<Temp: Log data Analysis>

Summer Project report

Ravi Charan, ravicharan.vsp@gmail.com
Supervisors: Dr Domenico Giordano & ¡filler¿

Abstract: Massive amount of data is generated by the Openstack cloud services in the format of service logs. Besides timestamps and log level fields, these logs contain additional information useful for pattern analysis. Unfortunately this information is generally exposed in semi-structured text format, not allowing direct analysis without additional munging of the data. Traditional approaches to extract information from those fields are rule-based, mainly applying regular expressions upon knowledge of the text structure. These approaches require a pre-knowledge of all text patterns and are not scalable with the services growth. We propose to build a data analytic approach for log data, leveraging the technologies adopted for Web [1] and genomics [2] data mining (such as minhashing [3] and locality-sensitive hashing [4])LaTeX.

1 Introduction

Analysing logs to identify patterns and problems has been attempted ever since logs have been generated. At its most general, to log is "to put information into a written record" and is a process as old as writing systems themselves. Previously, manual investigation was used whereby administrators would read the logs stored on each computer, perhaps aided by grepping for keywords. As the scale of services has increased in the age of Big Data, this approach is no longer feasible. Attempts have been made in the last 20 years to automate this process. One elementary method that is investigated regularly and is a classic example of expert systems that emulate the human expert decision making process is the use of regular expressions (regex), nested if-then rules used to select key features. However, this requires prior knowledge of the dataset, is proportional in difficulty to the number of distinct message types present and require continuous updating if the nature of log files changes. Alternatievly, supervised and unsupervised machine learning algorithms have been investigated. In supervised learning training datasets where anomalous and normal data are labelled by hand are used to train an algorithm. Decision trees[5][3] have been used in this regime for classification of logs files when the error categories are already known. Since logs are generally labelled with a time stamp, a different school if techniques have utilised techniques adapted to time series data. . A Finite State Automata was trained assuming that each log key can be mapped to a state transition. In this way, the chronology of interrelated messages is captured, and anomalies in transition timings and loop executions are used to identify system errors. This method is relatively successful but requires that edit distances between all logs is calculated which is infeasible for large datasets.

We majorly deal with the log data which contains the "ERROR" messages and these messages are analyzed and then clustered into classes and example of the log error messages produced could be seen below. ¡**ADD RAW LOG DATA FIGURE**¿

2 Methodology

As we've seen in the previous section the log data comprises of error messages of multiple unknown classes. The basic idea is to analyse the data present and cluster the messages in to different classes. One can see from the error logs that a few features of all the messages are unique (eg. timestamps, ErrorID.. etc). These unique features are given lesser priority in the clustering of the data hence corresponding weights are assigned to these features. The distance between the messages is compared by ignoring these unique features. The trivial logic is to put the messages whose distance is less than a predefined threshold are put into the same class. A dummy illustration of the algorithm is shown below.

The figure shows the demonstration of the clustering and the classification procedure. Each one of the 3 shapes represents as class and the variables inside each shape represent the unique features in the dataset. The data is clustered based on how similar the shapes are ignoring the unique features of the message. Once the messages are clustered, the principal features are extracted from each class

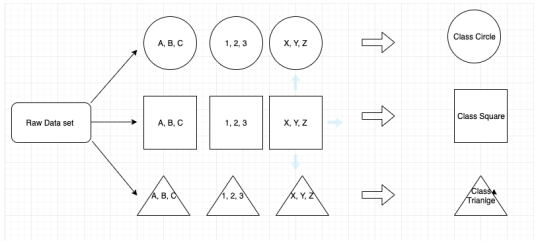

Figure 2.1: Illustration of the log data analysis procedure

and marked as a feature vector of that class. Now, every new entry in the data point is compared with these principal feature vectors to

The first step idea is to extract message "keys" which correspond to messages generated by the same lines of code but with different message variables, implemented using regex. They proceed to utilise the similarity in clustering raw log keys by using the edit distance as their metric, which is defined as the number of insertion, deletion or replacement operations required to convert one message into another. This distance does not account for the position of the words in the logs, and so they use the weighted edit distance which uses sigmoid function to place greater emphasis on words appearing earlier in the message than later based on the idea/observation that programmers tend to place important information in this way. The edit distances to all other logs is calculated, and all members whose distance is less than a threshold is connected via a link.

In this section, we define how we approach the problem mathematically, starting with our choice of similarity measure, the Jaccard Similarity, J: in order to identify similar items, we must first identify how to define and quantify similarity. We progress to describe shingling, a nave method to fully iterate over the parameter space in order to compute J, and learn how it is infeasible to apply to large datasets, such as when hundreds of thousands of logs are generated every minute, as can be the case in distributed systems.

2.1 Min hash

In order to identify similar items, a similarity measure is necessary. A common and intuitive means is that of the Jaccard Similarity. This treats two strings as sets of words, and defines the size of their intersection relative to their union as the defining metric of similarity. We therefore see that not only can the relation between two logs be evaluated,but the distances between different logs represent hierarchies of similarity, and these can be mined to reveal underlying clusters. Other alternative methods include measuring distance as the number of edits required to transform one message in another. he signatures we desire to construct for sets are composed of the results of a large number of calculations, say several hundred, each of which is a minhash. To minhash a set represented by a column of the characteristic matrix, pick a permutation of the rows. The minhash value of any column is the number of the first row, in the permuted order, in which the column has a 1. The probability that the minhash function for a random permutation of rows produces the same value for two sets equals the Jaccard similarity of those sets. [**Add reference for minhash here**]

2.2 Min Hash Locality Sensitive Hashing

MinHash is an effective means for reducing the dimensionality of variable length alphanumerical data to fixed length numerical output, with which Jaccard similarity can be probabilistically calculated via random sampling. However, each log hash must still be compared to every other in order to identify relationships, resulting in complexity O(n**2). There are a range of search methods for identifying nearest neighbours (examples, references) based on space partitioning but it was shown that they all degrade to linear search for sufficiently high dimensions. Locality Sensitive Hashing is used to reduce the search space by using hash collisions as a proxy for similarity. [**Add reference for LSH here**]

2.3 Shingling

3 Comparative Study

We implemented multiple approaches to implement the clustering of the log data and are tested for better performances in terms of accuracy and efficiency. The methods implemented include K means clustering, Minhash LSH (tested with shingles of variable sizes), and a Graph based approach that treats all the messages as nodes and the weight of the edges as the distance between the messages.

3.1 K means clustering

3.2 Minhash

3.3 Wieghted Minhash

3.4 Performance with the use of shingles

3.5 Graph Based clustering methods

4 Classification of Log Data

TO BE FILLED

5 Anomaly Detection

To be filled

6 Conclusions

To be filled

References

A Appendix

In the Appendix (or Appendices) you may give the details that did not fit in the main text. If necessary, you may use a one-column lay-out here. Start the first appendix on a new page.

B Appendices

If you have more than one Appendix, use letters to "number" them. You may start every appendix on a new page, but this is not necessary. If you have many appendices, it may be helpful for the reader to have a list of appendices on the first page of the appendices.