DOI: https://doi.org/10.17721/ISTS.2020.1.50-59
УДК 004.8, 004.62, 004.93

COMPARATIVE ANALYSIS OF SYSTEM LOGS AND STREAMING DATA ANOMALY DETECTION ALGORITHMS

Andriy Lishchytovych [1]
orcid.org/0000-0002-3395-8616

Volodymyr Pavlenko [2]
orcid.org/0000-0002-3958-0415

Alexander Shmatok [3]
orcid.org/0000-0002-3351-2745

Yuriy Finenko [4]
orcid.org/0000-0003-1887-2475

[1] Kyiv, Ukraine, The Open International University of human development “Ukraine”, AL@sors.me
[2] Kyiv, Ukraine, The Open International University of human development “Ukraine”, Pavlenko.v@i.ua
[3] Kyiv, Ukraine, The Open International University of human development “Ukraine”, sh_al_st@ukr.net
[4] Kyiv, Ukraine, The Open International University of human development “Ukraine”, talaveryuriy@gmail.com

Історія статті:
Надійшла до редакції 21.07.2019
Прийнято 29.07.2019

Ключові слова:
виявлення аномалій;
системні журнали;
дерево прийняття рішень;
кластеризація;
аналіз даних;
ієрархічна часова пам’ять

Анотація: У цьому документі подано опис та порівняльний аналіз декількох загальноприйнятих підходів до аналізу системних журналів та потокових даних, що масово генеруються ІТ-інфраструктурою компанії, та виявленню аномалій. Важливість виявлення аномалії продиктована зростаючими витратами у випадку простою системи через події, які могли б бути передбачені на основі записів журналу з попереджувальними даними. Системи виявлення аномалій побудовані за допомогою стандартного процесу збору даних, аналізу, вилучення інформації та виявлення відхилень. Виявлення аномальної поведінки системи відіграє важливу роль у масштабних системах управління інцидентами. Своєчасне виявлення дозволяє ІТ-адміністраторам швидко виявити проблеми та негайно їх вирішити. Такий підхід значно скорочує час простою системи. Більшість ІТ-систем генерують журнали з детальною інформацією про операції. Тому журнали стають ідеальним джерелом даних рішень виявлення аномалії. Обсяг журналів унеможливлює їх аналіз вручну та вимагає автоматизованих підходів.Більша частина документа стосується кроку виявлення аномалії та таких алгоритмів, як регресія, дерево рішень, SVM, кластеризація, аналіз основних компонентів, видобуток інваріантів та ієрархічна модель тимчасової пам'яті. Алгоритми пошуку аномалії, що базуються на моделях, та ієрархічні алгоритми тимчасової пам'яті використовувались для обробки наборів даних HDFS, BGL та NAB з ~16 млн. повідомленнями журналу та ~365 тис. точками потокових даних. Дані були вручну позначені мітками, щоб дозволити навчання моделей та розрахунок точності їх роботи. Відповідно до результатів, системи контрольованого виявлення аномалій досягають високої точності, але потребують значних зусиль для тренувань моделей, тоді як алгоритм на основі HTM моделі показує найвищу точність виявлення при відсутності тренування.

Abstract: This paper provides with the description, comparative analysis of multiple commonly used approaches of the analysis of system logs, and streaming data massively generated by company IT infrastructure with an unattended anomaly detection feature. An importance of the anomaly detection is dictated by the growing costs of system downtime due to the events that would have been predicted based on the log entries with the abnormal data reported. Anomaly detection systems are built using standard workflow of the data collection, parsing, information extraction and detection steps. Most of the document is related to the anomaly detection step and algorithms like regression, decision tree, SVM, clustering, principal components analysis, invariants mining and hierarchical temporal memory model. Model-based anomaly algorithms and hierarchical temporary memory algorithms were used to process HDFS, BGL and NAB datasets with ~16m log messages and 365k data points of the streaming data. The data was manually labeled to enable the training of the models and accuracy calculation. According to the results, supervised anomaly detection systems achieve high precision but require significant training effort, while HTM-based algorithm shows the highest detection precision with zero training. Detection of the abnormal system behavior plays an important role in large-scale incident management systems. Timely detection allows IT administrators to quickly identify issues and resolve them immediately. This approach reduces the system downtime dramatically.Most of the IT systems generate logs with the detailed information of the operations. Therefore, the logs become an ideal data source of the anomaly detection solutions. The volume of the logs makes it impossible to analyze them manually and requires automated approaches.

1. INTRODUCTION

Modern enterprises utilize large scale networks, including hybrids where cloud and on-premise items are connected into the single mesh. These systems are in use as the core part of IT, supporting different services – e-commerce, social networks, search engines and knowledge bases. IT infrastructure is designed to work 24/7 serving huge number of users globally. Any issues with this system will break down company services and lead to significant revenue loss.

Detection of the abnormal system behavior plays an important role in large-scale incident management systems. Timely detection allows IT administrators to quickly identify issues and resolve them immediately. This approach reduces the system downtime dramatically.

Most of the IT systems generate logs with the detailed information of the operations. Therefore, the logs become an ideal data source of the anomaly detection solutions. The volume of the logs makes it impossible to analyze them manually and requires automated approaches.

This paper provides the review of multiple anomaly detection algorithms with the evaluation of their efficiency.

2. ANOMALY DETECTION WORKFLOW AND ALGORITHMS

Common log-based anomaly detection systems are built using the following workflow:

Collection step provides the ways to gather logs information and put it into the common storage. Outcome of the logs parsing is structured timestamped events stream. At this stage some additional information could be also extracted in case of stable logs structure. To use the machine learning models information has to be transformed into the numerical form so called feature vectors. Every event becomes the vector and all the vectors are event matrix. Having the matrix, machine learning algorithms could be used to identify patterns and detect any anomalies.

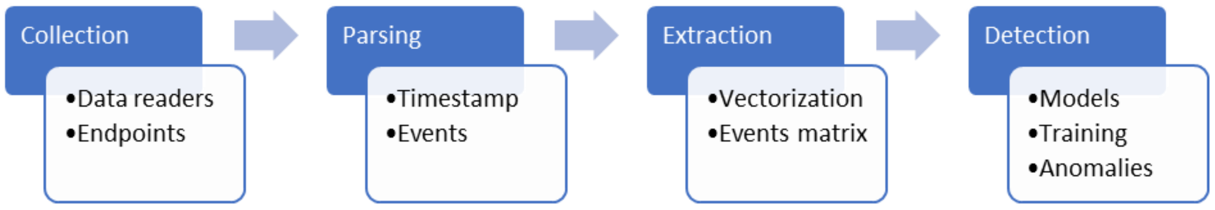


Fig. 1 - Anomaly detection system workflow

Let's define an anomaly as a point in time where the behavior of the system is unusual and significantly different from previous, normal behavior. An anomaly may signify a negative change in the system, like a fluctuation in the turbine rotation frequency of a jet engine, possibly indicating an imminent failure. An anomaly can also be positive, like an abnormally high number of web clicks on a new product page, implying stronger than normal demand. Either way, anomalies in data identify abnormal behavior with potentially useful information. Anomalies can be spatial, where an individual data instance can be considered anomalous with respect to the rest of data, independent of where it occurs in the data stream.

The description of the workflow steps and algorithms are described in [1].

3. LOG PARSING

Logs are plain text that consists of constant parts and variable parts, which may vary among different occurrences. For instance, given the logs of "Connection from 10.10.34.12 closed" and "Connection from 10.10.34.13 closed", the words "Connection", "from" and "closed" are considered as constant parts because they always stay the same, while the remaining parts are called variable parts as they are not fixed. Constant parts are predefined in source codes by developers, and variable parts are often generated dynamically (e.g., port number, IP address) that could not be well utilized in anomaly detection. The purpose of log parsing is to separate constant parts from variable parts and form a wellestablished log event (i.e., “Connection from * closed” in the example). There are two types of log parsing methods: clustering based (e.g., LKE [5], LogSig [15]) and heuristic-based (e.g., iPLoM [11], SLCT [16]). In clustering-based log parsers, distances between logs are calculated first, and clustering techniques are often employed to group logs into different clusters in the next step. Finally, event template is generated from each cluster. For heuristic-based approaches, the occurrences of each word on each log position are counted. Next, frequent words are selected and composed as the event candidates. Finally, some candidates are chosen to be the log events.

4. FEATURE EXTRACTION

The main purpose of this step is to extract valuable features from log events that could be fed into anomaly detection models. The input of feature extraction is log events generated in the log parsing step, and the output is an event count matrix. In order to extract features, we firstly need to separate log data into various groups, where each group represents a log sequence. To do so, windowing is applied to divide a log dataset into finite chunks [3].

Fixed window. Both fixed windows and sliding windows are based on timestamp, which records the occurrence time of each log. Each fixed window has its size, which means the time span or time duration (the window size is Δt, which is a constant value, such as one hour or one day). Thus, the number of fixed windows depends on the predefined window size. Logs that happened in the same window are regarded as a log sequence.

Sliding window. Different from fixed windows, sliding windows consist of two attributes: window size and step size, e.g., hourly windows sliding every five minutes. In general, step size is smaller than window size, therefore causing the overlap of different windows. The number of sliding windows, which is often larger than fixed windows, mainly depends on both window size and step size. Logs that occurred in the same sliding window are also grouped as a log sequence, though logs may duplicate in multiple sliding windows due to the overlap.

Session window. Compared with the above two windowing types, session windows are based on identifiers instead of the timestamp. Identifiers are utilized to mark different execution paths in some log data. For instance, HDFS logs with block_id record the allocation, writing, replication, deletion of certain block. Thus, we can group logs according to the identifiers, where each session window has a unique identifier. After constructing the log sequences with windowing techniques, an event count matrix X is generated. In each log sequence, we count the occurrence number of each log event to form the event count vector. For example, if the event count vector is [0, 0, 2, 3, 0, 1, 0], it means that event 3 occurred twice and event 4 occurred three times in this log sequence. Finally, plenty of event count vectors are constructed to be an event count matrix X, where entry X(i,j) records how many times the event j occurred in the i-th log sequence.

5. SUPERVISED ANOMALY DETECTION

Supervised learning (e.g., decision tree) is defined as a machine learning task of deriving a model from labeled training data. Labeled training data, which indicate normal or anomalous state by labels, are the prerequisite of supervised anomaly detection. The more labeled the training data, the more precise the model would be. We will introduce three representative supervised methods: logistic regression, decision tree, and support vector machine (SVM) in the following.

6. LOGISTIC REGRESSION

Logistic regression is a statistical model that has been widely-used for classification. To decide the state of an instance, logistic regression estimates the probability p of all possible states (normal or anomalous). The probability p is calculated by a logistic function, which is built on labeled training data. When a new instance appears, the logistic function could compute the probability p, p∈(0,1) of all possible states. After obtaining the probabilities, the states with the largest probability is the classification output. To detect anomalies, an event count vector is constructed from each log sequence, and every event count vector together with its label are called an instance. Firstly, we use training instances to establish the logistic regression model, which is actually a logistic function. After obtaining the model, we feed a testing instance X into the logistic function to compute its possibility p of anomaly, the label of X is anomalous when p≥0.5 and normal otherwise.

7. DECISION TREE

Decision Tree is a tree structure diagram that uses branches to illustrate the predicted state for each instance. The decision tree is constructed in a top-down manner using training data. Each tree node

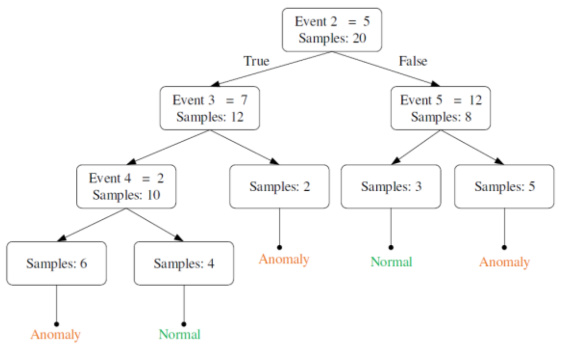


Fig. 2 -Decision Tree

is created using the current “best” attribute, which is selected by attribute’s information gain [6]. For example, the root node in Figure 2 shows that there are totally 20 instances in our dataset. When splitting the root node, the occurrence number of Event 2 is treated as the “best” attribute. Thus, the entire 20 training instances are split into two subsets according to the value of this attribute, in which one contains 12 instances and the other consists of 8 instances.

Decision Tree was first applied to failure diagnosis for web request log system in [4]. The event count vectors together with their labels described in Section III-B are utilized to build the decision tree. To detect the state of a new instance, it traverses the decision tree according to the predicates of each traversed tree node. In the end of traverse, the instance will arrive one of the leaves, which reflects the state of this instance.

8. SUPPORT VECTOR MACHINE

Support Vector Machine (SVM) is a supervised learning method for classification. In SVM, a hyperplane is constructed to separate various classes of instances in high-dimension space. Finding the hyperplane is an optimization problem, which maximizes the distance between the hyperplane and the nearest data point in different classes.

In [8], Liang et al. employ SVM to detect failures and compared it with other methods. Similar to Logistic Regression and Decision Tree, the training instances are event count vectors together with their labels. In anomaly detection via SVM, if a new instance is located above the hyperplane, it would be reported as an anomaly, while marked as normal otherwise. In this paper, we only discuss linear SVM.

9. UNSUPERVISED ANOMALY DETECTION

Unlike supervised methods, unsupervised learning is another common machine learning task but its training data is unlabeled. Unsupervised methods are more applicable in real-world production environment due to the lack of labels. Common unsupervised approaches include various clustering methods, association rule mining, PCA and etc.

10. LOG CLUSTERING

In [9], Lin et al. design a clustering-based method called Log Cluster to identify online system problems. Log Cluster requires two training phases, namely knowledge base initialization phase and online learning phase. Thus, the training instances are divided into two parts for these two phases, respectively.

Knowledge base initialization phase contains three steps: log vectorization, log clustering, representative vectors extraction. Firstly, log sequences are vectorized as event count vectors, which are further revised by Inverse Document. Frequency (IDF) [14] and normalization. Secondly, Log Cluster clusters normal and abnormal event count vectors separately with agglomerative hierarchical clustering, which generates two sets of vector clusters (i.e., normal clusters and abnormal clusters) as knowledge base. Finally, we select a representative vector for each cluster by computing its centroid.

Online learning phase is used to further adjust the clusters constructed in knowledge base initialization phase. In online learning phase, event count vectors are added into the knowledge base one by one. Given an event count vector, the distances between

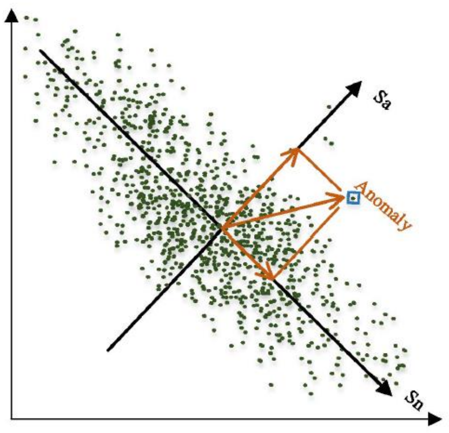


Fig. 3 -Log Cluster anomaly detection

it and existing representative vectors are computed. If the smallest distance is less than a threshold, this event count vector will be added to the nearest cluster and the representative vector of this cluster will be updated. Otherwise, Log Cluster creates a new cluster using this event count vector. After constructing the knowledge base and complete the online learning process, Log Cluster can be employed to detect anomalies. Specifically, to determine the state of a new log sequence, we compute its distance to representative vectors in knowledge base. If the smallest distance is larger than a threshold, the log sequence is reported as an anomaly. Otherwise, if the nearest cluster is a normal/an abnormal cluster, the log sequence is reported as normal/abnormal.

11. PRINCIPAL COMPONENT ANALYSIS

Principal Component Analysis (PCA) is a statistical method that has been widely used to conduct dimension reduction. The basic idea behind PCA is to project high-dimension data (e.g., high-dimension points) to a new coordinate system composed of k principal components (i.e., k dimensions), where k is set to be less than the original dimension. PCA calculates the k principal components by finding components (i.e., axes) which catch the most variance among the high-dimension data. Thus, the PCA-transformed low-dimension data can preserve the major characteristics (e.g., the similarity between two points) of the original high-dimension data. For example, in Figure 3, PCA attempts to transform two-dimension points to the one-dimension points. S_n is selected as the principal component because the distance between points can be best described by mapping them to S_n

where n (∗) represents the number of logs which belong to corresponding event type ∗. Intuitively, Invariants mining could uncover the linear relationships between multiple log events that represent system normal execution behaviors. Linear relationships prevail in real-world system events. For example, normally, a file must be closed after it was opened. Thus, log with phrase "open file" and log with phrase "close file" would appear in pair. If the number of log events "open file" and that of "close file" in an instance are not equal, it will be marked abnormal because it violates the linear relationship. Invariants mining, which aims at finding invariants (i.e., linear relationships), contains three steps. The input of invariants mining is an

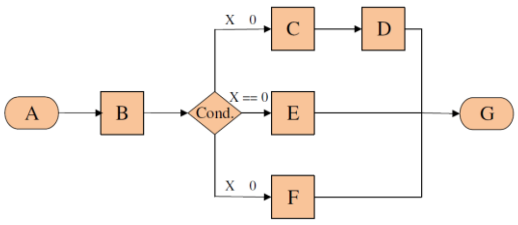


Fig. 4 - Principal Component Analysis

PCA was first applied in log-based anomaly detection [17]. In their anomaly detection method, each log sequence is vectorized as an event count vector. After that, PCA is employed to find patterns between the dimensions of event count vectors. Employing PCA, two subspaces are generated, namely normal space Sn and anomaly space Sa. Sn is constructed by the first k principal components and Sn is constructed by the remaining $(n-k)$, where n is the original dimension. Then, the projection $y_a =(1-PP^T)y$ of an event count vector y to S_a is calculated, where $p=[v_1, v_2, v_3 ,..., v_n]$ is the first k principal components. If the length of y_a is larger than a threshold, the corresponding event count vector will be reported as an anomaly. For example, the selected point in Figure 3 is an anomaly because the length of its projection on S_a is too large

12. INVARIANTS MINING

Program Invariants are the linear relationships that always hold during system running even with various inputs and under different workloads. Invariants mining was first applied to log-based anomaly detection in [10]. Logs that have the same session id (e.g., *block_id* in HDFS) often represent the program execution flow of that session.

In this execution flow, the system generates a log message at each stage from A to G. Assuming that there are plenty of instances running in the system and they follow the program execution flow in Figure 4, the following equations would be valid:

event count matrix generated from log sequences, where each row is an event count vector. Firstly, the invariant space is estimated using singular value decomposition, which determines the amount r of invariants that need to be mined in the next step. Secondly, this method finds out the invariants by a brute force search algorithm. Finally, each mined invariant candidate is validated by comparing its support with a threshold (e.g., supported by 98% of the event count vectors). This step will continue until r independent invariants are obtained. In anomaly detection based on invariants, when a new log sequence arrives, we check whether it obey the invariants. The log sequence will be reported as an anomaly if at least one invariant is broken.

13. TOOL IMPLEMENTATION

[1] implemented six anomaly detection methods in Python with over 4,000 lines of code and packaged them as a toolkit. For supervised methods, [1] utilizes a widely-used machine learning package, scikit-learn [13], to implement the learning models of Logistic Regression, Decision Tree, and SVM.

There are plenty of parameters in SVM and logistic regression, and we manually tune these parameters to achieve the best results during training. For SVM, we tried different kernels and related parameters one by one, and we found that SVM with linear kernel obtains the better anomaly detection accuracy than other kernels. For logistic regression, different parameters are also explored, and they are carefully tuned to achieve the best performance.

Implementing unsupervised methods, however, is not straightforward. For log clustering, [1] were not able to directly use the clustering API from scikit-learn, because it is not designed for large-scale datasets, where our data cannot fit to the memory. We implemented the clustering algorithm into an online version, whereby each data instance is grouped into a cluster one by one. There are multiple thresholds to be tuned. We also paid great efforts to implement the invariants mining method, because we built a search space for possible invariants and proposed multiple ways to prune all unnecessary invariants. It is very time-consuming to test different combination of thresholds. [1] finally implemented PCA method according to the original reference based on the use of an API from scikit-learn. PCA has only two parameters and it is easy to tune.

14. SUPERVISED METHODS EXPERIMENTAL DATASETS

Log Datasets. Publicly available production logs are scarce data because companies rarely publish them due to confidential issues. Fortunately, by exploring an abundance of literature and intensively contacting the corresponding authors, [1] has successfully obtained two log datasets, HDFS data [17] and BGL data [12], which are suitable for evaluating existing anomaly detection methods. Both datasets are collected from production systems, with a total of 15,923,592 log messages and 365,298 anomaly samples, that are manually labeled by the original domain experts. Thus, [1] took these labels (anomaly or not) as the ground truth for accuracy evaluation purposes. More statistical information of the datasets is provided in Table I. HDFS data contain 11,175,629 log messages, which were collected from Amazon EC2 platform [17]. HDFS logs record a unique block ID for each block operation such as allocation, writing, replication, deletion. Thus, the operations in logs can be more naturally captured by session windows, as introduced in III-B, because each unique block ID can be utilized to slice the logs into a set of log sequences. Then we extract feature vectors from these log sequences and generate 575,061 event count vectors. Among them, 16,838 samples are marked as anomalies.

BGL data contain 4,747,963 log messages, which were recorded by the Blue Gene supercomputer system at Lawrence Livermore National Labs (LLNL) [12]. Unlike HDFS data, BGL logs have no identifier recorded for each job execution. Thus, we have to use fixed windows or sliding windows to slice logs as log sequences, and then extract the corresponding event count vectors. But the number of windows depends on the chosen window size (and step size). In BGL data, 348,460 log messages are labeled as failures, and a log sequence is marked as an anomaly if any failure logs exist in that sequence

15. SUPERVISED METHODS ACCURACY RESULTS

To explore the accuracy of supervised methods, we use them to detect anomalies on HDFS data and BGL data. [1] uses session windows to slice HDFS data and then generate the event count matrix, while fixed windows and sliding windows are applied to BGL data separately. In order to check the validity of three supervised methods (namely Logistic Regression, Decision Tree, SVM), we first train the models on training data, and then apply them to testing data. [1] reports both training accuracy and testing accuracy in different settings, as illustrated in Tables 1 - 4. We can observe that all supervised methods achieve high training accuracy (over 0.95), which implies that normal instances and abnormal instances are well separated by using our feature representation. However, their accuracy on testing data varies with different methods and datasets. The overall accuracy on HDFS data is higher than the accuracy on BGL data with both fixed windows and sliding windows. This is mainly because HDFS system records relatively simple operations with only 29 event types, which is much less than that in BGL data, which is 385. Besides, HDFS data are grouped by session windows, thereby causing a higher correlation between events in each log sequence. Therefore, anomaly detection methods on HDFS perform better than on BGL.

16. USING HTM TO DETECT THE ANOMALIES

There are a number of other restrictions that can make methods unsuitable for real-time streaming anomaly detection, such as computational constraints that impede scalability. An example is Lytics Anomalyzer [25], which runs in O(n2), limiting its usefulness in practice where streams are arbitrarily long. Dimensionality is another factor that can make some methods restrictive. For instance, online variants of principle component analysis (PCA) such as osPCA [26] or window-based PCA [27] can only work with high-dimensional, multivariate data streams that can be projected onto a low dimensional space. Techniques that require data labels, such as supervised classification-based methods [28], are typically unsuitable for real-time anomaly detection and continuous learning. On the other hand, Hierarchical Temporal Memory models are best suited for the streaming data by the nature of its architecture [18].

Table 1. Supervised HDFS data set with session windows

	Supervised HDFS with session windows					
	Training			Testing		
	Precision	Recall	F-measure	Precision	Recall	F-measure
Logistic	0.96	1	0.98	0.95	1	0.98
Decision Tree	1	1	1	1	0.99	1
SVM	0.96	1	0.98	0.95	1	0.98

Table 2. Supervised BGL data set with fixed windows

	Supervised BGL with fixed windows					
	Training			Testing		
	Precision	Recall	F-measure	Precision	Recall	F-measure
Logistic	0.99	1	1	0.95	0.57	0.71
Decision Tree	1	1	1	0.95	0.57	0.72
SVM	1	1	1	0.95	0.57	0.71

Table 3. Supervised BGL data set with sliding windows

	Supervised BGL with sliding windows					
	Training			Testing		
	Precision	Recall	F-measure	Precision	Recall	F-measure
Logistic	0.99	0.81	0.89	1	0.7	0.82
Decision Tree	1	1	1	0.92	0.63	0.74
SVM	1	1	1	0.99	0.75	0.85

Table 4. Unsupervised HDF & BGL data sets

	Unsupervised models on HDFS & BGL					
	HDFS			BGL		
	Precision	Recall	F-measure	Precision	Recall	F-measure
Log Clustering	0.87	0.74	0.8	0.42	0.87	0.57
Invariant Mining	0.88	0.95	0.91	0.83	0.99	0.91
PCA	0.98	0.67	0.79	0.5	0.61	0.55

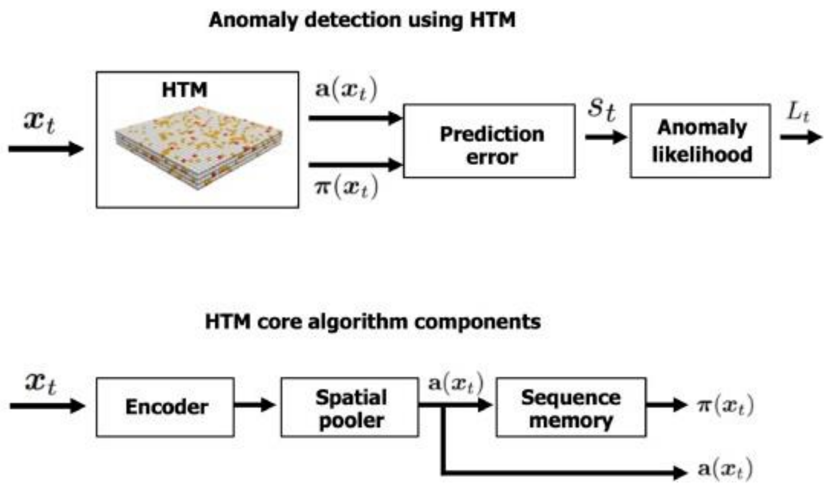


Fig. 5 - HTM-based anomaly detection workflow

Model-based approaches have been developed for specific use cases, but require explicit domain knowledge and are not generalizable. Domain-specific examples include anomaly detection in aircraft engine measurements [19], cloud datacenter temperatures [20], and ATM fraud detection [21]. Kalman filtering is a common technique, but the parameter tuning often requires domain knowledge and choosing specific residual error models [22 - 24]. Model-based approaches are often computationally efficient but their lack of generalizability limits their applicability to general streaming applications.

17. HTM BENCHMARK DATASET

The aim of the dataset is to present algorithms with the challenges they will face in real-world scenarios, such as a mix of spatial and temporal anomalies, clean and noisy data, and data streams where the statistics evolve over time. The best way to do this is to provide data streams from real-world use cases, and from a variety of domains and applications. The data currently in the dataset represents a variety of sources, ranging from server network utilization to temperature sensors on industrial machines to social media chatter.

Dataset (NAB) version 1.0 contains 58 data streams, each with 1000–22,000 records, for a total of 365,551 data points [29]. Also included are some artificially-generated data files that test anomalous behaviors not yet represented in the corpus's real data, as well as several data files without any anomalies. All data files are labeled, either because we know the root cause for the anomalies from the provider of the data, or as a result of the well-defined NAB labeling procedure. These labels define the ground truth anomalies used in the NAB scoring process.

Traditional scoring methods, such as precision and recall, don't suffice because they don't effectively test anomaly detection algorithms for real-time use. For example, they do not incorporate time and do not reward early detection. Therefore, the standard classification metrics – true positive (TP), false positive (FP), true negative (TN), and false negative (FN) are not applicable for evaluating algorithms for the above requirements.

NAB includes three different application profiles: standard, reward low FPs, and reward low FNs. The standard profile assigns TPs, FPs, and FNs with relative weights (tied to the window size) such that random detections made 10% of the time would get a zero-final score on average. The latter two profiles accredit greater penalties for FPs and FNs, respectively. These two profiles are somewhat arbitrary but designed to be illustrative of algorithm behavior. The NAB codebase itself is designed such that the user can easily tweak the relative weights and re-score all algorithms. The application profiles thus help evaluate the sensitivity of detectors to specific applications criteria. The combination of anomaly windows, a smooth temporal scoring function (details in the next section), and the introduction of application profiles allows researchers to evaluate online anomaly detector implementations against the requirements of the ideal detector. Specifically, the overall NAB scoring system evaluates real-time performance, prefers earlier detection of anomalies, penalizes "spam" (i.e. FPs), and provides realistic costs for the standard classification evaluation metrics TP, FP, TN, and FN.

Table 5 Streaming data anomaly detection results

Detector	**Standard Profile**	**Reward Low FP**	**Reward Low FN**
Perfect	*100*	*100*	*100*
Numenta HTM	70.5-69.7	62.6-61.7	75.2-74.2
CAD OSE	69.9	67	73.2
earthgecko Skyline	58.2	46.2	63.9
KNN CAD	58	43.4	64.8
Relative Entropy	54.6	47.6	58.8
Random Cut Forest	51.7	38.4	59.7
Twitter ADVec v1.0.0	47.1	33.6	53.5
Windowed Gaussian	39.6	20.9	47.4
Etsy Skyline	35.7	27.1	44.5
Bayesian Changepoint	17.7	3.2	32.2
EXPoSE	16.4	3.2	26.9
Random	11	1.2	19.5
Null	*0*	*0*	*0*

18. UNSUPERVISED METHODS ACCURACY RESULTS

The scores are normalized such that the maximum possible is 100.0 (i.e. the perfect detector), and a baseline of 0.0 is determined by the "null" detector (which makes no detections) [29].

19. CONCLUSION

Decision tree is more interpretable than the other two methods, as developers can detect anomalies with meaningful explanations (i.e., predicates in tree nodes). Logistic regression cannot solve linearly non-separable problems, which can be solved by SVM using kernels. However, parameters of SVM are hard to tune (e.g., penalty parameter), so it often requires much manual effort to establish a model. Unsupervised methods are more practical and meaningful due to the lack of labels. Log clustering uses the idea of online learning. Therefore, it is suitable for processing large volume of log data. Invariants mining not only can detect anomalies with a high accuracy, but also can provide meaningful and intuitive interpretation for each detected anomaly. However, the invariants mining process is time consuming. PCA is not easy to understand and is sensitive to the data. Thus, its anomaly detection accuracy varies over different datasets.

The following findings have been made:

- Supervised anomaly detection methods achieve high precision, while the recall varies over different datasets and window settings.
- Anomaly detection with sliding windows can achieve higher accuracy than that of fixed windows.
- Unsupervised methods generally achieve inferior performance against supervised methods. But invariants mining manifests as a promising method with stable, high performance.
- The settings of window size and step size have different effects on supervised methods and unsupervised methods.
- Most anomaly detection methods scale linearly with log size, but the methods of Log Clustering and Invariants Mining need further optimizations for speedup.
- Anomalies detection in streaming, real-time applications generally better to handle using

20. REFERENCES

[1] Shilin He, Jieming Zhu, Pinjia He, and Michael R. Lyu, Experience Report: System Log Analysis for Anomaly Detection, 2016 IEEE 27th International Symposium on Software Reliability Engineering

[2] Subutai Ahmad, Alexander Lavin, Scott Purdy, Zuha Agha, Unsupervised real-time anomaly detection for streaming data, Neurocomputing, Volume 262, 1 November 2017, Pages 134-147

[3] T. Akidau, R. Bradshaw, C. Chambers, S. Chernyak, R. J. FernÃ¡ndez- Moctezuma, R. Lax, S. McVeety, D. Mills, F. Perry, E. Schmidt, and S. Whittle. The dataflow model: a practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-oforder data processing. In PVLDB'15: Proc. of the VLDB Endowment, volume 8, pages 1792–1803. VLDB Endowment, 2015.

[4] M. Chen, A. X. Zheng, J. Lloyd, M. I. Jordan, and E. Brewer. Failure diagnosis using decision trees. In ICAC'04: Proc. of the 1st International Conference on Autonomic Computing, pages 36–43. IEEE, 2004.

[5] Q. Fu, J. Lou, Y. Wang, and J. Li. Execution anomaly detection in distributed systems through unstructured log analysis. In ICDM'09: Proc. of International Conference on Data Mining, 2009.

[6] J. Han, M. Kamber, and J. Pei. Data mining: concepts and techniques. Elsevier, 2011.

[7] P. He, J. Zhu, S. He, J. Li, and R. Lyu. An evaluation study on log parsing and its use in log mining. In DSN'16: Proc. of the 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 2016.

[8] Y. Liang, Y. Zhang, H. Xiong, and R. Sahoo. Failure prediction in ibm bluegene/l event logs. In ICDM'07: Proc. of the 7th International Conference on Data Mining, 2007.

[9] Q. Lin, H. Zhang, J.G. Lou, Y. Zhang, and X. Chen. Log clustering based problem identification for online service systems. In ICSE'16: Proc. of the 38th International Conference on Software Engineering, 2016.

[10] J. Lou, Q. Fu, S. Yang, Y Xu, and J. Li. Mining invariants from console logs for system problem detection. In ATC'10: Proc. of the USENIX Annual Technical Conference, 2010.

[11] A. Makanju, A. Zincir-Heywood, and E. Milios. Clustering event logs using iterative partitioning. In KDD'09: Proc. of International Conference on Knowledge Discovery and Data Mining, 2009.

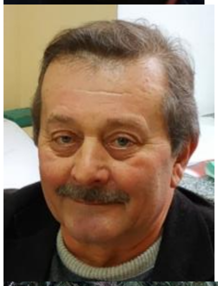
[12] A. Oliner and J. Stearley. What supercomputers say: A study of five system logs. In DSN'07:Proc. of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 2007.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, et al. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830, 2011.

[14] G. Salton and C Buckley. Term weighting approaches in automatic text retrival. Technical report, Cornell, 1987.
[15] L. Tang, T. Li, and C. Perng. LogSig: generating system events from raw textual logs. In CIKM’11: Proc. of ACM International Conference on Information and Knowledge Management, pages 785–794, 2011.
[16] R. Vaarandi. A data clustering algorithm for mining patterns from event logs. In IPOM’03: Proc. of the 3rd Workshop on IP Operations and Management, 2003.
[17] W. Xu, L. Huang, A. Fox, D. Patterson, and M.I. Jordon. Detecting large-scale system problems by mining console logs. In SOSP’09: Proc. of the ACM Symposium on Operating Systems Principles, 2009.
[18] Yuwei Cui, Subutai Ahmad, Jeff Hawkins The HTM Spatial Pooler—A Neocortical Algorithm for Online Sparse Distributed Coding, Front. Comput. Neurosci., 29 November 2017, https://doi.org/10.3389/fncom.2017.00111
[19] D.L. Simon, A.W. Rinehart A model-based anomaly detection approach for analyzing streaming aircraft engine measurement data Proceedings of Turbo Expo 2014: Turbine Technical Conference and Exposition, ASME (2014), pp. 665-672, 10.1115/GT2014-27172
[20] Lee E.K., H. Viswanathan, D. Pompili Model-based thermal anomaly detection in cloud datacenters Proceedings of the IEEE International Conference on Distributed Computing in Sensor Systems (2013), pp. 191-198, 10.1109/DCOSS.2013.8
[21] T. Klerx, M. Anderka, H.K. Buning, S. Priesterjahn Model-based anomaly detection for discrete event systems Proceedings of the 2014 IEEE 26th International Conference on Tools with Artificial Intelligence, IEEE (2014), pp. 665-672, 10.1109/ICTAI.2014.105
[22] F. Knorn, D.J. Leith Adaptive Kalman filtering for anomaly detection in software appliances Proceedings of the IEEE INFOCOM (2008), 10.1109/INFOCOM.2008.4544581
[23] A. Soule, K. Salamatian, N. Taft Combining filtering and statistical methods for anomaly detection Proceedings of the 5th ACM SIGCOMM conference on Internet measurement, 4 (2005), p. 1, 10.1145/1330107.1330147
[24] Lee H., S.J. Roberts On-line novelty detection using the Kalman filter and extreme value theory Proceedings of the 19th International Conference on Pattern Recognition, (2008), pp. 1-4, 10.1109/ICPR.2008.4761918
[25] A. Morgan, Lytics Anomalyzer Blog, (2015). https://www.getlytics.com/blog/post/check_out _anomalyzer.
[26] Lee Y.J., Y.R. Yeh, Wang Y.C.F. Anomaly detection via online oversampling principal component analysis IEEE Trans. Knowl. Data Eng, 25 (2013), pp. 1460-1470, 10.1109/TKDE.2012.99
[27] A. Lakhina, M. Crovella, C. Diot Diagnosing network-wide traffic anomalies ACM SIGCOMM Comput. Commun. Rev, 34 (2004), p. 219, 10.1145/1030194.1015492
[28] N. Görnitz, M. Kloft, K. Rieck, U. Brefeld Toward supervised anomaly detection J. Artif. Intell. Res, 46 (2013), pp. 235-262, 10.1613/jair.3623
[29] The Numenta Anomaly Benchmark, https://github.com/numenta/NAB (accessed 2020-02-09)

Ліщитович Андрій Леонідович *– Medical Brain – темничній директор. Сфера наукових інтересів – інформаційні технології.*

Павленко Володимир Іванович *- професор кафедри Комп’ютерної інженерії. Сфера наукових інтересів – інформаційні технології.*



Шматок Олександр Станіславович – *директор інституту Комп’ютерних технологій університету «Україна». Сфера наукових інтересів – інформаційні технології.*

Фіненко Юрій Іванович – *інженер-дослідник лабораторії кіберфізичних систем університету «Україна». Сфера наукових інтересів – машинне навчання, захист інформації, програмування ПЛІС.*