Demystifying Numenta Anomaly Benchmark

Nidhi Singh
Senior Researcher
Intel Security, Germany
Email: nidhi.singh@intel.com

Craig Olinsky
Senior Director
Intel Security, Germany
Email: craig.olinsky@intel.com

***Abstract*—Detecting anomalies in large-scale, streaming datasets has wide applicability in a myriad of domains like network intrusion detection for cyber-security, fraud detection for credit cards, system health monitoring, and fault detection in safety critical systems. Due to its wide applicability, the problem of anomaly detection has been well-studied by industry and academia alike, and many algorithms have been proposed for detecting anomalies in different problem settings. But until recently, there was no openly available, systematic dataset and/or framework using which the proposed anomaly detection algorithms could be compared and evaluated on a common ground. Numenta Anomaly Benchmark (NAB), made available by Numenta[1] in 2015, addressed this gap by providing a set of openly-available, labeled data files and a common scoring system, using which different anomaly detection algorithms could be fairly evaluated and compared. In this paper, we provide an in-depth analysis of the key aspects of the NAB framework, and highlight inherent challenges therein, with the objective to provide insights about the gaps in the current framework that must be addressed so as to make it more robust and easy-to-use. Furthermore, we also provide additional evaluation of five state-of-the-art anomaly detection algorithms (including the ones proposed by Numenta) using the NAB datasets, and based on the evaluation results, we argue that the performance of these algorithms is not sufficient for practical, industry-scale applications, and must be improved upon so as to make them suitable for large-scale anomaly detection problems.**

I. Introduction

Anomaly detection addresses the problem of identifying points and/or patterns that do not conform to an expected sense of normality in a dataset. This problem has wide applications in a myriad of domains like, fraud detection for credit cards, system health monitoring, event detection in sensor networks, network intrusion detection for cyber-security, and fault detection in safety critical systems[1–6]. Due to its wide applicability, the problem of anomaly detection has been studied in detail, and various methods have been proposed based on the forms of input data (e.g., categorical or continuous, labeled or unlabeled) and types of anomalies (e.g., point, contextual or collective) that need to be identified in the given dataset[7–13]. But until recently, there was a lack of an openly-available dataset and/or framework using which various anomaly detection methods could be fairly evaluated on common ground. This gap has been recently addressed by Numenta Anomaly Benchmark[14] which provides an open source framework and labeled datasets for comparison of anomaly detection methods. This was indeed an important contribution to open research community in the domain of anomaly detection.

[1]http://numenta.org/

The Numenta Anomaly Benchmark (NAB), has two main components: a scoring system and labeled datasets. The NAB scoring system is essentially focused on real-life, streaming environments i.e., those real world settings where there exists a continuous stream of data points and the problem involves identification of anomalous data points in such continuous stream of data. The NAB dataset is composed of 58 labeled data files which span multiple domains like Twitter traffic, CPU utilization, New York taxi demand and temperature failure systems. These files are manually labeled following a well-documented procedure in Numenta, and hence contain reliable ground truth that can be used for robust evaluation of different anomaly detection algorithms.

In this work, we provide an in-depth analysis of the key aspects of the NAB framework, i.e., its scoring system and the labeled datasets. In doing so, we highlight the challenges inherent in the NAB framework that must be addressed so as to make it more robust, thereby increasing its applicability in real-world problem scenarios. Concretely, we identify the gaps in the NAB scoring system and we argue that such gaps hinder the applicability of the NAB scores in industry-scale problem settings. In addition, we also illustrate the challenges that are faced in the NAB datasets (e.g., missing values and difference in data distribution) which makes it difficult to fit a conventional time series based or machine learning based model to these datasets. This kind of in-depth analysis of the underpinnings of the NAB framework, including the scoring system as well as the datasets, lays out a path to further improvements in the NAB framework which would, in turn, benefit the open research community by means of more robust and easy-to-use framework for comparison of anomaly detection methods.

Another noteworthy contribution of the NAB framework is the comparative evaluation results of five state-of-the-art anomaly detection algorithms, namely Contextual Anomaly Detector(ContextOSE)[15], two variations of Numenta's Hierarchical Temporal Memory (HTM)[14], Twitter's AnomalyDetectionVec (ADVec)[16] and Etsy's Skyline[17], on all 58 NAB datasets. These algorithms are evaluated and compared using the NAB scoring system, and the NAB score of each algorithm for each dataset is made openly available as part of the framework. Since our deeper analysis of the NAB framework reveals important gaps in the NAB scoring system, we perform additional evaluation of these state-of-the-art algorithms using standard metrics like precision and recall and we show that the performance of these algorithms is not sufficient for industry-scale applications, and must be improved upon so as to make them suitable for large-scale anomaly detection problems.

To summarize, the main contributions of this work are as follows:

1) We provide in-depth analysis of the NAB scoring system and identify important gaps therein.
2) We illustrate the challenges that are faced in handling NAB datasets, which hinders the predictive performance of conventional time-series based or machine learning based models on these datasets.
3) We also perform comprehensive evaluation of five state-of-the-art anomaly detection algorithms (that are specified in the NAB framework) using standard metrics like precision, recall and false positive rate, which reveals another perspective of the predictive performance of these algorithms than is shown using the NAB scoring system in the framework.

The rest of the paper is organized as follows. In Section II, we briefly survey state-of-the-art literature in the domain of anomaly detection. In Section III, we provide an overview of the important aspects of the holistic NAB framework. Thereafter, we describe the key challenges in the NAB scoring system and in NAB datasets in Section IV, and additional evaluation results of five state-of-the-art anomaly detection methods in Section V. We conclude our work in Section VI.

II. Related Work

Anomaly detection methods have been developed for a variety of domains like network intrusion detection [6], fraud detection [3], [4], host based intrusion detection [1], [5] and outlier detection in sensor data [2]. Different types of data (e.g., text, images or videos) require different anomaly detection methods, some of which can be found in [18–20]. These methods typically pertain to a particular application domain and it is not straightforward to adapt these methods to other domains. But, certain methods deal with the problem of anomaly detection in generality, i.e., without limiting to a particular domain. For instance, Kloft and Laskov [7] proposed method for online anomaly detection in the presence of an adversary, Gornitz et al. [8] described method for supervised anomaly detection (i.e., when labeled anomalies are present in the datasets) while Blanchard et al. [13] described method for semi-supervised novelty detection. Other important aspects of anomaly detection like automatic selection of relevant features have also been studied and explored in detail[12]. A comprehensive survey of such anomaly detection methods can be found in [21].

In this work, we focus on the Numenta Anomaly Benchmark [14], [22] which provides a common scoring system and a set of 58 labeled data files gathered from multiple domains that can be used for comparison and evaluation of different anomaly detection methods. In this framework, the following anomaly detection methods have been so far evaluated and the results have been made publicly available: ContextOSE[15], two variations of Numenta's Hierarchical Temporal Memory(HTM) algorithm[14], Twitter's ADVec[16], and Etsy's Skyline [17]. Each of these algorithms deals with detection of anomalies in streaming environments, and have been shown to work well in terms of NAB scoring system. But in this work, apart from analyzing the NAB framework, we provide additional evaluation results for each of these algorithms in terms of standard metrics like precision, recall and false positive rate, and we argue that the performance of these algorithms is not sufficient for general, large-scale industry applications.

III. Numenta Anomaly Benchmark- An Overview

Numenta Anomaly Benchmark (NAB) provides an open source framework for comparison of anomaly detection algorithms in a variety of real-life applications, especially the ones that involve streaming, time-series data (e.g., monitoring of CPU and network utilization in a data center, social media activity on Twitter, and the like). It is composed of 58 files, each with 1,000-22,000 instances, for a total of 365,551 data points. Each of these files is manually labeled following a well-defined, documented procedure in Numenta. The holistic NAB dataset span across multiple domains and applications like twitter traffic, New York taxi demand, and temperature failure system of machines. Such diverse dataset indeed enables robust evaluation and comparison of anomaly detection algorithms.

The NAB datasets are primarily composed of two main attributes, i.e., timestamp and the true value of an observation. In addition, the benchmark provides (in a separate set of files) true anomaly labels for each observation in a dataset which indicates whether the observation is anomalous or not. Another noteworthy contribution of the NAB framework is the comparative results of five state-of-the-art anomaly detection algorithms, namely ContextOSE [15], two variations of Numenta's Hierarchical Temporal Memory (HTM) [14], Twitter's AnomalyDetectionVec [16] and Etsy's Skyline [17], on all 58 datasets. Apart from these algorithms, comparative results of other (simple) baselines like null detector (which outputs a constant anomaly score of 0.5 for all instances) and random detector (which outputs a random anomaly score between 0 and 1), are also made available. But in this work, we focus on the five aforementioned anomaly detection methods.

There are two types of metrics included in the comparative evaluation results. First, the standard metrics like true positives, false positives, true negatives and false negatives. Second, *NAB anomaly score*[14] which is designed specifically to reward early detections of anomalies in a given dataset. The NAB anomaly score, in turn, has the following three components:

1) *Anomaly window* - An anomaly window consists of a sequence of data points centered around one or more true anomalies in a dataset. These windows are used by the NAB scoring function to compute weights of individual anomaly detections for a given dataset. It is

of note that, in the NAB scoring system, if there are multiple detections within a particular anomaly window, then only the earliest detection is considered as a true positive, and all subsequent anomaly detections within the window are ignored.

2) *Application profile* - NAB defines three different *application profiles*: standard, reward low false positives, and reward low false negatives. In the standard profile, relative weights are assigned to true positives, false positives and false negatives, based on the window size, whereas in the reward low false positives profile and the reward low false negatives profile, greater penalties are assigned for false positives and false negatives respectively. NAB provides anomaly scores for each of these application profiles for every dataset.

3) *Scoring function* - Given an anomaly window and an application profile, NAB uses the following sigmoidal scoring function to compute weight of each anomaly detection[14]:

$$\sigma^A(y) = (A_{TP} - A_{FP})\Big(\frac{1}{1+e^{5y}}\Big) - 1 \tag{1}$$

where A is the given application profile, y is the relative position of the detection in the given anomaly window, A_{TP} is the corresponding weight of true positives in profile A, and A_{FP} is the corresponding weight of false positives in profile A. This scoring function is designed so as to give higher positive scores to true positive detections earlier in a window and negative scores to detections outside the window. Using the weights of individual detections, the 'raw score' of a dataset, denoted by S_d^A is calculated as follows[14]:

$$S_d^A = \Big(\sum_{y \in Y_d} \sigma^A(y)\Big) + A_{FN} f_d \tag{2}$$

In this equation, the sum of weighted scores of true positives and false positives is computed, which is then discounted by the weighted sum of missed detections computed as $A_{FN} f_d$, where A_{FN} denotes the weight of false negatives in profile A, and f_d denotes the total number of false negatives (or missed detections) in dataset d. Using Eq. 2, the final normalized, reported score over all 58 datasets is computed as follows:

$$S_{NAB}^A = 100.\frac{S^A S_{null}^A}{S_{perfect}^A S_{null}^A} \tag{3}$$

where S^A is the sum of raw scores over all datasets, (i.e., $\sum_d S_d^A$), S_{null}^A is the sum of raw scores achieved by a null detector (i.e., the one that outputs no anomaly detections), and , $S_{perfect}^A$ is the sum of raw scores achieved by a (hypothetical) perfect detector (i.e., the one that outputs all true positives and no false positives/negatives). This equation ensures that the maximum NAB score of an anomaly detection algorithm is 100, and the score of a null detector is 0.

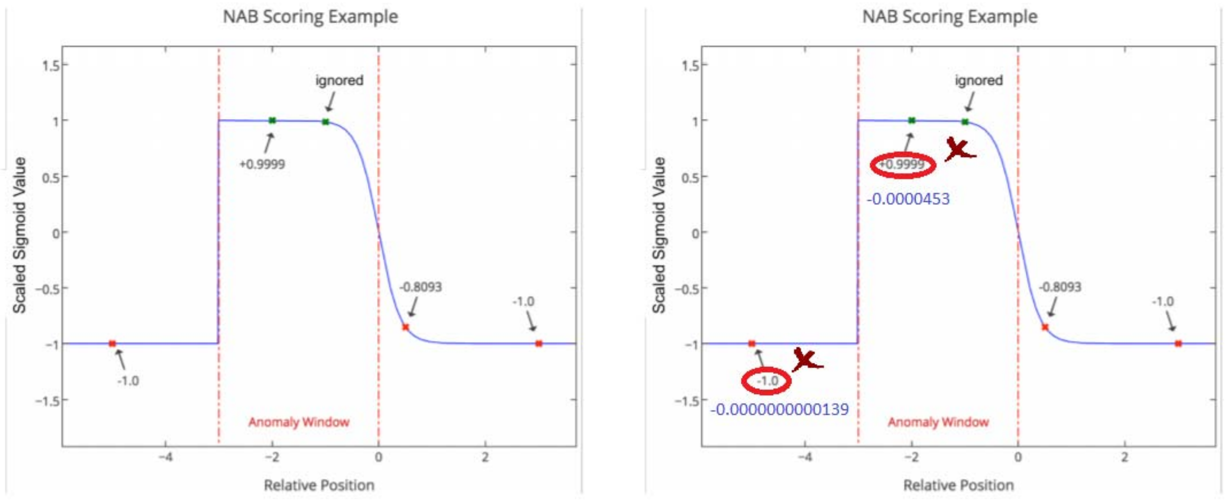
IV. Key Challenges in the NAB Framework

Though NAB provides valuable labeled datasets for comparison of anomaly detection algorithms, the framework has certain challenges which makes it difficult to use in real-world industry applications. In this section, we outline and describe these challenges in detail.

A. Challenges in the NAB Scoring System

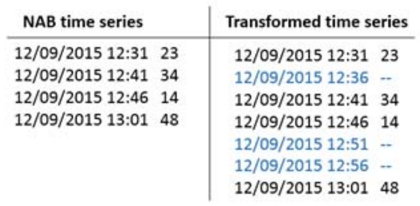
1) *Determining anomaly window's size-* The NAB scoring system is based on anomaly windows, but there is no systematic way of determining the optimal size of anomaly windows. Lavin and Ahmad in [14] chose the window size to be 10% the number of instances in a dataset, divided by the number of anomalies in the given dataset. But, window size cannot be chosen in this way in many real-world problem settings, especially in streaming environments, because the number of instances in a dataset may not be known in advance (e.g., in streaming environment) and more importantly, it is impossible to know the number of anomalies in a streaming dataset *before* an anomaly detection algorithm is executed on the dataset. This leaves little room for applicability of the proposed way of selecting window size in real-life problem scenarios.

2) *Gaps in scoring function-* The scoring function of the NAB framework is primarily based on Eq. 1, which is used to compute the weight of each detection given an anomaly window and an application profile. But, this equation is not well-defined and has the following gaps. First, it is unclear as to how y (which denotes the relative position of a detection in the given anomaly window) should be computed. Hence, the range of values that y can take on remains unknown. Second, in Eq. 1, value of *5y* is chosen as the power of e (in the second term). But the reason for choosing the value of 5 is not clear, due to which it remains unknown as to what impact can other values, like 5.1 or 50, have on the weights of detections and how will it change the overall NAB score of an algorithm. Third, the scoring example provided in [14] does not match with Eq. 1. For convenience sake, we replicate that figure in Fig. 1a, where we see that when the relative position of the detection is outside the anomaly window, i.e., at y (approximately) equal to -5, the scaled sigmoid value (i.e., the second term of Eq. 1) is -1. This is incorrect as, in this case, the second term of Eq. 1 would yield a value close to 0, i.e., -0.0000000000139 (calculated as $(\frac{1}{1+e^{5(-5)}}) - 1$). This is shown pictorially in Fig. 1b. Similarly, when the relative position of the detection is -2 in the anomaly window, then the scaled sigmoid value is +0.9999. This is incorrect as well as, the second term of Eq. 1 would still yield a value close to 0, i.e., -0.0000453, (calculated as $(\frac{1}{1+e^{5(-2)}}) - 1$).
Lastly, in this example, the authors in [14] state that the scaled sigmoid values are multiplied by the *relevant*



(a) (b)

Fig. 1: Example of NAB scoring system: (a) Example as provided originally in the NAB framework description[14] (b) Example after analysis and corrections



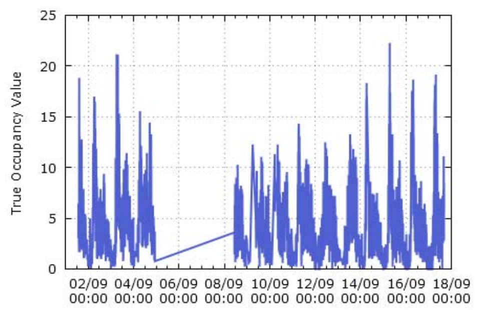
NAB time series		Transformed time series	
12/09/2015 12:31	23	12/09/2015 12:31	23
12/09/2015 12:41	34	12/09/2015 12:36	--
12/09/2015 12:46	14	12/09/2015 12:41	34
12/09/2015 13:01	48	12/09/2015 12:46	14
		12/09/2015 12:51	--
		12/09/2015 12:56	--
		12/09/2015 13:01	48

Fig. 2: Example of time series snippet after transformation

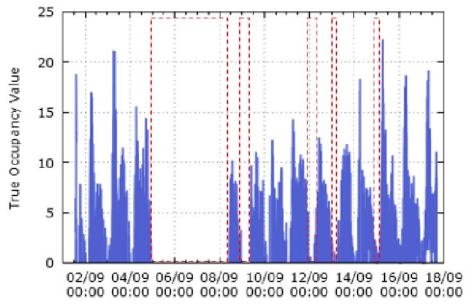
application profile weight as per Eq. 1, thereby yielding the NAB score for this example as $-1.0A_{FP} + 0.9999A_{TP} - 0.8093A_{FP} - 1.0A_{FP}$. Again, this is not correct because in Eq. 1, the scaled sigmoid value of each detection is multiplied by the difference of false positives weight and true positives weight of the given application profile A, i.e., $(A_{TP} - A_{FP})$. Hence, the correct, unnormalized NAB score for this example is $-1.0(A_{TP} - A_{FP}) + 0.9999(A_{TP} - A_{FP}) - 0.8093(A_{TP} - A_{FP}) - 1.0(A_{TP} - A_{FP})$, or simply $(A_{TP} - A_{FP})(-1.0 + 0.9999 - 0.8093 - 1.0)$.

B. Challenges in the NAB Datasets

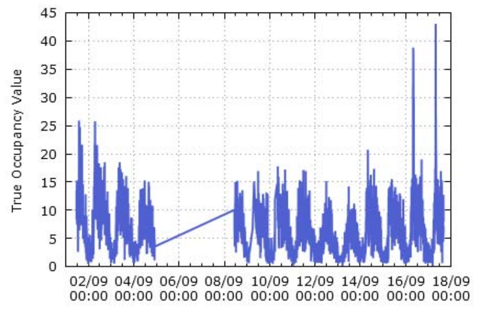
1) *Missing values in datasets-* A time-series is generally composed of data points that are collected at a constant interval or frequency (e.g., every 1 hour, every 10 minutes). In other words, the spacing of observation times is constant in a general time series. In scenarios where observations are not collected at a constant interval, one or more forms of data preprocessing is done to create equally spaced time series before fitting a model to the dataset, failing which a time series model may yield inaccurate results. NAB provides time series datasets that are mostly composed of observations gathered at a constant interval, but some of the datasets do not have this desirable property due to which time series models fail to perform well on these datasets. Some of these datasets are shown in Fig. 3. Fig. 3a shows the `occupancy_6005` dataset as provided in the NAB framework. This dataset contains observations for 17 days, most of which are collected every 5 minutes but some are collected at other varying intervals. Going by Fig. 3a (which shows the data as-is, provided by the NAB framework), it is difficult to assess the problem of uneven spacing of observation times in this dataset. In order to highlight this problem clearly in this dataset, we fill in the missing timestamps (with empty values) between every pair of observations that are collected at interval greater than 5 minutes. This process is illustrated in Fig. 2 where we see that the transformed time series has additional observations (marked in blue) albeit with empty values, with the result that the transformed time series is equally spaced and hence fit for use in a conventional time series model.
The transformed time series corresponding to `occupancy_6005` dataset is shown in Fig. 3b, in which it can be clearly seen that, post transformation, `occupancy_6005` dataset has many missing values (marked in red) which must be taken into consideration before fitting any time series or machine learning model to this dataset. On similar lines, Fig. 3c, 3e and 3g show `occupancy_t4013`, `speed_6005`, and `traveltime_387` datasets respectively as provided by the NAB framework, and Fig. 3d, 3f and 3h, show respective corresponding transformed time series for these datasets. In these figures, it can be seen that each of these datasets has many missing values as a result of which any conventional time series and/or



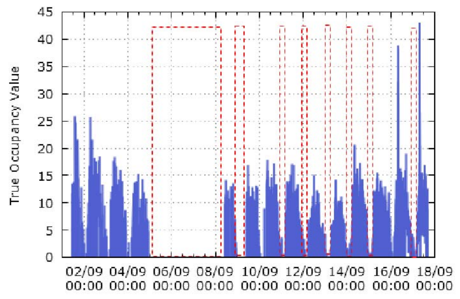
(a)



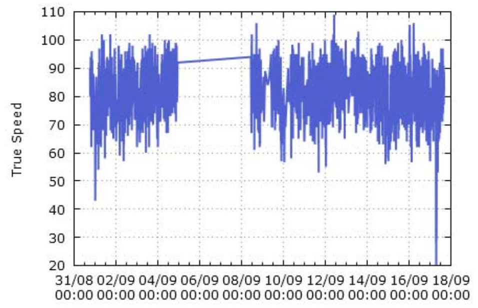
(b)



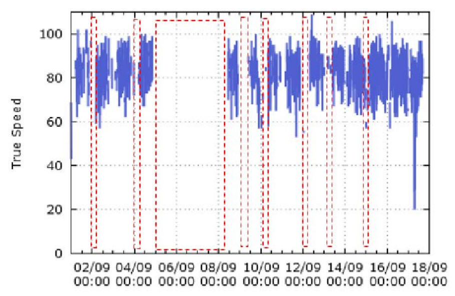
(c)



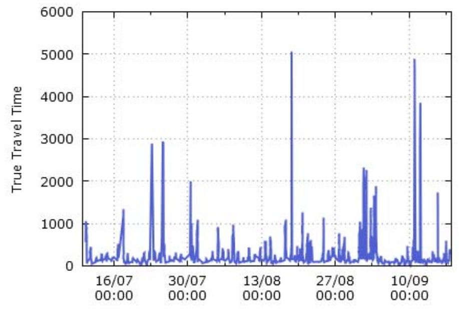
(d)



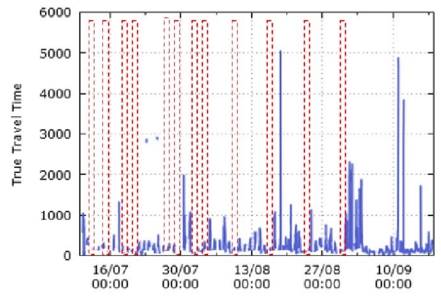
(e)



(f)



(g)



(h)

Fig. 3: Selected datasets with missing values: (a) occupancy-6005 dataset as provided in the NAB framework,(b) occupancy-6005 dataset after transformation, (c) occupancy-t4013 dataset as provided in the NAB framework,(d) occupancy-t4013 dataset after transformation, (e) speed-6005 dataset as provided in the NAB framework,(f) speed-6005 dataset after transformation, (g) traveltime-387 dataset as provided in the NAB framework,(h) traveltime-387 dataset after transformation

machine learning model, will fail to perform well on these datasets.

2) *Difference in data distribution-* In many NAB datasets (like `occupancy_6005`, `speed_6005`), the distribution of data for the first few days (e.g., 4 or 5 days) happens to be quite different from the distribution for rest of the time series. This characteristic of certain NAB datasets negatively impacts with the training of machine

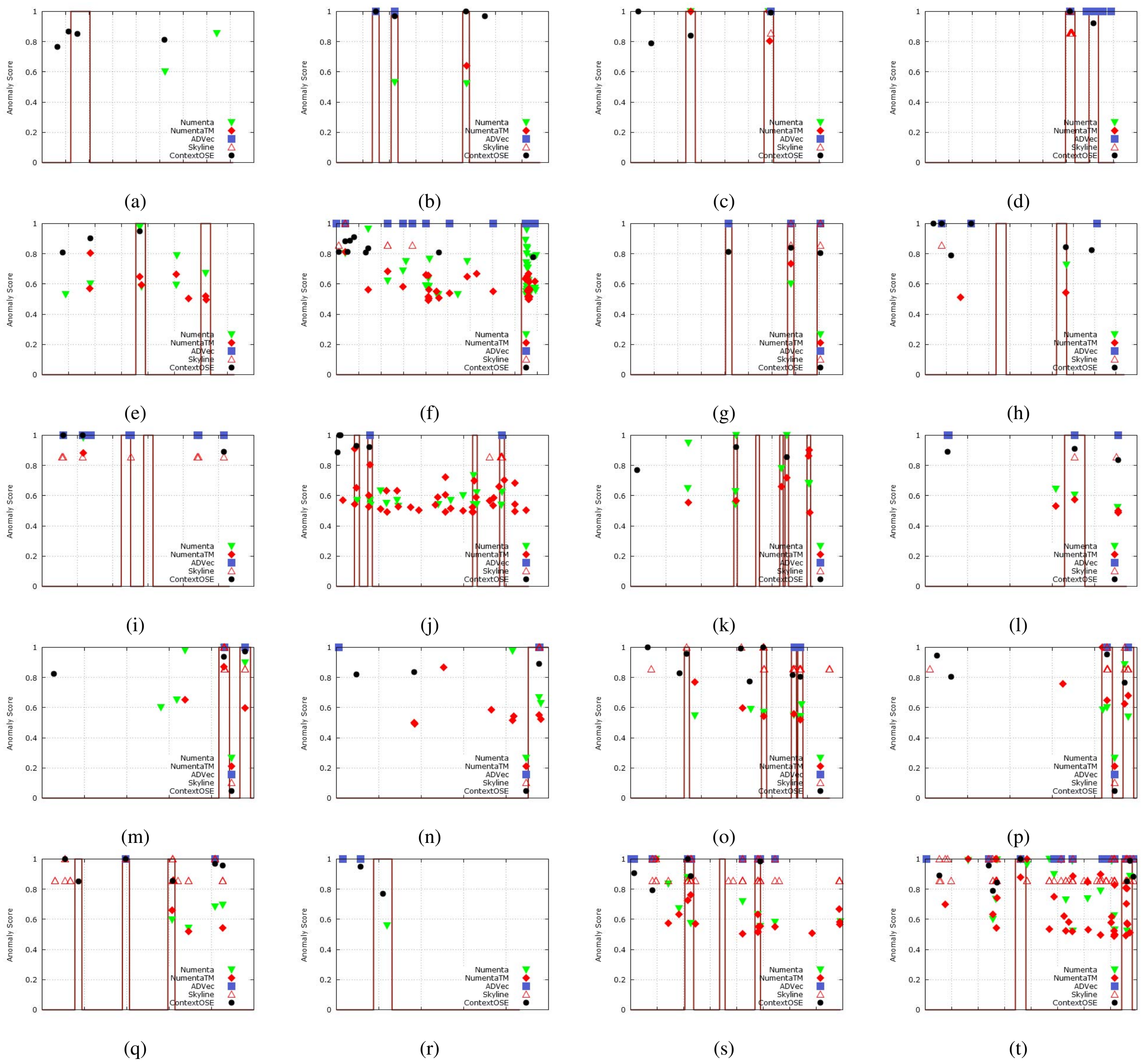


Fig. 4: Detection of anomalies by five algorithms on following, randomly selected 20 NAB datasets: (a)exchange-2-cpc-results (b) exchange-3-cpc-results (c) ec2-cpu-utilization-5f5533 (d) rds-cpu-utilization-cc0c53 (e) ambient-temperature-system-failure (f) cpu-utilization-asg-misconfiguration (g) ec2-request-latency-system-failure (h) rogue-agent-key-hold (i) rogue-agent-key-updown (j) machine-temperature-system-failure (k) nyc-taxi (l) occupancy-t4013 (m) occupancy-6005 (n) speed-6005 (o) speed-7578 (p) speed-t4013 (q) traveltime-387 (r) traveltime-451 (s) Twitter-volume-GOOG (t) Twitter-volume-IBM

learning or time series models because the first few days of data is typically used for training these models while rest of the dataset is used for the testing purpose. This difference in training and test distributions is one of the potential causes of low performance of different models that have been so far evaluated on NAB datasets (which we will illustrate in detail in the next Section).

V. Comparative Evaluation on NAB Datasets Using Standard Metrics

In this section, we perform evaluation of the predictive performance of five state-of-the-art anomaly detection algorithms (i.e., ContextOSE [15], two variations of Numenta's Hierarchical Temporal Memory (HTM) [14] (named as 'Numenta' and 'NumentaTM' in the benchmark), Etsy's Skyline [17] and Twitter's AdVec [16]), in terms of standard

TABLE I: Comparative evaluation results of five state-of-the-art anomaly detection algorithms on randomly selected 20 NAB datasets

Dataset	ContextOSE			Numenta			NumentaTM			Skyline			ADVec		
	Precision	Recall	NAB	Precision	Recall	NAB	Precision	Recall	NAB	Precision	Recall	NAB	Precision	Recall	NAB
exchange-2-cpc-results	0.5	0.006	0.83	0	0	-1.22	0	0	-1	0	0	-1	0	0	-1
exchange-3-cpc-results	0.75	0.02	2.48	1	0.013	0.64	1	0.007	-1.17	0	0	-3	1	0.02	0.71
ec2-cpu-utilization-5f5533	1	0.005	1.51	1	0.007	1.71	1	0.01	1.71	1	0.002	-0.35	1	0.002	-0.35
rds-cpu-utilization-cc0c53	1	0.005	1.72	1	0.002	-0.14	1	0.002	-0.14	1	0.10	-0.13	0.62	0.012	1.41
ambient-temperature-system-failure	0.33	0.001	-0.31	0.42	0.004	1.32	0.5	0.006	1.33	0	0	-2	0	0	-2
cpu-utilization-asg-misconfiguration	0.12	0.001	-0.14	0.64	0.018	-0.68	0.52	0.01	-0.79	0	0	-1.55	0.74	0.01	-0.02
ec2-request-latency-system-failure	1	0.009	1.99	1	0.009	1.98	1	0.009	1.98	1	0.014	0.09	1	0.02	2.02
machine-temperature-system-failure	1	0.001	-0.16	0.58	0.004	2.89	0.27	0.004	0.20	0.97	0.01	-2.13	1	0.02	-0.23
nyc-taxi	1	0.002	-1.32	0.77	0.007	2.34	0.85	0.006	2.34	0	0	-5	0	0	-5
rogue-agent-key-hold	0.33	0.005	-0.95	1	0.005	-0.86	0.5	0.005	-0.95	0	0	-2	0	0	-2.22
rogue-agent-key-updown	0	0	-2.22	0.14	0.002	-1.18	0	0	-2.11	0	0	-2.55	0.11	0.002	-1.39
occupancy-6005	0.5	0.004	0.75	0.33	0.004	0.64	0.2	0.004	0.42	0.5	0.004	0.76	0.5	0.004	0.75
occupancy-t4013	1	0.008	1.73	0.4	0.008	1.39	0.66	0.008	1.61	1	0.04	1.73	1	0.02	1.72
speed-6005	0.5	0.004	0.73	0.66	0.008	0.72	0.25	0.008	0.17	1	0.01	0.84	1	0.01	0.84
speed-7578	0.57	0.03	3.36	0.66	0.03	1.34	0.6	0.02	1.45	0.86	0.16	3.39	1	0.01	-0.34
speed-t4013	1	0.008	1.86	1	0.01	1.98	0.8	0.01	1.87	1	0.06	1.86	1	0.01	1.74
TravelTime-387	0.6	0.01	2.22	0.25	0.004	-1.58	0.33	0.004	-1.45	0.62	0.07	-0.61	0.2	0.004	-1.57
TravelTime-451	1	0.005	0.86	1	0.005	0.56	0	0	-1	0	0	-1	0	0	-1
Twitter-volume-GOOG	0.75	0.002	0.05	0.36	0.003	0.06	0.38	0.005	-0.31	0.59	0.02	-2.06	0.81	0.01	0.29
Twitter-volume-IBM	0.37	0.002	1.25	0.15	0.002	-0.07	0.22	0.005	-1.21	0.22	0.01	-4.38	0.5	0.009	0.30

metrics, like precision and recall, over 20 randomly selected NAB datasets. The results of this evaluation are presented in Table 1, where we see the standard metrics, like precision and recall, as well as the NAB score for each algorithm over selected 20 NAB datasets. In this table, it can be observed that for certain datasets (like, ec2-cpu-utilization-5f5533 and ec2-request-latency-system-failure), each algorithm achieves perfect precision, i.e., 1. But, it must be noted that each algorithm detects only 1–4 anomalies out of 346–402 anomalies in these datasets, which is reflected in poor recall of 0.005–0.01 for each algorithm. For other datasets like ambient-temperature-system-failure and rogue-agent-key-updown, all algorithms under consideration perform poorly in terms of precision (i.e.,0–0.5). The total number of anomalies detected by these algorithms on ambient-temperature-system-failure and rogue-agent-key-updown is also low, i.e., 1–4 out of 530–726 anomalies, which is again reflected in low recall of these algorithms.

In Table 1, it can be observed that across all 20 datasets, each algorithm under consideration achieves substantially low recall, i.e., in the range of 0–0.16. This implies that these anomaly detection algorithms are essentially ineffective on these datasets because the total number of anomalies detected by these algorithms is in the range of 1–7 while the total number of true anomalies in these datasets is usually in the range of 150–1588. Interestingly, NAB still assigns high scores to these algorithms for some of the datasets, as NAB scores are designed to reward early anomaly detection and not total detection rate (or recall). For instance, NAB assigns a (high) score of 3.36 to ContextOSE for speed-7578 dataset, which implies the algorithm performs well (from early detection perspective) on this dataset, but in absolute terms, ContextOSE detects only 4 out of 116 anomalies in this dataset, and in the process, it also yielded 3 false positives. Going by this performance in absolute terms, one might argue that this performance is not sufficient for applicability of this algorithm in practical scenarios, but (high) NAB scores in such cases seem to suggest the opposite.

In certain applications and/or domains, early detection of anomalies is indeed desirable and must be rewarded higher than anomalies that are detected later. But, in general problem scenarios, the total number of detections made by an algorithm does reflect the efficacy of the algorithm on given datasets, and hence must also be taken into consideration. To assess the efficacy of the aforementioned five algorithms on these lines, we plot the total number of anomaly detections made by each algorithm on same 20 datasets in Fig. 4. In this figure, the true anomalies are plotted in red line which pictorially forms one or more ‘windows’ in each dataset (as the true anomalies are marked in a sequence in all datasets), and data points classified as anomalous (i.e., the ones that have anomaly score greater than the thresholds specified in the NAB framework) for each of the five algorithms are also presented. In these plots, we see that for certain datasets like ec2-cpu-utilization-5f5533 (shown in Fig. 4c) and exchange-2-cpc-results (shown in Fig. 4a), the anomalies detected by these algorithms does seem to fall within the anomaly windows with very few false positives, but for other datasets like cpu-utilization-asg-misconfiguration, machine-temperature-system-failure, Twitter-volume-GOOG, and Twitter-volume-IBM (shown in Fig. 4f, Fig. 4j, Fig. 4s, and Fig. 4t respectively), none of anomaly detection algorithms under consideration seem to perform well as most of the detections made are outside the anomaly windows, which in turn implies that they resulted in many false positives and few true positives.

In order to have a closer look at the aggregated false positives and false negatives for each algorithm, we consider the results provided under standard application profile of the NAB framework and present the total number of true positives, false positives, and false negatives aggregated over all 58 datasets for each algorithm in Table 2. In this table, we see that the number false positives relative to the number of true positives is quite high for each algorithm. For instance, NumentaTM correctly classifies 201 anomalies in all datasets but at the same time, it falsely classifies 225 data points as anomalous. Similarly, ADVec yields 317 true positives but also yields 295 false positives. Furthermore, the number of missed anomalies (i.e., false negatives) is too high for each algorithm. For real-world industry applications, this kind of performance is usually unacceptable, which makes these algorithms unsuitable for use in commercial anomaly detection software products and solutions. This demonstrates a need for better anomaly detection methods that could perform well not just from the perspective of early detection of anomalies, but also from the general perspective of standard metrics like precision, recall and false positive rate. In the near future, we will aim to address this gap by proposing an anomaly detection algorithm that could perform well on NAB datasets using standard metrics.

TABLE II: Total number of true positives, false positives and false negatives aggregated over all 58 NAB datasets

Method	True Positives	False Positives	False Negatives
ContextOSE	96	64	33399
Numenta	167	147	33328
NumentaTM	201	225	33294
ADVec	317	295	33178
Skyline	664	497	32831

VI. Conclusion

In this work, we provide an in-depth analysis of the key components of the NAB framework, i.e., the NAB scoring system and labeled datasets. Specifically, we described the gaps that exist today in the NAB scoring system which makes it difficult to use in real-world problem settings. We also illustrated the challenges in using NAB datasets which hinders the applicability of a conventional time series based or a machine learning based model to these datasets. Furthermore, we also demonstrated the performance of five state-of-the-art anomaly detection algorithms on NAB datasets and showed that these algorithms do not perform well on most of the datasets in terms of standard metrics like precision and recall. In our future work, we will propose methods (based on time series and/or machine learning) to overcome the challenges of the NAB framework, thereby adding a significant milestone for open research in the domain of anomaly detection.

References

[1] K. Sequeira and M. Zaki, "Admit: Anomaly-based data mining for intrusions," in *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '02, 2002.

[2] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos, "Online outlier detection in sensor data using non-parametric models," in *Proceedings of the 32nd International Conference on Very Large Data Bases*, ser. VLDB '06, 2006.

[3] C. C. Aggarwal, "On abnormality detection in spuriously populated data streams," in *Proceedings of ACM SIAM Conference on Data Mining*, 2005.

[4] S. L. Scott, "Detecting network intrusion using a markov modulated non-homogeneous poisson process," *Journal of the American Statistical Association*, 2000.

[5] D. Dasgupta and F. Nino, "A comparison of negative and positive selection algorithms in novel pattern detection," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, vol. 1, 2000.

[6] M. Atallah, W. Szpankowski, and R. Gwadera, "Detection of significant sets of episodes in event sequences," in *Proceedings of the Fourth IEEE International Conference on Data Mining, ICDM '04.*, Nov 2004.

[7] M. Kloft and P. Laskov, "Online anomaly detection under adversarial impact." in *AISTATS*, 2010, pp. 405–412.

[8] N. Görnitz, M. M. Kloft, K. Rieck, and U. Brefeld, "Toward supervised anomaly detection," *Journal of Artificial Intelligence Research*, 2013.

[9] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: Identifying density-based local outliers," *SIGMOD Rec.*, vol. 29, no. 2, May 2000.

[10] X. Song, M. Wu, C. Jermaine, and S. Ranka, "Conditional anomaly detection," *IEEE Trans. on Knowl. and Data Eng.*, vol. 19, no. 5, May 2007.

[11] I. Steinwart, D. Hush, and C. Scovel, "A classification framework for anomaly detection," *J. Mach. Learn. Res.*, vol. 6, Dec. 2005.

[12] M. Kloft, U. Brefeld, P. Düessel, C. Gehl, and P. Laskov, "Automatic feature selection for anomaly detection," in *Proceedings of the 1st ACM workshop on Workshop on AISec*. ACM, 2008, pp. 71–76.

[13] G. Blanchard, G. Lee, and C. Scott, "Semi-supervised novelty detection," *J. Mach. Learn. Res.*, vol. 11, Dec. 2010.

[14] A. Lavin and S. Ahmad, "Evaluating real-time anomaly detection algorithms - the numenta anomaly benchmark," *CoRR*, vol. abs/1510.03336, 2015.

[15] "Contextual Anomaly Detector," https://github.com/smirmik/CAD, 2015.

[16] O. Vallis, J. Hochenbaum, and A. Kejariwal, "A novel technique for long-term anomaly detection in the cloud," in *Proceedings of the 6th USENIX Conference on Hot Topics in Cloud Computing*, ser. HotCloud'14, 2014.

[17] "Etsy Skyline," https://anomaly.io/detect-anomalies-skyline, 2013.

[18] A. N. Srivastava and B. Zane-Ulman, "Discovering recurring anomalies in text reports regarding complex space systems," in *2005 IEEE Aerospace Conference*, March 2005.

[19] S. Singh and M. Markou, "An approach to novelty detection applied to the classification of image regions," *IEEE Trans. on Knowl. and Data Eng.*, vol. 16, no. 4, Apr. 2004.

[20] D. Pokrajac, "Incremental local outlier detection for data streams," in *In Proceedings of IEEE Symposium on Computational Intelligence and Data Mining*, 2007, pp. 504–515.

[21] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, Jul. 2009.

[22] "Numenta Anomaly Benchmark," http://numenta.com/numenta-anomaly-benchmark/, 2013.