Anomaly detection and prediction in operational log data*

Mohammed Ibrahim[WAFH4I], Ammar Reslan[YNBLGQ], Vitalii Naumov[FY46IN], Sakina Hajiyeva[DKXCU0]

Eötvös Loránd University, H-1053 Budapest, Egyetem tér 1-3
mhdibrahim@student.elte.hu
https://www.elte.hu

Abstract. In software companies, there is an absolute requirement to ensure that systems once developed or servers once worked, functions at its best throughout its lifetime. operational log data is critical to maintaining performance and thus techniques to parse, understand and detect anomalies in operational log data are critical to ensuring efficiency in software company.
In this project, we explore anomaly detection, we analyze the most promising anomaly detection techniques and use LSTM Neural Network.

Keywords: Anomaly Detection · LSTM · RNN · Time Series · SARIMA.

1 Introduction

Applications generate massive amounts of log data, which automatically produced timestamped data that represents every single system and user event for all users of the application. This data is generated throughout the lifetime of an application and tends to be time-series, incredibly unstructured, textual, poorly formatted and is generated at an incredible rate as the application scales and adds more users.
Log data also contains anomalies which represent potential system faults and are thus critical to debugging application performance and errors. The details and timestamps of the anomaly offer a starting point for discovering when, how and where errors in the application occurred.

2 ANOMALIES IN LOG DATA

Anomalies in application log data are considered to be patterns or characteristics which do not follow the average or normal behavior during perfect operation.
Anomaly detection for application log data is particularly challenging whether automated or done manually for the following reasons:

- Unstructured Plain Text
- Redundant Runtime Information
- Large Unbalanced Data

* Supervised by Dr. Imre Lendak

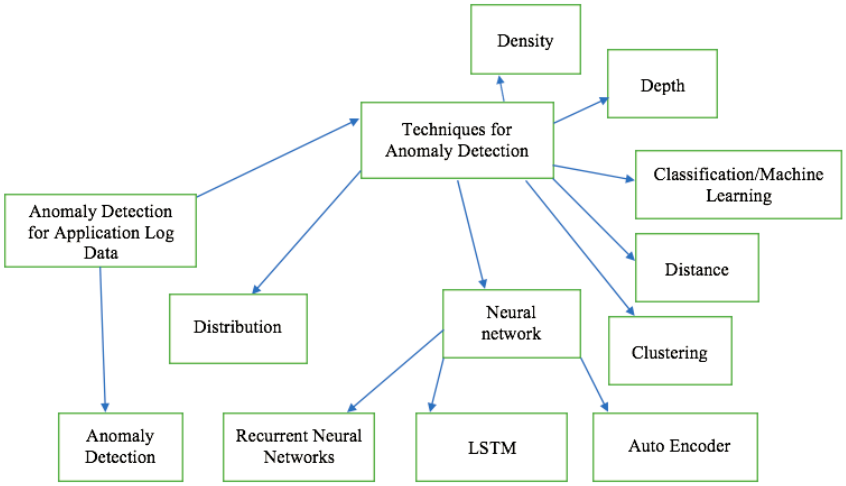

Fig. 1. Conceptual map of literature review.

3 TYPES OF ANOMALIES

3.1 Point Anomaly

A point anomaly is data which deviates significantly from the average or normal distribution of the rest of the data. Point anomalies are the simplest to detect and multiple techniques exist to automate point anomaly detection.

3.2 Contextual Anomaly

A contextual anomaly is identified as anomalous behavior restricted to a specific context, and normal according to other contexts. This type of anomaly also referred to as conditional anomaly, is often difficult to detect as it requires deep domain knowledge to understand the context within which the anomaly arises.

3.3 Collective Anomaly

Unlike contextual and point anomalies, collective anomalies appear as a group of anomalous values in data. Collective anomalies are anomalous behavior of a collection of data instances with respect to the complete dataset. Individual data instances might not represent an anomaly.

4 Neural Network Based Models For Anomaly Detection

RNNs are extensively used for Anomaly detection with approaches varying from clustering, classification to reconstruction. Recent Neural Network based anomaly detection research has revolved around Long Short-Term Memory Neural Networks and Auto-Encoders, both of which often leverage recurrent Neural Networks.

4.1 Anomaly Detection with LSTM Neural Network

Long Short-Term Memory (LSTM) Neural networks are a subcategory of Recurrent Neural Networks especially suited towards learning long-term dependencies between the input data. LSTM architecture is represented by memory blocks which themselves are essentially, recurrently connected structures

4.2 Anomaly Detection with Auto Encoder

Auto-Encoders are artificial neural networks designed to induce a representation for datasets by learning approximations of the dataset's identity function. They are generally paired with a Decoder which is used to recreate the initial dataset using the representation defined by autoencoders.

4.3 Anomaly Detection with using Statistics

In this part we create anomaly detection algorithms with the usage of moving averages, weighted average, exponential smoothing, double exponential smoothing. With these approaches it is easy to see the anomalies on the plots.

5 Dataset

The operational log we have is structured one, it contains 31808404 rows and 6 columns, we will take sample of it to do our training and predicting using LSTM model.

6 System Architecture

6.1 Kafka Connect

Kafka Connect is a tool for scalably and reliably streaming data between Apache Kafka and other data systems. we use it to import data from the dataset (csv file) via kafka producer and keep it in a topic to do some processing.
We convert the epoch in the dataset to datetime during the stream process via kafka consumer.

```
# of rows and columns = (31808404, 6)
```

	response_time	event_start_ms	event_start_epoc	year	month	day
0	8.674542	564	1544956946	2018	12	16
1	8.994416	388	1544956946	2018	12	16
2	10.833192	493	1544956945	2018	12	16
3	18.026684	315	1544956945	2018	12	16
4	10.089430	32	1544956945	2018	12	16

Fig. 2. Data exploration

6.2 Cassandra

The Apache Cassandra database is the right choice when you need scalability and high availability without compromising performance.
We use it to store the data after process it in kafka stream, we also use kafka connect cassandra tool to take the data from the topic and store it in tables in cassandra keyspace.

6.3 Kafka-python

It is a Python client for the Apache Kafka distributed stream processing system. Kafka-python is designed to function much like the official java client, with a sprinkling of pythonic interfaces (e.g., consumer iterators).
Using this python library, we have created Kafka Producer and Kafka Consumer. Taking into account that Kafka is running with python script, we were able to launch our LSTM model even before the data was sent to MongoDB.

6.4 MongoDB

MongoDB is a document database, which means it stores data in JSON-like documents. We believe this is the most natural way to think about data, and is much more expressive and powerful than the traditional row/column model, which is why we decided to use this database as well.
In the database at the moment when we were about to submit the project, we had two collections of data. First one is for normal logs for analysis nad the other one is for anomalies we detected during Kafka streaming process.
Moreover we used the database to store the csv file we had as the dataset to simulate a data-stream from it.

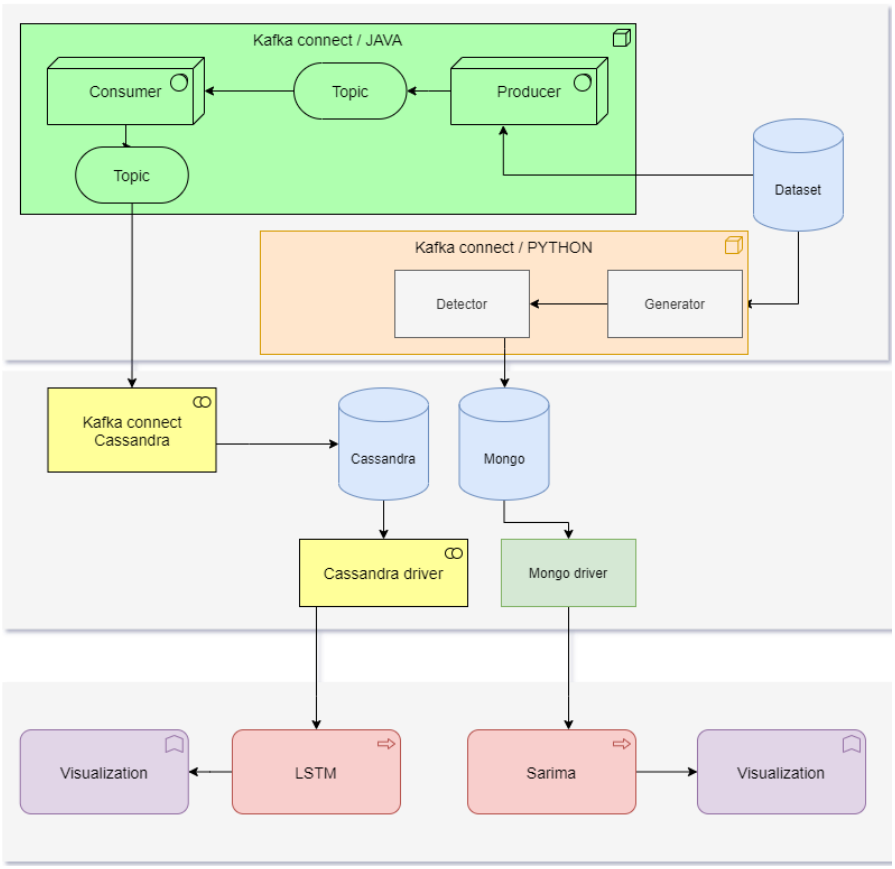

Fig. 3. System architecture

6.5 Docker

Docker is a tool for MacOS and Windows machines for the building and sharing of containerized applications and microservices. In this project we have used it as an open-source software to share the containers for the development.
Thanks to containerized Kafka and MongoDB, we were able to start using these services on any machine that has a Docker.

6.6 LSTM

Long Short-Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies, they were introduced by Hochreiter Schmidhuber (1997). LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, and it's overcome the vanishing gradients problem by replacing an ordinary neuron with a complex architecture called the LSTM unit or block, the main components of the LSTM architecture are shown in the Fig. 4 [6]

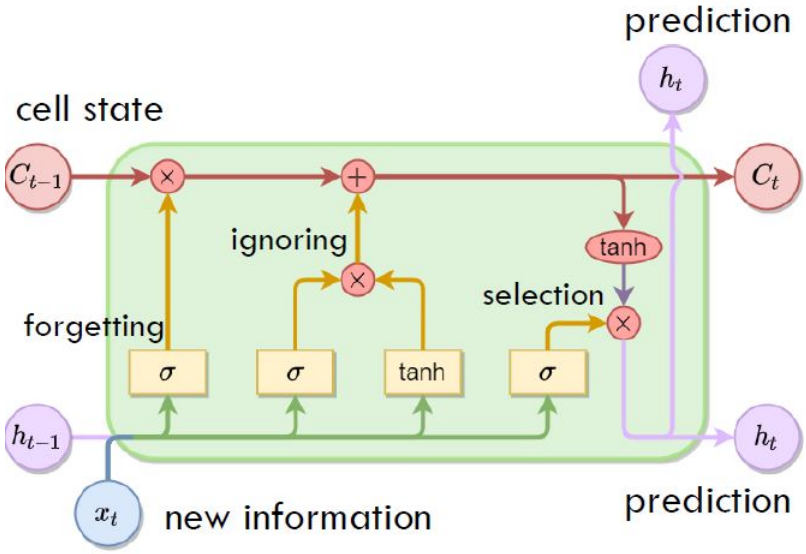

Fig. 4. LSTM Architecture

7 Our Prediction Model

We use LSTM RNN as the time series prediction model, the network consists of hidden recurrent layers followed by an output layer, where the number of

hidden recurrent layers and the number of units in each layer vary for each dataset. Two consecutive recurrent layers are fully connected with each other. To avoid overfitting dropout is used between two consecutive layers. The output layer is a fully connected dense NN layer, and we use linear activation in it, where MAE as the loss function, and Adam as optimizer. we fit the model to our training data and train it for 50 epochs and with validation$_split$ = $0.1, then we plot the training losses to evaluate our model's performance.$

8 The Anomaly Detection Method

The anomaly detection algorithm used in the project consists of two main steps. First, a summary prediction model is built to learn normal time series patterns and predict future time series. Then anomaly detection is performed by computing anomaly scores from the prediction errors.[2]
Here is in Fig 6 we plot the real data (Testing data) as shown

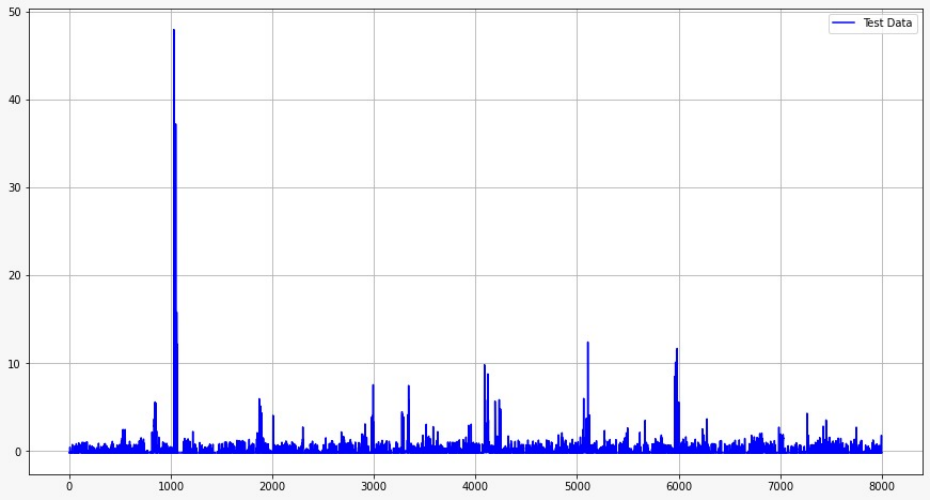

Fig. 6. The Testing Data

And in Fig. 7 we plot the Prediction data

We do the anomaly detection by using the prediction errors as anomaly indicators, which is define as the difference between prediction made at time t – 1 and the input value received at time t. On new data, the log probability densities PD of errors are calculated and used as anomaly scores: with lower values indicating a greater likelihood of the observation being an anomaly. Then we set a threshold on log PD values that can separate anomalies from normal

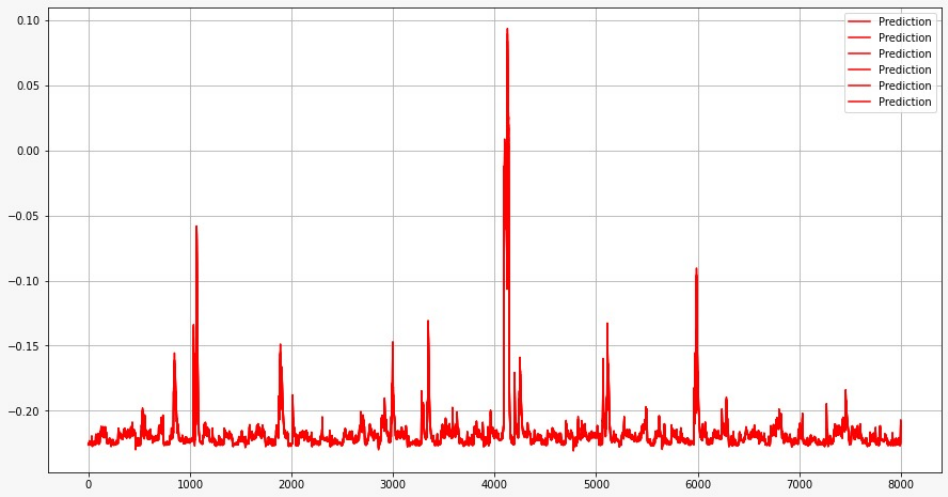

Fig. 7. The Predicting Data

observations and incur as few false positives as possible. A separate test set is used to evaluate the model.

After calculating the threshold and found it's equal to 2.975480894423082, we detected 80 anomalies. The Fig. 5. Show the response$_{t}imecorrespondingtothetimestampwithredpointrefertoth$

9 Visualization

Matplotlib is a 2-D plotting library that helps in visualizing figures, and by using it we explore our data set, like if we want to check the frequency of logs during days and nights of weekdays and weekends, we are going to preprocess our data accordingly. First, we will specify hours, then days, then weekdays and nights. Finally, we will visualize the temperature during these time-periods using a histogram. below histogram shows that the Logs are comparatively more stable during Week Days in the daylights, and extremely high between October and December especially at the weekends. Additional visualizations were made using Plotly, Scipy, Msticpy for showing more statistical outputs.

10 SARIMAx

Visualizing the log according to the frequency in the month we can notice that in November we have the most operation logs, as Fig. 7 The respond time corresponding to the timestamp plot

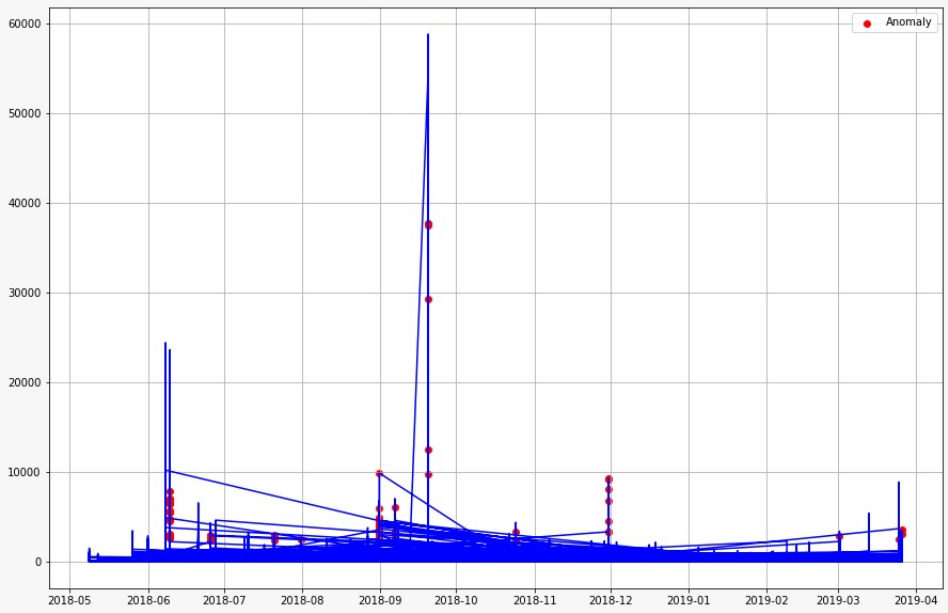

Fig. 8. The Anomalies

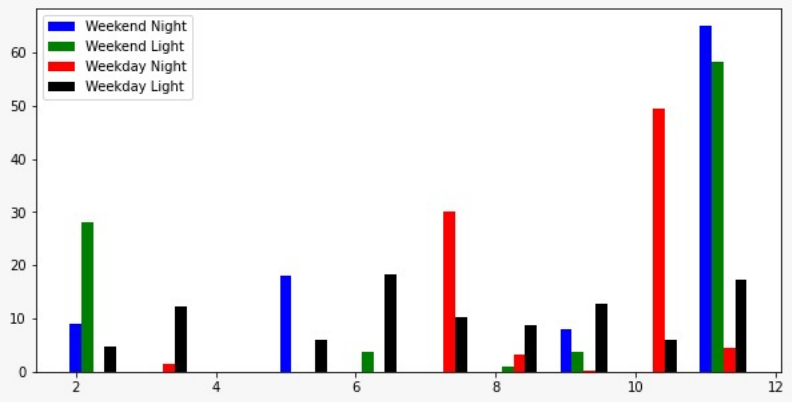

Fig. 9. The Log month histogram

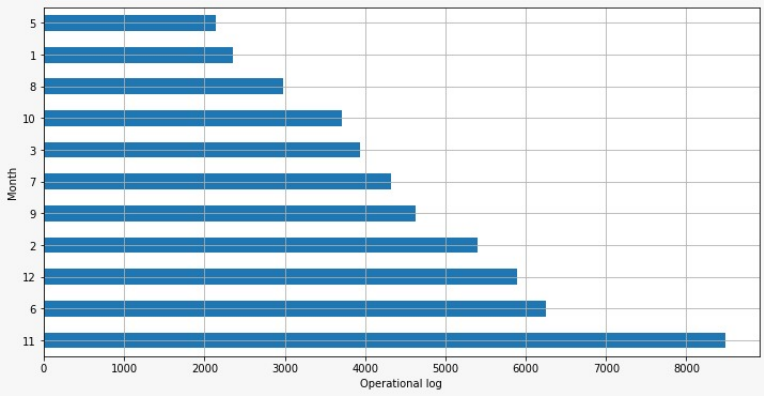

Fig. 10. Frequency in the month

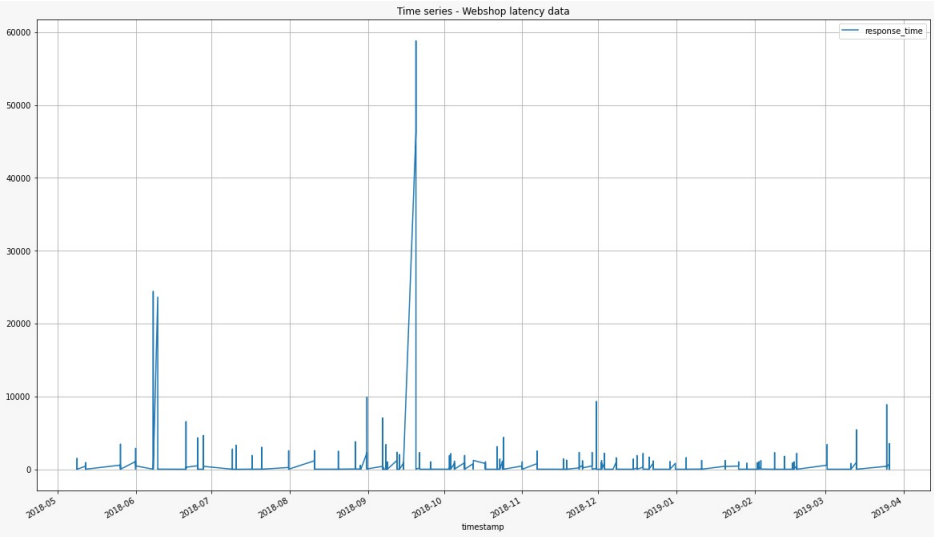

Fig. 11. Time series - Webshop latency data

References

1. Grover, A.: Anomaly Detection for Application Log Data. San Jose State University (2018).
2. SINGH, A.: Anomaly Detection for Temporal Data using Long Short-Term Memory (LSTM).KTH ROYAL INSTITUTE OF TECHNOLOGY, STOCKHOLM, SWEDEN (2017).
3. The dataset, https://dsd.inf.elte.hu/nextcloud/s/6w5dRkWe3krpKe2. Last accessed 14 Dec 2020
4. The dataset, https://kafka.apache.org/documentation/connect_overview. Last accessed 18 Dec 2020
5. The dataset, https://cassandra.apache.org/doc/latest/. Last accessed 26 Dec 2020
6. The dataset, http://colah.github.io/posts/2015-08-Understanding-LSTMs/. Last accessed 30 Dec 2020