Large-scale Anomaly Detection based on Ensemble Learning

Xi Zhang[1]
[1]Huawei Technologies Co. LTD., Huawei Industrial Base Bantian, Longgang District，
Shenzhen 518129 P.R.China

Abstract
Nowadays large companies have many systems and applications built as web-based services, to ensure undisrupted operations, one needs to closely monitor various metrics, such as total number of users, response time, or usages. Detecting anomalies in key metrics and making timely troubleshooting is crucial to prevent potential failure on relevant applications. This paper proposed automated ensemble anomaly detection methods composed by creating more than 20 different detectors, and over 15 different machine learning detection models, which is designed for large-scale metrics to be detected and lack of anomaly labeling. We compared our methods with published research work done by well-known professor Dan Pei and his students, our results are in competitive positions, and have been proved practical working, accurate, and efficient in real-world production.

Key Words: Anomaly Detection, Ensemble learning, Machine learning

1. Introduction

Today due to the rapid digital transformation of enterprises, many companies run their business through online service systems, such as Google, Amazon, Microsoft and many large companies, consists of thousands of distributed components and support a large number of concurrent users[5]. Engineers need to actively monitor various key performance metrics (KPI *henceforth*) and write many rules to trigger alerts. For example, if the response time of an API exceeds a given threshold (e.g. 15 seconds), an alert is generated to notify the API responsible engineer. Then this engineer examines this alert to check if there is a potential system outage. However, simple manual rules cannot sufficiently adapt to the dynamic system behaviors, nor capture the patterns of complex and interacting factors that influences the alerts. Furthermore, it is labor intensive for engineers to manually define and maintain rules, because:

1) Different business application systems have different behaviors, which makes different types of alerts
2) New types of alerts might be added due to system changes
3) Engineers may have different preference to handle alerts [6].

There are many literatures focusing on detections of maintenance induced changes in service performance, see [5]-[9] for details. In [1], D. Liu et al. 2015 proposed a supervised machine learning based framework ‘Opprentice’ with following aspects. Performance data were extracted into difference anomaly features, a.k.a Detectors. Operators are asked to label the anomalies in the performance data with a convenience tool. Then the features and labels are used to train a random forest classifier to automatically select the appropriate detector-parameter combinations and the thresholds.

H. Xu et al 2018 [2] proposed *Donut,* an unsupervised anomaly detection based on Variational Auto-Encoder (VAE). Meanwhile, Zeyan Li et al 2019 [10] proposed an improved framework *Bagel*, a robust and unsupervised anomaly detection algorithm for KPI that can handle time information related anomalies, using Conditional Variational Auto-Encoder(CVAE) to incorporate time information and dropout layer to avoid overfitting. Z. Li 2018 [11] proposed *ROCKA*, a robust and rapid KPI clustering algorithm, which consists of four steps: preprocessing, baseline extraction, clustering and assignment.

Our paper focused on real-time large-scale KPI anomaly detection, which faces two major challenges:

1. Dealing with over 20000 KPIs at real-time
2. High accuracy is required to ensure the system operating normally (high precision and high recall are both required)

The major contributions of the paper are summarized as follows. First, to the best of our knowledge, we handle over 20000 KPIs at real-time, which includes: preprocessing, feature extraction, detecting algorithms, and alarm sending. We achieved supreme precision: over 90%, and prefect recall: 100%. Second, we proposed an unsupervised machine learning algorithm, which does NOT require labelling at all. Third, we make good combination between statistical methods and machine learning algorithm, with which all models can produce anomaly detection results within 200ms for single KPI series.

The rest of this paper is organized as follows. In Section 2, we describe the background of business problems, and the current challenging part. Then we specify our methodology and the feature engineering work in Section 3. Experimental results are analyzed in Section 4. Finally, we conclude our work and put forward future work in Section 5.

2. Problem setting

2.1 A real system failure case

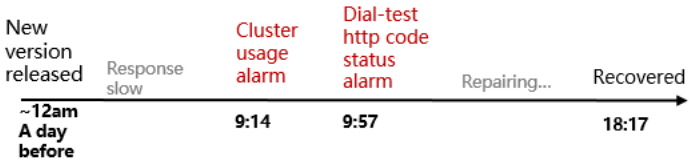

Figure 1: A system failure case caused by new version update with old SQL sequences to process data.

Figure 1 describes the real system failure cases happened before. The evening before the failure, Software Engineering (SE *henceforth*) released the new version of the application, checked the data process, tested the system function, which all looked fine. However, the next day morning, many users experienced slow response of that app, which were under the tolerance of users in the early morning before 8am without many

users using at one time. However, after 9am, the application can't work at all, and two serious types of alarms send out: the application cluster usage over 80% alarm, and the dial test http code status 500 alarm. SREs and SEs worked hard whole day to figure out which part caused this failure and finally around 6pm they fixed the problem, and the application system functioned normally.

This case intrigued the SREs to seek help: is there any technique that can tell them if there is a potential system outage in the coming 20 minutes or 1 hour? Fortunately, the famous Heinrich law says: "that in a workplace, for every accident that causes a major injury, there are 29 accidents that cause minor injuries and 300 accidents that cause no injuries." We, statisticians, were boarding to seek such methods that we can tell from the historical data performance and detect if there are any anomalies in the new coming KPI performance.

2.2 Problem Setting

We structure the problem as a time series anomaly detection problem. We define an anomaly as follows:

At any time t, given historical observations $x_{\{t-T+1\}}, \ldots, x_t$. *Determine whether an anomaly occurs* ($y_t = 1$).

$$P(y_t = 1 | x_{\{t-t+1\}}, \ldots . x_t)$$

We asked the SREs to collect the historical data for each KPI, and we use Flink to aggregate the raw collected data into certain time interval, i.e. 5mins average. Then we extract features from the aggregated data. Meanwhile, the SREs asked us to send them back the potential system failure alarm within 5 mins after the data aggregated.

3. Our Methodology

In order to overcome aforementioned two major challenges, we designed a robust and rapid algorithm framework:

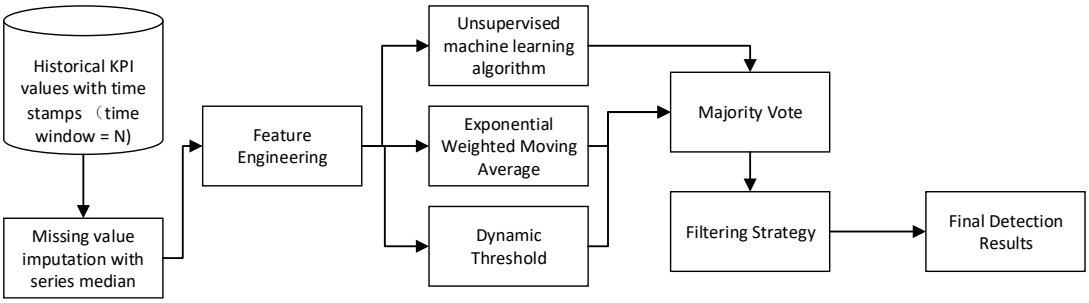

Figure 2: our robust and rapid anomaly detection framework

First of all, we impute the missing points with median of the closest historical N observations for each KPI. Secondly, we start the feature engineering process, which we will describes in details later. Then we use three kinds of detections methods: unsupervised machine learning algorithm, exponential weighted moving average with 3-sigma， and dynamic threshold, which composites 17 detection results. Thirdly, we gather all 17 results, and set up a voting rule, such as 90% of the results determine the current observation is an anomaly as a true anomaly from the algorithm. Fourthly, we

incorporate some domain knowledge, such as during 5 consecutive time window, if there is more than 3 anomalies then the alarm should be sent out. Finally, the SREs and the application system owners will be notified with the detection results and explanation.

3.1 Feature Engineering

As mentioned in [1], inspired by [15, 16], they represent different detectors with a unified model:

$$\text{Data point} \xrightarrow{\text{a detector with parameters}} \text{severity} \xrightarrow{\text{sThld}} \{1, 0\}$$

We follow the detector requirements in [1], and specified following 8 detectors with 28 configurations:

Table 1: Basic detectors and sampled parameters. Abbreviation: win(dow).

Detectors/# of configurations	Sampled parameters
Simple threshold	none
Diff/3	last-slot, last-day, last-week
Simple moving average	win= 10, 20,30,40,50 points
Weighted moving average	
Moving average of diff	
Exponential weighted moving average	alpha = 0.1, 0.3, 0.5, 0.7, 0.9
Median Absolute Distance	win= 1 week
Dynamic threshold	win = 1， 4 week(s)
In total: 8 basic detectors/28 configurations	

Simple threshold was already used by SREs, for example, they set up a threshold as 15s for response time, which simply intrigue a failure alarm when the response time of an application is greater than or equal to 15s. "Diff" simply measures anomaly severities using the difference between the current point and the point of last slot, the point of last day, and the point of last week. "Moving average" aims at identifying local anomalies with a short window. "Moving average of diff" measures severities using the moving average of the difference between current point and the point of last slot. "Exponential weighted moving average (EWMA *henceforth*)" uses following formula:

$$S_t = \begin{cases} Y_1, t = 1 \\ \alpha * Y_t + (1 - \alpha) * S_{t-1}, t > 1 \end{cases}.$$

α is the decay factor, which is specified as 0.1, 0.3, 0.5, 0.7, and 0.9. Using above formula, we transfer the original data points Y_t into the EWMA series S_t. "Median Absolute Distance" around the median instead of the standard deviation around the mean, which can improve the robustness to missing data and outliers [8, 17]. "Dynamic threshold" simply uses the 90th percentile of the data points within a window, 1 week or 4 weeks.

3.2 Ensemble Method

We adopt the classical machine learning methodology: ensemble method and majority vote. However with little changes, since we don't have labels from the historical data

points, we combine the results comparing with thresholds and set up a majority vote level as 90% of the model. The majority vote idea is described in Figure 3.

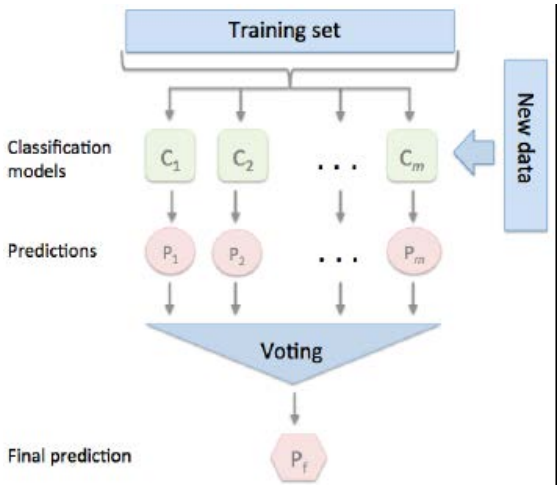

Figure 3: how majority voting works in classification

3.2.1 Isolation Forest

In the interest of space, we only introduce some basic ideas of isolation forest. More details are in [12, 21]. Isolation forest builds an ensemble of iTrees for a given data set, then anomalies are those instances which have short average path lengths on the iTrees. There are only two variables in this method: the number of trees to build and the sub-sampling size. The algorithm is described in Figure 4.

Algorithm 1 $iTree(X, e, h)$

Input: X - input data; e - current height; h - height limit.
Output: an $iTree$.

1: **if** $e \geqslant h$ OR $|X| \leqslant 1$ **then**
2: **return** $exNode\{Size \leftarrow |X|\}$;
3: **else**
4: Randomly select an attribute q;
5: Randomly select a split point p between min and max values of attribute q in X;
6: $X_l \leftarrow filter(X, q < p)$, $X_r \leftarrow filter(X, q \geqslant p)$;
7: **return** $inNode\{ Left \leftarrow iTree(X_l, e+1, h)$,
$Right \leftarrow iTree(X_r, e+1, h)$,
$SplitAttr \leftarrow q, SplitValue \leftarrow p\}$;
8: **end if**

Figure 4: Isolation Forest Algorithm

A rich body of literature has been devoted to anomaly detection, e.g., One-Class SVM [18], Local Outlier Factor [19] and clustering based methods [20]. However, these algorithms suffer from either high computational cost or poor performance. Isolation Forest is a popular anomaly detection algorithm and has shown good performance with a linear time complexity [21], and thus we leverage Isolation Forest to detect anomalies in our scenario. The normal and outliers can be illustrated in Figure 5.

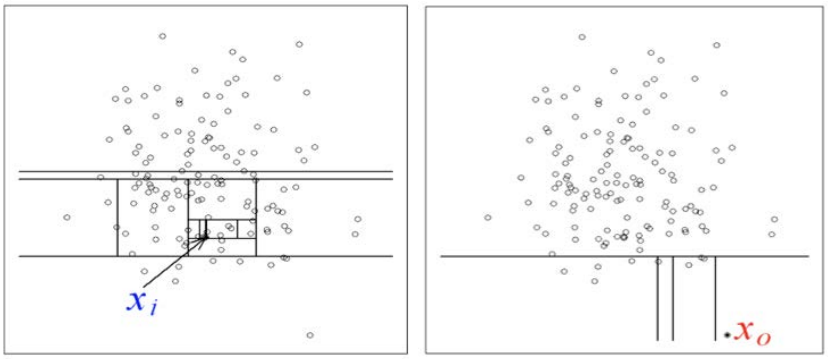

Figure 5: Normal data points (left) versus Outlier (right)

We use detectors with various configurations as input of Isolation Forest model, hence we got 10 different combinations shown in Table 2.

Table 2: 10 Isolation Forest models with different combinations of detectors.

Features	MODEL1	MODEL2	MODEL3	MODEL4	MODEL5	MODEL6	MODEL7	MODEL8	MODEL9	MODEL10
diff_last_slot	yes	yes	yes	yes	yes					
diff_last_day	yes	yes	yes	yes	yes					
diff_last_week	yes	yes	yes	yes	yes					
MA10	yes					yes				
MA20		yes					yes			
MA30			yes					yes		
MA40				yes					yes	
MA50					yes					yes
MA_diff_10						yes				
MA_diff_20							yes			
MA_diff_30								yes		
MA_diff_40									yes	
MA_diff_50										yes
EWMA_0.1	yes					yes				
EWMA_0.3		yes					yes			
EWMA_0.5			yes					yes		
EWMA_0.7				yes					yes	
EWMA_0.9					yes					yes

3.2.2 EWMA with adjusted 3-Sigma

For this method, we made a little change to the confidence interval method with adjusted standard deviation, which makes the threshold band even larger (i.e. upper level threshold higher, lower level threshold smaller) . First the mean and standard deviations were derived from aforementioned EWMA S_t series, then the upper level and lower level with over 99% confidence level were calculated with this formula:

$$mean \pm 3\sigma_{adjusted}, \sigma_{ajusted} = \sigma * \sqrt{\frac{\alpha}{2-\alpha}}$$

3.2.3 Statistical Discrimination method

In order to find extreme large value of a series quickly, such as data points within a week or 4 weeks, we adopt the percentile method: we calculate the 90th percentile of a series,

and if the current data point is greater than the 90th percentile of the past 1 or 4 week's data points, then this data point is marked as outlier.

4. Results

The fundamental goal of anomaly detection is to be accurate, i.e. identifying more anomalies and avoiding false alarms. We use recall and precision formulas specified as follows to measure the detection accuracy. Precision describes what matters to SREs better than false positive rate (FPR), because anomalies are infrequent [22]. Precision is also equivalent to 1-FDR (false discovery rate).

$$\text{Precision} = \frac{TP}{(TP + FN)} = \frac{\#\ of\ true\ anomalous\ points\ detected}{\#\ of\ anomalous\ points\ detected}$$

$$\text{Recall} = \frac{TP}{(TP + FP)} = \frac{\#\ of\ true\ anomalous\ points\ detected}{\#\ of\ true\ anomalous\ points}$$

$$\text{FDR} = \frac{\#\ of\ false\ anomalous\ points\ detected}{\#\ of\ anomalous\ points\ detected}$$

We asked our SREs to give feedbacks on the anomalous points detected with a convenient online tool. The following Table 3 shows the summary feedback during one earlier week after using our methodology. Currently the SREs feed that the precision of our detection method is higher than 90%, and recall is over 95%.

Table 3: Summary of evaluation during one week.

Evaluation Metric	Score	Formula
Recall	89% （24/27）	TP/(TP+FN)
Precision	87.3%（152/174）	TP/(TP+FP)
F-Score	88.30%（beta=1.2）	$\frac{(1+\beta^2) * Precision * Recall}{\beta^2 * Precision + Recall}$

5. Conclusions and Feature work

Applying anomaly detection to an Internet-based service has been challenging in practice. This is because the anomalies are difficult to quantitatively define, and existing detectors have parameters and thresholds to tune before they can be deployed. Our proposed framework tackles the above challenges through an unsupervised machine learning based approach, which overcomes the "No label" issue. The unclear anomaly concepts are captured by machine learning from real data, while numerous existing detectors can be automatically combined by machine learning to train an Isolation Forest classifier to identify the anomalies. Our evaluation on real-world KPIs show that our method consistently performs very well with high precision and high recall.

Acknowledgements

This work was originated from industrial daily Site Reliability Engineers in our company. They actively monitors tons of systems every day. Thanks to their professional domain knowledge, and SRE experiences, without which we could not understand the business

insights behind those numbers of system KPIs. We would also thank many coworkers, for their valuable inspirations and feedbacks.

References

[1] D. Liu, Y. Zhao, H. Xu, Y. Sun, D. Pei, J. Luo, X. Jing, and M. Feng, "Opprentice: towards practical and automatic anomaly detection through machine learning," in Proc. of IMC. ACM, 2015, pp. 211–224.
[2] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, and et.al, "Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications," in WWW, 2018.
[3] Nengwen Zhao, Jing Zhu, Rong Liu, Dapeng Liu, Ming Zhang, and Dan Pei. 2019. Label-Less: A Semi-Automatic Labelling Tool for KPI Anomalies. In IEEE INFOCOM 2019-IEEE Conference on Computer Communications. IEEE, 1882–1890.
[4] Breiman, L. 2001. Random forests. *Machine Learning 45,* 1, 5–32.
[5] Q. Lin, H. Zhang, J.-G. Lou, Y. Zhang, and X. Chen, "Log clustering based problem identification for online service systems," in Proceedings of the 38th International Conference on Software Engineering Companion, pp. 102–111, ACM, 2016
[6] G. Jiang, H. Chen, K. Yoshihira, and A. Saxena, "Ranking the importance of alerts for problem determination in large computer systems," Cluster Computing, vol. 14, no. 3, pp. 213–227, 2011.
[7] Yingying Chen, Ratul Mahajan, Baskar Sridharan, and Zhi-Li Zhang. A provider-side view of web search response time. In *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, pages 243–254. ACM, 2013.
[8] Ajay Anil Mahimkar, Han Hee Song, Zihui Ge, Aman Shaikh, Jia Wang, Jennifer Yates, Yin Zhang, and Joanne Emmons. Detecting the performance impact of upgrades in large operational networks. In *Proceedings of the ACM SIGCOMM 2010 Conference*, SIGCOMM '10, pages 303–314. ACM, 2010.
[9] Ajay Mahimkar, Zihui Ge, Jia Wang, Jennifer Yates, Yin Zhang, Joanne Emmons, Brian Huntley, and Mark Stockert. Rapid detection of maintenance induced changes in service performance. In *Co-NEXT*, page 13. ACM, 2011.
[10] Zeyan Li, Wenxiao Chen, and Dan Pei. "Robust and Unsupervised KPI Anomaly Detection Based on Conditional Variational Autoencoder." 2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC). IEEE, 2018.
[11] Z. Li, Y. Zhao, R. Liu and D. Pei, "Robust and Rapid Clustering of KPIs for Large-Scale Anomaly Detection," *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, Banff, AB, Canada, 2018, pp. 1-10, doi: 10.1109/IWQoS.2018.8624168.
[12] F. Liu, K. Ting and Zhihua Zhou, "Isolation-based Anomaly Detection", *ACM Transactions on Knowledge Discovery from Data* , March 2012 Article No.3 https://doi.org/10.1145/2133360.2133363
[13] N. Zhao, P. Jin, L. Wang, X. Yang, R. Liu, W. Zhang, K. Sui, D. Pei. *INFOCOM 2020, Virtual Conference*, Jul 6-9, 2020
[14] N. Zhao, J. Chen, X. Peng, H. Wang, X. Wu, Y. Zhan, Z. Chen , X. Zheng, X. Nie, G. Wang, Y. Wu, F. Zhou, W. Zhang, K. Sui, D. Pei, "Understanding and Handling Alert Storm for Online Service Systems", *ICSE SEIP 2020, Virtual Conference*, Jul 6-11 2020
[15] Shashank Shanbhag and Tilman Wolf. Accurate anomaly detection through parallelism. *Network, IEEE*, 23(1):22–28, 2009.
[16] F. Silveira and C. Diot. Urca: Pulling out anomalies by their root causes. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9, March 2010.

[17] Benjamin I.P. Rubinstein, Blaine Nelson, Ling Huang, Anthony D. Joseph, Shing-hon Lau, Satish Rao, Nina Taft, and J. D. Tygar. Antidote: Understanding and defending against poisoning of anomaly detectors. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '09, pages 1–14, New York, NY, USA, 2009. ACM.
[18] Mennatallah Amer, Markus Goldstein, and et.al. 2013. Enhancing one-class support vector machines for unsupervised anomaly detection. *In SIGKDD*. ACM.
[19] Markus M. Breunig, Hans-Peter Kriegel, and et.al. 2000. LOF: Identifying Density based Local Outliers. *In SIGMOD*. ACM.
[20] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM computing surveys (CSUR) 41*, 3 (2009), 15.
[21] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In 2008 *Eighth IEEE International Conference on Data Mining*. IEEE, 413–422.
[22] Alexey Tsymbal. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*, 2004.