

# Real-time Statistical Log Anomaly Detection with Continuous AIOps Learning

Lu An<sup>a</sup>, An-Jie Tu, Xiaotong Liu and Rama Akkiraju

*IBM Watson, 555 Bailey Ave, San Jose, U.S.A.*

**Keywords:** AI for IT Operations, Log Anomaly Detection, Online Statistical Learning, Error Entity Extraction, Continuous Model Updating.

**Abstract:** Anomaly detection from logs is a fundamental Information Technology Operations (ITOps) management task. It aims to detect anomalous system behaviours and find signals that can provide clues to the reasons and the anatomy of a system's failure. Applying advanced, explainable Artificial Intelligence (AI) models throughout the entire ITOps is critical to confidently assess, diagnose and resolve such system failures. In this paper, we describe a new online log anomaly detection algorithm which helps significantly reduce the time-to-value of Log Anomaly Detection. This algorithm is able to continuously update the Log Anomaly Detection model at run-time and automatically avoid potential biased model caused by contaminated log data. The methods described here have shown 60% improvement on average F1-scores from experiments for multiple datasets comparing to the existing method in the product pipeline, which demonstrates the efficacy of our proposed methods.


## 1 INTRODUCTION

The exploding growth of Information Technology (IT) systems and services make the systems and applications become increasingly more complex to operate, manage and monitor. By utilizing log processing, machine learning and other advanced analytics technologies, Artificial Intelligence for IT Operations (AIOps) (Lerner, 2017) provides a promising solution to enhance the reliability of the IT operations. Today, most planet-scale service operators employ their own AIOps to collect logs, traces and telemetry data, and analyze the collected data to enhance their offerings (Levin et al., 2019). One of the critical tasks in AIOps is the anomaly detection which is the essential step to detect anomalous system behaviours and find signals that can provide clues to the reasons and the anatomy of a system's failure (Goldberg and Shan, 2015; Gu et al., 2017; Chandola et al., 2009).

As system logs are records of the system states and events at various critical points and log data is universally available in nearly all IT systems, it is a valuable resource for the AIOps to process, analyze and perform anomaly detection algorithms. We call the anomaly detection methods utilizing logs as data source as Log Anomaly Detection (LAD). The tradi-

tional LAD methods were mostly manual operations and rule-based methods, while such methods were no long suitable for the large-scale IT systems with sophisticated system incidents. In the recent years, with the development of AI technologies, machine learning based anomaly detection methods have received more and more attention. For instance, some works utilized unsupervised clustering-based methods (Givental et al., 2021a; Givental et al., 2021b) to detect outliers. Though such methods do not require labeled data for training, the anomaly detection performance is not guaranteed and unstable. Moreover, it is hard to apply such methods onto streaming log data as the log patterns are changing over time.

Another popular and widely used LAD approach is to first collect enough labeled training data during the system's normal operation period and adopt log templates-based method for feature engineering, and then employ Principal Component Analysis (PCA) based methods to learn normal log patterns from labeled training data and find anomalous log patterns during inference streaming log data (Liu et al., 2020; Liu et al., 2021). Even though the PCA-based method is successful in certain scenarios, it still suffers from some limitation in practice. Firstly, log templates learning often requires customers to provide one week's worth of training logs without incidents

<sup>a</sup>  <https://orcid.org/0000-0003-4050-3625>

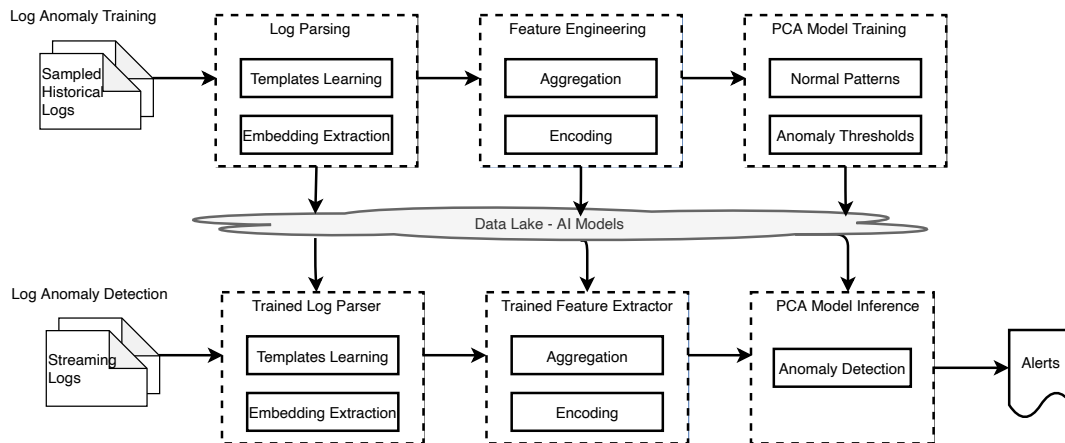


Figure 1: LAD pipeline for PCA-based Method.

which may need some time to collect thus training data is not available on Day 0. Moreover, the customers sometimes may not know if the training logs they provided are pure normal logs. Secondly, the log templates learning process takes hours or days to finish, depending on the size of the datasets.

In this paper, we propose a Real-Time Statistical Model (RSM) based LAD method, which aims to reduce the training time and achieve faster “time-to-value” while performing excellent online anomaly detection. The major contributions of this new algorithm include:

- (i) We introduce a fast error entity extraction method to extract different types of error entities including error codes and exceptions at run-time. In addition, this method is able to quickly categorize if an incoming log contains any faults.
- (ii) Instead of utilizing log templates for feature engineering, the RSM algorithm adopts the extracted error entities to build feature count vectors to learn normal log patterns.
- (iii) The proposed RSM method is able to keep the model up-to-date by kick-starting the accumulative retraining periodically, so that the model is able to continuously improve itself by learning from more and more log data.
- (iv) We introduce an automatic skipping mechanism in the model updating which can help avoid biased model generated by contaminated log data.

The remainder of the paper is organized as follows: Section 2 outlines the log anomaly detection system. Section 3 describes the technical details of the RSM method. Section 4 shows the experiment results of the proposed RSM method on multiple benchmark datasets. Finally, Section 5 concludes the paper and summarizes the future directions of the work.

## 2 LAD SYSTEM DESCRIPTION

In this section, we introduce the major steps of the LAD pipeline and how the new RSM algorithm affected the steps of log anomaly detection compared with the PCA-based method.

### 2.1 PCA-based Method

The PCA-based method is illustrated in Fig. 1 and it mainly includes the following steps:

- 1) **Log Anomaly Training:** We first ingested historical normal log data from log aggregators or streaming data from Apache Kafka. Then the data preparation component will apply data preprocessing to generate a clean, normalized log data and upload them into the cluster for further log template training. A tree-based templates learning algorithm is then applied onto the selected training logs to generate log templates, and these templates are used for feature engineering and building template/embedding count vectors.

After such count vectors are generated, they will be aggregated and encoded for a PCA model training where the model learns the normal patterns and the anomaly threshold from the training logs. After the model training, such models will be stored in the cluster for future inference.

- 2) **Log Anomaly Detection:** During the inference, the data preparation component first utilizes the trained log templates generated in the training step to perform feature engineering on the streaming inference log data. Such template/embedding count vectors are aggregated and encoded and then sent to the log anomaly detector with other metadata.

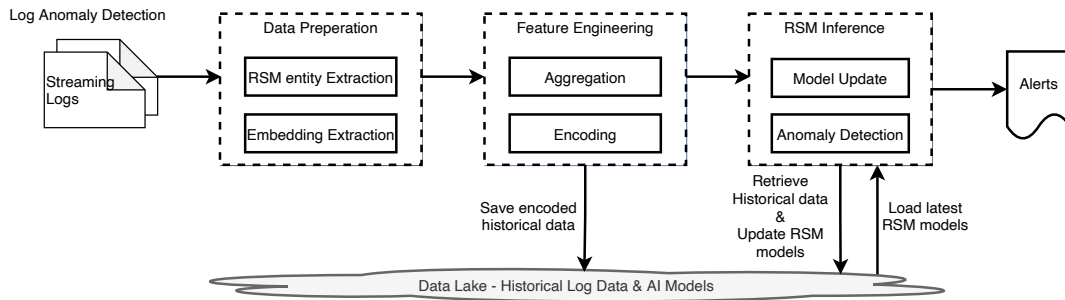


Figure 2: LAD pipeline for RSM-based Method.

Log anomaly detector then retrieves the relative PCA models from the cluster and applies them onto the encoded inference data. Those inference logs with PCA scores exceeding the trained threshold will be tagged as anomalies.

## 2.2 RSM-based Method

By introducing the RSM-based method into the product pipeline, the training step is eliminated and the major phases of the LAD pipeline become as follows and are shown in Fig. 2:

1) **Data Preparation:** In RSM-based method, data preparation component will directly apply entity extraction algorithm onto the streaming inference data. Instead of utilizing log templates, these extracted entities will be aggregated and encoded to generate feature count vectors. Along with other metadata, the encoded feature count vectors will be stored into the clusters and sent out to log anomaly detector simultaneously for inference.

2) **Model Updating:** After a preset time period, the LAD is able to retrieve all the encoded log data from the cluster ingested during the last time period and compute the statistical distribution for all the entities. Such statistical distribution information for both entity and embedding count vectors are stored as RSM models in the cluster for future anomaly detection. The model updating is scheduled to happen periodically unless contaminated log data was detected in the last period. Moreover, the model updating is accumulative so that the latest model always represents all the historical data the LAD has seen.

3) **Log Anomaly Detection:** Once the initial RSM models are available, the log anomaly detector is able to load the latest models from the cluster and utilize them to perform statistical testing and determine if the inference logs contain significantly different entities or embedding distributions than the normal patterns.

If yes, the inference logs will be tagged as anomalies and generate alerts.

From the above description of PCA-based and RSM-based methods, we can notice that the log anomaly training step in PCA-based method will need customers to provide some normal training log data. On the one hand, one week log data is typically required to guarantee a good quality of log training. On the other hand, larger size of dataset may cause long time template learning and model training. While in the RSM-based method, the online learning method utilizes entity extraction instead of log templates learning so that the time-consuming training step is eliminated which can significantly reduce “time-to-value” of the LAD pipeline.

## 3 THE DESIGN OF RSM-BASED METHOD

In this section, we will discuss the technical key-points of the Real-Time Statistical Model based log anomaly detection method proposed in this work in details.

### 3.1 Log Entity Extraction

```

"message": "SRVE0315E: An exception occurred: java.lang.Throwable:
java.lang.NullPointerException\n\tat
com.ibm.ws.webcontainer.webapp.WebApp.handleRequest(WebApp.java:5115)\n\tat ....
"threadid": "00000f0b", "datetime": "2021-03-25T14:16:35.490-0400",
"messageid": "SRVE0315E",
"module": "com.ibm.ws.webcontainer.webapp",
"loglevel": "SEVERE"
    
```

Figure 3: A log example from WebSphere Application Server.

The RSM-based LAD product pipeline utilizes log entities for feature engineering. Such log entities could be the domain information from specific types of logs, e.g. the logs from IBM WebSphere Application Server (IBM, 2022). In addition, the log entities could be specific errors, such as HTTP error response code or specific exceptions which can be the cause or

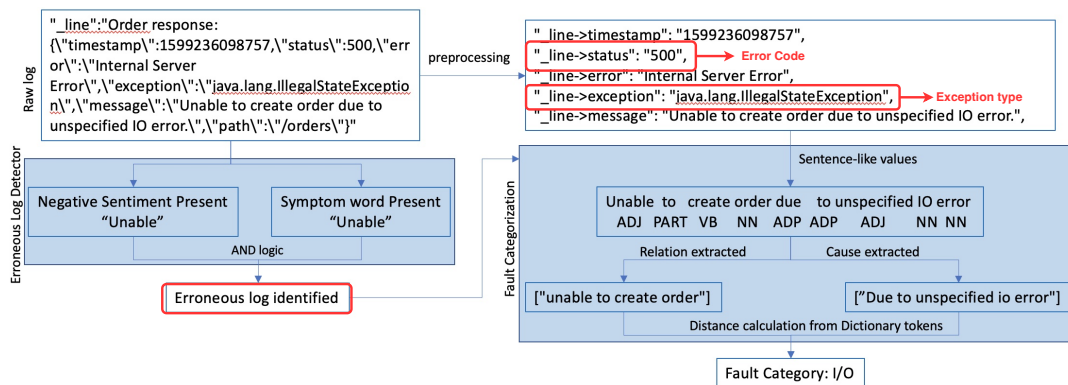


Figure 4: An example of error entity extraction for general logs.

symptoms for system incidents. During the data preprocessing phase, the data preparation component in the LAD pipeline is responsible for extracting such log entities in real-time and then aggregating and encoding them to form the feature count vectors for further anomaly detection.

### 3.1.1 WebSphere Logs

The IBM WebSphere Application Server is a flexible, security-rich Java server runtime environment for enterprise applications. Each WebSphere log contains a designated message ID or log level, or both. Fig. 3 shows an example of a WebSphere log which contains a specific message ID “SRVE0315E” and the log level “SEVERE”. Based on the prior domain knowledge of the WebSphere, such message ID’s and log levels are indicators of specific types of abnormal system behaviors. The task of the data preparation component is to identify if an incoming log is from a Websphere Application Server and then extract any shown message ID’s and log levels for feature engineering.

### 3.1.2 General Logs

For all other non-WebSphere types of logs, during the log data preparation stage, we utilize the SystemT framework (Krishnamurthy et al., 2009) to define the rules for log entity extraction (Mohapatra et al., 2018; Aggarwal et al., 2021; Aggarwal and Nagar, 2021). First, the sentiment analysis is performed on the log as log messages usually contain missing or faulty attributes while encountering any system errors. In order to satisfy the run-time analysis requirement, we adopt a dictionary-based approach as opposed to a full-blown machine learning approach. The negative sentiment dictionary used in our product pipeline was built based on our technical domain leveraging on open-source sentiment dictionaries such as Vader (Hutto and Gilbert, 2014) and SentiWordNet (Bac-

cianella et al., 2010). We have discarded some nouns candidates and added some words denoting negation because in log data, negative sentiments are mostly associated with actions which mostly comprise of verbs, adverbs or adjectives.

In addition, the relation and cause extraction are performed to extract any detected error codes and exceptions. Relation extraction includes the following steps: (1) Dependency parsing; (2) Verb filtration and Clause/Predicate selection; (3) Extension of Noun phrase; (4) Negative sentiment; (5) Relation clause/phrase generation. For cause extraction, we consider the following four rules: (1) Presence of Causative Verbs; (2) Presence of Phrasal Verbs; (3) Presence of prepositional Adjective or Adverbial Phrases; (4) Absence of Noun Phrase.

With the above sentiment analysis, relation and cause extraction tools, we are able to extract error codes, exceptions’ types from the logs if present, and also identify if a log message indicates erroneous system behaviour based on symptoms or negative sentiment dictionaries. Fig. 4 shows an example of the entity extraction for general logs. From Fig. 4, we can observe that the extracted error code is “500” and the exception type is “java.lang.IllegalStateException” and this log message is identified as an erroneous log based on the sentiment and symptoms.

## 3.2 Feature Encoding

By employing the above proposed log entity extraction method, the data preparation component is able to extract message ID’s and log levels for WebSphere logs and extract error entities for general erroneous logs. Those general logs without any error entities will be tagged as “normal” logs. We use these extracted entities to build the feature count vectors.

To build the feature count vectors, we first group any logs that occurred in a preset time period  $T$  to-

gether, e.g. every 10 seconds, thus all logs in one time period forms a time window. Assuming there are  $M$  logs in the time window  $t$ , labeled as  $L_1, L_2, \dots, L_M$ , and there are total of  $N$  different possible entities that can be extracted out for all the logs in time window  $t$ . If the  $n^{\text{th}}$  entity is extracted from the  $m^{\text{th}}$  log  $L_m$ , then we denoted  $e_m^n = 1$ , otherwise  $e_m^n = 0$ . Thus, the feature count vector for this time window  $t$  is constructed as  $\mathbf{X}_t = [x_1, x_2, \dots, x_N]$  where the count of the  $n^{\text{th}}$  feature is given by:

$$x_n = \sum_{m=1}^M e_m^n. \quad (1)$$

### 3.3 RSM Model Update

After the customer connects streaming logs to the system, LAD will start running with an empty statistical baseline log anomaly detection model. The RSM model updating happens periodically, e.g. every 30 minutes. The Data lake shown in Fig. 2 stores the encoded historical log data and the corresponding feature count vectors happened during the last period. Thus, the first RSM model should be ready in 30 minutes based on the historical data within the previous 30 minutes.

RSM is a statistical-based log anomaly detection method, where the RSM model contains all the extracted entities' and embedding vectors' statistical metrics, such as mean value, standard deviation (std), co-variance, sample size, etc. Given a batch of encoded and time-windowed logs  $[\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_T]$  uploaded by the data preparation components during the last period and a previous RSM model  $\mathcal{M}_{n-1}$ , we are able to compute and update all the above statistical metrics which form a new RSM model  $\mathcal{M}_n$ , which reflects the latest statistical distribution during normal operation period.

Another key feature of the RSM model updating is the automatic skipping mechanism. Before the model updating, the LAD will automatically check the anomaly-to-windowed-log ratio, which is defined as the ratio of number of detected anomalies to the number of total windowed logs ingested in the last period. If the anomaly-to-windowed-log ratio exceeds a preset threshold, then the LAD will tag the last period as an incident period and skip the model update temporarily for this round to avoid potential biased RSM models. With this automatic skipping mechanism, the RSM model is able to remember only the statistical distribution for logs in normal operation period, without manual human interference.

## 3.4 RSM Anomaly Detection

RSM anomaly detection is comprised of two independent anomaly detection methods: an entity-based detection method and an embedding-based detection method. The entity-based method uses the statistical metrics from extracted entities shown in Section 3.1, while the embedding-based detection method uses the statistical metrics from extracted embeddings (Liu et al., 2020). The inference results from the two individual detection methods are aggregated together to produce a single output that forms the RSM anomaly detection's final inference result.

### 3.4.1 Entity-based Detection

In a RSM model  $\mathcal{M}$ , using historical data, we store the sample mean  $\mu_{e^n}$ , the sample standard deviation,  $\sigma_{e^n}$ , the sample size  $N_{e^n}$ , and the indicator variable  $I_{e^n}$  for the  $n^{\text{th}}$  extracted entity  $e^n$ .  $I_{e^n} = 1$  if  $e^n$  corresponds to an error entity otherwise  $I_{e^n} = 0$ . Given a time-windowed log  $\mathbf{X}_t$ , let  $x_n$  be the corresponding counts for entity  $e^n$  in that given time-window, then time-windowed log  $\mathbf{X}_t$  is flagged as an anomaly if it is the case that

$$1 - \Phi\left(\frac{\mu_{e^n} - x_n}{\sigma_{e^n}}\right) < \varepsilon \mid I_{e^n} = 1 \quad (2)$$

for any entity  $e^n$  and for some threshold  $\varepsilon$ , e.g. 0.05.  $\Phi$  denotes the CDF for the standard normal distribution.

### 3.4.2 Embedding-based Detection

The RSM model  $\mathcal{M}$  will also store a sample embedding mean vector  $\mathbf{M}_n = [\mu_1, \mu_2, \dots, \mu_n]$  for some dimension  $n$ , e.g. 20, and a  $n \times n$  covariance matrix  $\Sigma$  with element  $(i, j)$  representing the covariance between dimension  $i$  and dimension  $j$ . These two metrics are also computed via the historical data that the RSM model observes. An incoming time-windowed log  $\mathbf{X}_t$  will contain a  $n$  dimension embedding vector,  $\mathbf{X}_{t,emb}$ , that corresponds to that time-window's embedding, we denote the Mahalanobis distance of  $\mathbf{X}_t$ ,  $MD(\mathbf{X}_t)$  as

$$MD(\mathbf{X}_t) = \sqrt{(\mathbf{X}_{t,emb} - \mathbf{M}_n)\Sigma^{-1}(\mathbf{X}_{t,emb} - \mathbf{M}_n)^{\top}}. \quad (3)$$

Under the assumption that the underlying  $n$  dimension embedding vector is a multivariate normal random variable, then it follows that

$$MD(\mathbf{X}_t) \sim \sqrt{\chi_n^2}, \quad (4)$$

with  $\chi_n^2$  being a chi squared distribution with  $n$  degrees of freedom. We flag time-windowed log  $\mathbf{X}_t$  as



an anomaly if  $MD(\mathbf{X}_t)$  exceeds a certain threshold, e.g. the square root of the 95 percentile of a  $\chi_n^2$  distribution.

### 3.4.3 Aggregation Rule

Let our entity-based detection method's inference result on time-windowed log  $\mathbf{X}_t$  be denoted as  $RSM_{entity}(\mathbf{X}_t)$ .  $RSM_{entity}(\mathbf{X}_t) = 1$  if  $\mathbf{X}_t$  is flagged as an anomaly by the entity-based model and 0 otherwise. Correspondingly, let our embedding-based detection method's inference result on  $\mathbf{X}_t$  be denoted as  $RSM_{embedding}(\mathbf{X}_t)$  and also be equal to 1 if flagged as anomaly by the embedding-based model and 0 otherwise. We aggregate the two models' results to create our RSM model's inference result on  $\mathbf{X}_t$ ,  $RSM(\mathbf{X}_t)$  to be

$$RSM(\mathbf{X}_t) = RSM_{entity}(\mathbf{X}_t) \vee RSM_{embedding}(\mathbf{X}_t). \quad (5)$$

## 4 EXPERIMENT RESULTS

In our experiments, We compare the anomaly detection performance of the LAD product pipeline before and after integrating the proposed RSM-based method with multiple datasets. The previous version of the LAD product pipeline adopted the PCA-based method only. By integrating the proposed RSM-based method, the LAD product pipeline is able to turn on/off either PCA-based or RSM-based methods through system configuration. Thus, in this section, we will show evaluation results for multiple datasets with the following three system configuration: (1) PCA-based method only (2) RSM-based method only (3) PCA-based & RSM-based methods.

### 4.1 Datasets

- **Sockshop:** The sockshop application is a user-facing part of an online shop that sells socks, which contains many microservice components including management of the user account, catalog, cart, orders, payment, shipping, etc. The training data for PCA-based method was collected when the application was running in normal operation with simulated user flows for one week. The testing data was collected by running the system in normal operation first for at least 30 minutes and then manually ingested different types of system incidents, with recording the timestamps of abnormal periods to generate ground truth labels.
- **Quote of the Day (QotD):** The QotD application is another simulated micro-service applica-

tion which randomly selects a quote from a famous person. The application contains many micro-services including rating, image, author, etc., which manage the different attributes of the quotes. Similarly, we first collect one week normal data for PCA-based log training. The testing data was collected in the similar way: normal logs first, followed by abnormal logs caused by manually ingested system errors.

- **IBM Watson AI service (WA):** In contrast to the above two simulated systems, we also evaluate the LAD performance using the logs generated by the IBM Watson AI service which is a real system with 48 cloud micro-services and 15 applications, ranging from distributed systems, supercomputers, operating systems to server applications. The training data and testing data for WA were collected via a data collection method similar to what was shown previously for the other two datasets. However, the WA system is much more complex, which results in a much larger size of logs than Sockshop and QotD datasets.

### 4.2 Metrics

Log anomaly detection is one type of binary classification problems, where the positives are the anomalous log data and the negatives are those normal log data. However, unlike other binary classification problems, the log data for anomaly detection is very imbalanced as the positives are usually far way less than the negatives. The reason is that exceptions or incidents are rare cases for the mature systems compared to the normal operations. Thus, for such imbalanced testing log data, we evaluate the LAD performance with the following metrics:

- True Negative Rate (TNR):

$$TNR = \frac{TN}{N} \quad (6)$$

- False Positive Rate (FPR):

$$FPR = \frac{FP}{N} \quad (7)$$

- Recall or True Positive Rate (TPR)

$$Recall = \frac{TP}{N} \quad (8)$$

- F1-score

$$F1 = \frac{2TP}{2TP + FP + FN} \quad (9)$$

Table 1: The experiment results for three datasets with different LAD methods.

LAD method	Datasets	TNR	FPR	Recall	F1-score
PCA-based	Sockshop	0.73	0.27	0.5	0.04
RSM-based		1	0	0.17	0.29
PCA & RSM		0.99	0.01	0.67	<b>0.73</b>
PCA-based	QotD	0.99	0.01	1	0.99
RSM-based		0.99	0.01	1	0.99
PCA & RSM		0.99	0.01	1	0.99
PCA-based	WA	0.98	0.02	0.29	0.33
RSM-based		0.84	0.16	0.66	0.35
PCA & RSM		1	0	0.35	<b>0.52</b>

### 4.3 Results

Table 1 shows the experiment results comparison for the three datasets with different LAD methods. First, we observe that the PCA-based LAD method performs very well on QotD dataset due to the relatively simpler log structure. However, PCA-based method performs bad on the Sockshop and WA dataset where the log structure is more comprehensive and information or parameters inside the log are more fluctuating. From Table 1, we observe that compared with the PCA-based LAD method, the proposed RSM-based method can improve the F1-score on both Sockshop and WA datasets without downgrading the LAD performance on QotD dataset. In addition, the LAD product pipeline allows both models to be enabled to further improve the detection accuracy. We can observe that by enabling both methods, the False Positive Rate (FPR) has significantly reduced for all three datasets while the Recall is improved, thus leading to an overall improvement on the F1-score. Comparing to the PCA-based method, we can observe the average F1-score over all three dataset is improved around 60% when both PCA-based and RSM-based methods are enabled in the LAD product pipeline.

## 5 CONCLUSION AND FUTURE WORK

Training data is not always available or sufficient, and identifying if the training data is contaminated, or not, may need a lot of human effort. It is important to introduce an online algorithm in the LAD which can provide predictive insights without off-line training data and learn a system’s normal behaviours gradually. In the meantime, it is vital for the LAD system to be smart enough to automatically identify incident periods and avoid potentially biased models. All such features are included by enabling the RSM-based log anomaly detection method in our current

product pipeline. Experiments on multiple datasets demonstrate the efficacy of the proposed method.

While we made significant progress in delivering a more robust and reliable LAD pipeline, the journey of continuous improvement of our LAD pipeline never stops. Here are a few things we are actively working on for future iterations.

- Enrich ensembled models for different LAD algorithms to work hand in hand to further enhance the accuracy of the model.
- Expose precision-recall tradeoff knobs to improve the transparency of LAD algorithms to gain trust of AI.
- Improve log comprehension so better representation can be learned from a mixture of formats.
- Seek and leverage user feedback to fine-tune individual LAD models and prediction aggregation.
- Customize our models to handle seasonality of log volumes and maintenance windows better.
- Differentiate anomalies, alerts and incidents to tell a meaningful incident story.
- Correlating alerts to golden signals, service level objectives, and error budgets for better separating alerts from incidents and for better incident prediction.

## REFERENCES

- Aggarwal, P., Bansal, S., Mohapatra, P., and Kumar, A. (2021). Mining domain-specific component-action links for technical support documents. In *8th ACM IKDD CODS and 26th COMAD, CODS COMAD 2021*, page 323–331, New York, NY, USA. Association for Computing Machinery.
- Aggarwal, P. and Nagar, S. (2021). Fault localization in cloud systems using golden signals. *Management*, 21:4.
- Baccianella, S., Esuli, A., and Sebastiani, F. (2010). SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings*

- of the *Seventh International Conference on Language Resources and Evaluation (LREC'10)*. European Language Resources Association (ELRA).
- Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3).
- Givental, G., Bhatia, A., and An, L. (2021a). Hybrid machine learning to detect anomalies. <https://patents.google.com/patent/US20210281592A1/>. Pub. No.: US 2021/0281592 A1.
- Givental, G., Bhatia, A., and An, L. (2021b). Modification of machine learning model ensembles based on user feedback. <https://patents.google.com/patent/US20210279644A1/>. Pub. No.: US 2021/0279644 A1.
- Goldberg, D. and Shan, Y. (2015). The importance of features for statistical anomaly detection. In *7th {USENIX} Workshop on Hot Topics in Cloud Computing (HotCloud 15)*.
- Gu, T., Dolan-Gavitt, B., and Garg, S. (2017). Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*.
- Hutto, C. and Gilbert, E. (2014). Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the International AAAI Conference on Web and Social Media*, pages 216–225.
- IBM (2022). IBM WebSphere Application Server. <https://www.ibm.com/cloud/websphere-application-server>.
- Krishnamurthy, R., Li, Y., Raghavan, S., Reiss, F., Vaithyanathan, S., and Zhu, H. (2009). Systemt: a system for declarative information extraction. *ACM SIGMOD Record*, 37(4):7–13.
- Lerner, A. (2017). AIOps platforms. <https://blogs.gartner.com/andrew-lerner/2017/08/09/aiops-platforms/>.
- Levin, A., Garion, S., Kolodner, E. K., Lorenz, D. H., Barabash, K., Kugler, M., and McShane, N. (2019). Aiops for a cloud object storage service. In *2019 IEEE International Congress on Big Data (BigDataCongress)*, pages 165–169.
- Liu, X., Tong, Y., Xu, A., and Akkiraju, R. (2020). Using language models to pre-train features for optimizing information technology operations management tasks. In *International Conference on Service-Oriented Computing*, pages 150–161. Springer.
- Liu, X., Tong, Y., Xu, A., and Akkiraju, R. (2021). Predicting information technology outages from heterogeneous logs. In *IEEE International Conference On Service-Oriented System Engineering*.
- Mohapatra, P., Deng, Y., Gupta, A., Dasgupta, G., Paradkar, A., Mahindru, R., Rosu, D., Tao, S., and Aggarwal, P. (2018). Domain knowledge driven key term extraction for it services. In Pahl, C., Vukovic, M., Yin, J., and Yu, Q., editors, *Service-Oriented Computing*, pages 489–504, Cham. Springer International Publishing.