location).
3) By search for mapping: The next partition is done by searching for relationships between unique tokens into two positions based on a heuristic criterion.
4) Generate the log templates: The log templates are generated as a final step for each of these clusters.



All the clustered message templates can be clustered into a list of log events and a structure log can be generated by matching each log with a corresponding event. Hence the first and foremost work will be to extract information conveyed in logs to this level of granularity for our problem.

One of the major information to remember here is that the messages are highly correlated with one another. When grouped in clusters, there is a strong correlation present between them. A group of messages is generally a good predictor than a single message. Incomplete messages are the most important sources of anomalies. With the help of the correlated messages we can compare them amongst their respective clusters or groups and if we find some bits and pieces as missing this in turn can point out that this is an anomaly.

FEATURE EXTRACTION

This step is used to extract valuable features from log events to feed into the anomaly detection models. This done by extracting feature vectors from the grouped messages. Hence the input to this step is the log events that are generated after using the parsing algorithm and the output is an event count matrix. We should first separate the data into groups with each group being a representation of its log sequence. After the log sequence is constructed, a windowing operation is done to divide the data.

After which an event count matrix is generated. In the event count matrix, each row represents a block and each event indicates an event type. Then the matrix counts how many times an event is repeated for respective blocks.

TF-IDF Vectorizer:

Once the count matrix is generated, instead of directly using them for our analysis we have used the TF-IDF normalizer (which is similar to the count vectorizer) to process it. The goal of using tf-idf instead of the raw frequencies of occurrence of a token in a given document is to scale down the impact of tokens that occur very frequently in a given corpus and that are hence empirically less informative than features that occur in a small fraction of the training corpus. Hence for our analysis the TF-IDF code assigns less weights to the event types which are most commonly occurring and are likely to contribute less for our process of anomaly detection. The main difference between the TF-IDF vectorizer and the count vectorizer is that the count vectorizer gives number of frequencies with respect to index of vocabulary whereas tfidf vectorizer considers overall documents of weight of words. TF-IDF is especially useful in dealing with the most frequent set of words. By using this vectorizer we are penalizing them.

Count Vectorizer

	blue	bright	sky	sun
Doc1	1	0	1	0
Doc2	0	1	0	1

TD-IDF Vectorizer

	blue	bright	sky	sun
Doc1	0.707107	0.000000	0.707107	0.000000
Doc2	0.000000	0.707107	0.000000	0.707107

spicy sparse matrix of count and tf-idf vectorizer

This process is carried out with the help of the *loglizer package* which is a part of the logpai package. The output of this is a matrix which will further be split into a x_train, y_train, x_test and y_test dataset. This split is generated depending on the ratio of the split specified by us.

Once the test and train datasets are generated, we can carry out multiple analysis on them. In this project I have carried out three different types of analysis and supervised algorithm models on them.

ANOMALY DETECTION

In this project we have decided to use three different types of analysis on the final processed count matrix which was created in the previous step.

DECISION TREE CLASSIFIER:

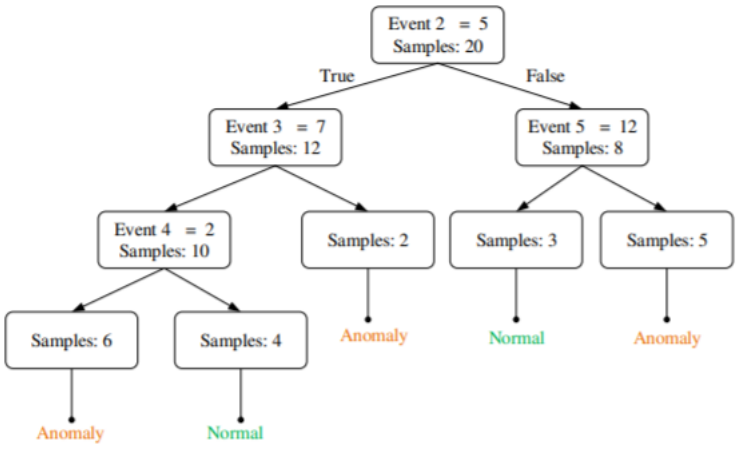


Figure 2: An example of decision tree

Decision Tree[4] models help highlight how to use machine learning effectively to enhance our decision-making abilities. One of the major reasons for choosing decision trees in machine learning models is that we can use logics and maths to generate rules rather than just selecting them based on intuition and subjectivity. The most basic underlying idea is choosing yes or no questions to classify our data. In our case these questions will be anomaly or normal.

The decision tree makes explicit all possible alternatives and traces each alternative to its conclusion in a single view, to make easy comparison among the various alternatives. The major advantage of decision trees is its transparency. It also has the ability of selecting the moist biased and comprehensibility features. It is popularly known for its ease of interpretability.

Here we have used the basic decision tree algorithms for predicting is a specified log entry is an anomaly or not. The algorithm uses the concept of Gini index impurity, the split is made in such a way that the impurity is reduced to the maximum possible. In other words, each split is made considering the best-case scenario at that specific point. [5]

In this project after splitting the dataset into test and train sets the training set of the data is used to train the decision tree model (this involves both the vectorized matrix and the response variables- which in this case is the label anomaly or normal). According to the values of the splitting attributes, the training data are partitioned into several subsets. Until all the instances in a subset belong to the same class in any decision tree the algorithm proceeds recursively.

After training the model the test dataset is fed into the model and the new values are predicted based on the rules generated while training a model. Then the predicted values are compared against the true values (as it is supervised modelling and we have the response variable). If the values are close to one another, the set can be said to be precise. Here we can measure two different things: Accuracy and Precision. Here we measured the accuracy for easy comparison with other models.[4][10]

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

$$Precision = \frac{TP}{(TP + FP)}$$

TP = True positive, TN = True Negative
FP = False Positive, FN = False Negative

We can also get a better understanding about the number of properly classified and misclassified through the confusion matrix[7]. A confusion matrix is a table that is often used

to describe the performance of a classification model on a set of test data for which the true values are known. When we implemented the decision tree algorithm on the train data and then tested the results with the testing data, we got very high accuracy.

NAÏVE BAYE'S ALGORITHMN:

Naïve bayes classifier is widely used for text classification in machine learning based on conditional property of features belonging to a class, where the features are selected by feature selection methods. It is a supervised learning algorithm based on Bayes theorem and primarily is used for solving classification problems. It is mainly used for high dimensional dataset. Hence our approach is of using Naïve bays classifier for solving this problem is justified.

PCA: PRINCIPLE COMPOINENT ANALYSIS:

Principle Components Analysis (PCA) is an unsupervised method primary used for dimensionality reduction within machine learning. It is calculated based on calculating the covariance matrix of the data and performing eigen value decomposition on the covariance matrix. The results of the PCA will have a low dimensional structure of data which is rotated across the axis and the leading (uncorrelated) latent factors determining the variations in the data[1][2]. Overall, this is a statistical method that is used for dimensionality reduction. The basic idea behind PCA is to project high-dimension data to a new coordinate system composed of *k* principal components/ dimensions, where *k* is set to be less than the original dimension. PCA calculates the k principal components by finding components (i.e., axes) that catch the most variance among the high-dimension data. Hence, we get a low dimensional data with the maximum information preserved.[1]

We have taken this concept of PCA and derived the final rotated values of the original predictors, over which we then perform logistic regression.

Logistic Regression after PCA:

After transforming the required predictors into smaller dimensions, we are performing logistic regression over the transformed values. Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist[10]. In regression analysis, logistic regression is estimating the parameters of a logistic model. It is mainly used to obtain odds ratio in presence of more than one explanatory variable. It requires average or no multicollinearity between independent variables which in our case is achieved after the PCA. After performing the logistic regression, we get a good accuracy rate with the transformed data.

MULTI LAYER PERCEPTRON:

A multilayer perceptron (MLP) is a class of feedforward artificial neural network (ANN). An MLP 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.

We have used this simple technique here to understand the effects of neural networks on our analysis. They provide a very high accuracy as compared to other networks. In fact, they have the least misclassification rate and have not been that biased towards the larger dataset.

CONCLUSIONS AND FUTURE WORK

We proposed a general approach to problem detection via analysis of console logs. We can understand the structure of console logs, because of which we are able to parse logs accurately. Because of this accurate parsing we were able to o extract the identifiers and state variables, which are widely found in logs. After which we used powerful feature extraction techniques to give us the required count vector matrix. Over which we have done the required modelling. Here we have used two different supervised techniques: The decision tree and the Naïve Bayes classifier. We have used one unsupervised algorithm – Principle Component Analysis and one simple neural network. When comparing between all these values we can see that the values are predicted with very high accuracy. This answer can be considered both good as well as a little fishy. The major issue with our dataset is that there is a heavy imbalance between the normal and anomalies. This imbalance usually pushes the model to predict to the majority class. To deal with this issue, we should not just look at the accuracy of the model but also look at the other relevant evaluation metrics. Few of the metrics which we can consider are [10]:

- Precision/Specificity: how many selected instances are relevant.
- Recall/Sensitivity: how many relevant instances are selected.
- F1 score: harmonic mean of precision and recall.
- MCC: correlation coefficient between the observed and predicted binary classifications.

With our dataset we have used these techniques and found that the Multi Layer Perceptron model is by far the best model. However, though we have come to a conclusion about the model, we can make some more improvements from the understanding that we got from our analysis so far.
One such improvement will be using semi supervised techniques[11] – combination of the supervised and unsupervised techniques, as this helps doing an unbiased classification which also leverages the presence of the labelled values. Though the number of these labelled anomaly values are less, we can use them as the starting points for getting the final answers in semi-supervised algorithms.

REFERENCES

[1] Wei Xu, Ling Huang, Armando Fox, David Patterson, Michael I. Jordan, *Detecting Large-Scale System Problems by Mining Console Logs,* 2009

[2 Pinjia He, Jieming Zhu, Shilin He, Jian Li, Michael R. Lyu. *An Evaluation Study on Log Parsing and Its Use in Log Mining. IEEE/IFIP International Conference on Dependable Systems and Networks* (DSN), 2016

[3] Adetokunbo Makanju, A. Nur Zincir-Heywood, Evangelos E. Milios, *Clustering Event Logs Using Iterative Partitioning*, 2009

[4] Shilin He, Jieming Zhu, Pinjia He, and Michael R. Lyu*, Experience Report: System Log Analysis for Anomaly Detection*, 2016

[5] Harsh Patel, Purvi Prajapati, *Study and Analysis of Decision Tree Based Classification Algorithms*, 2018

[6] Jieming Zhu, Shilin He, Jinyang Liu, Pinjia He, Qi Xie, Zibin Zheng, Michael R. Lyu. *Tools and Benchmarks for Automated Log Parsing. International Conference on Software Engineering (ICSE)*, 2019.

[7] Adetokunbo Makanju; A. Nur Zincir-Heywood; Evangelos E. Milios, *A Lightweight Algorithm for Message Type Extraction in System Application Logs, IEEE Transactions on Knowledge and Data Engineering* (Volume: 24, Issue: 11, Nov. 2012)

[8] Pinjia He, Jieming Zhu, Shilin He, Jian Li, Michael R. Lyu, *Towards Automated Log Parsing for Large-Scale Log Data Analysis*, 2017

[9] Pinjia He, Jieming Zhu, Zibin Zheng, Michael R. Lyu, Drain: *An Online Log Parsing Approach with Fixed Depth Tree*, 2017 IEEE 24th International Conference on Web Services

[10] Peter Bodík, Moises Goldszmidt, *Fingerprinting the Datacenter: Automated Classification of Performance Crises*

[11] Yassine Ouali, Céline Hudelot, Myriam Tami, An Overview of Deep Semi-Supervised Learning, 2006