Data Quality in Data Streams by Modular Change Point Detection

Yaron Kanza¹, Rajat Malik¹, Divesh Srivastava¹, Caroline Stone¹ and Gordon Woodhull¹

¹AT&T Chief Data Office, New Jersey, USA

Abstract

Sensors that collect data from complex systems generate a stream of measurements, for example, measuring CPU utilization of machines in a data center, gathering meteorological data like atmospheric pressure and humidity levels across the USA, or tracking the occupancy of taxis in a large city. Downstream systems use the streamed data in a variety of applications, including training machine learning models and making data-driven decisions as part of automation. This makes data quality critical and requires detecting significant, unexpected, and rapid changes in indicative features of the streaming data. This can be done by detecting change points in the stream – points where the underlying distribution of a statistical feature of the stream fundamentally changes. In this paper, we discuss different types of change points in the data stream – changes that indicate a potential data quality problem. We present a modular method for combining operations on data streams to examine data quality in a flexible and adaptable way. Experiments over real-world and synthetic data streams show the effectiveness of the modular approach in comparison to traditional anomaly detection methods.

Keywords

Anomaly detection, change point detection, data streams, modular architecture

1. Introduction

When monitoring complex systems like cellular networks, data centers, cloud infrastructures and content delivery networks, the monitoring system generates a data stream of telemetry, such as processing times, data transfer times, communication latency, CPU utilization, memory usage, network throughput, and other statistics that can help to track the health of the system. Monitoring is also used for collecting meteorological data for weather forecasting, traffic data to regulate and mitigate congestion in highways and highly-used roads, tracking the operation of machines and facilities, and continuously gathering data for real-time systems.

Data streams are often analyzed to detect anomalies and irregularities. Anomalies and irregularities in the stream may indicate a problem in the underlying system or may reveal an event that requires intervention. Since the data in the stream is the basis for critical decisions, poor data quality may affect those decisions. In addition, collected data sets are often used for training machine learning models. The models are trained to learn the expected behavior of systems and applications. Thus, the data that is fed into these models in the training process should be accurate and representative. This requires high data quality. Otherwise, the trained models could be biased or yield inaccurate results. The impact of data quality on machine learning is discussed in [1].

Maintaining high-quality data is crucial when critical applications depend on the monitored system or on models that are trained over the data. This is essential in applications for forecasting events, and for detecting security attacks, frauds, outages, and the effect of natural events like storms on infrastructures and services.

Data quality has many aspects, including *completeness* (no missing data), *consistency* (the data does not lead to contradictory inferences), *cleanliness* (no noise), *conformity* (complying with standards and rules), and *continuity* (uniformity in the arrival of the data). Some of these aspects can be evaluated using standard anomaly detection tools, but only to a limited extent. Therefore, there is a need to combine a variety of tools for effective data-quality assurance.

There are many tools and methods for detecting anomalies (outliers) in streaming data [2]. Anomalies are values in the data stream that are significantly different from the values that are expected based on previous observations. Often, anomalies can indicate that the system does not function properly. However, most anomalies are ephemeral and can be ignored because by the time that they are noticed the system is already back to normal. So, it is often essential to focus on lasting changes in the data stream, detect them, and alert on them. This raises several questions. First, what type of changes should the system detect? Second, how should changes be detected? Third, how should the changes be reported to users in a way that is effective and actionable,

Joint Workshops at 49th International Conference on Very Large Data Bases (VLDBW'23) — the 12th International Workshop on Quality in Databases (QDB'23), August 28 - September 1, 2023, Vancouver, Canada

[☆] kanza@research.att.com (Y. Kanza); rmalik@research.att.com (R. Malik); divesh@research.att.com (D. Srivastava);

cs831k@att.com (C. Stone); gordon@research.att.com

⁽G. Woodhull)

O 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License
Attribution 4.0 International (CC BY 4.0).
CEUR Workshop Proceedings (CEUR-WS.org)

without overwhelming the user with too many alerts but also without missing critical alerts?

In this paper our focus is on detection of *change points*, that is, points where the underlying distribution of a statistical feature of the stream changes in a significant, nonephemeral, and unexpected way. We present a modular architecture for change point detection over streaming data, to provide flexibility and adaptability for a large variety of data streams and diverse use cases.

The paper is organized as follows. In Section 2 we discuss related work. Section 3 introduces quality measures for data streams. In Section 4 we present methods for detection of change points. Section 5 describes our modular architecture and its benefits. Section 6 presents the results of our experimental evaluation. In Section 7 we discuss our conclusions and future work.

2. Related Work

The study in this paper is related to the following three research areas: data quality, anomaly detection and change point detection. These areas were studied extensively, however, the approach of a modular change point detection, which we present in this paper, is novel.

Data quality. Quality measures for data streams have been studied in different contexts [3, 4, 5, 6]. Klein [7] examined data quality in sensor data streaming. Karkouch et al. [8] explored data quality in streams produced by IoT devices. Brown et al. [9] studied methods for coping with glitches in spatiotemporal streams by applying smoothing and imputation to data streams produced by spatially distributed sensors. The importance of empiricism in data quality studies has been emphasized in [10].

Anomaly detection. Anomaly detection in time series has received a lot of attention in the literature. Many different anomaly detection methods have been developed and tested [11, 12, 13]. See Schmidl et al. [2] for a recent comparison of many methods. However, point anomalies are often ephemeral and do not reflect significant changes in the stream or data-quality issues. Some studies of anomalies considered anomalous subsequences rather than point anomalies. Boniol et al. [14, 15] studied a method for finding subsequences of a time series that are the farthest from a normal distribution. However, their assumption of normal distribution in the data does not hold in many real-world data streams, like those that we explore. Moreover, these studies do not focus on data quality or on change point detection.

Change point detection. Change detection has been studied for time series [16] and data streams [17, 18, 19], however, these methods were not designed for data quality measures and do not explore the modular approach that we present in this paper.

3. Quality Measures over Streams

In this section we provide formal definitions and present the problem of discovering changes in the underlying distribution of quality measures over a data stream. Unlike time series with a bounded number of points, streams often have high volume, velocity, variety and veracity, so quality measurements should be adapted to streams, accordingly [20]. We present examples and illustrate our method based on real data taken from the Numenta Anomaly Benchmark¹, e.g., a sequence from a stream of Taxi occupancy in The Twin Cities.

A data stream is a sequence of measurements $S = m_1, m_2, \ldots$ where each measurement $m_i = (t_i, x_i)$ is a pair of valid time t_i and measured value x_i . The valid time t_i is the time when the value x_i was measured. The time when the measurement is processed as part of the stream is considered as *transaction time*. The *delay* δ_i of measurement m_i is the difference between the valid time and the transaction time.

For a time series where all the measurements are given a priori, computing statistics like mean and variance is simple. But for streaming data, new values arrive continuously and the statistics changes frequently. So, values like mean and variance should be based on recent values in the stream, not on the entire history. This can be done using a sliding window [21] or a decaying mean [22, 23]. **Sliding window.** When using a sliding window Wof size w, at time $i \geq w$, the sub-sequence $S_i[w] =$ $x_{i-w+1}, x_{i-w+2}, \ldots, x_i$ of the stream S comprises the most recent \boldsymbol{w} values in the stream up to measurement m_i . The mean μ_i , variance σ_i^2 , standard deviation σ_i , median ν_i , and other statistics of $S_i[w]$ are computed in the usual way. Since for each measurement m_i there is a different window, the statistics of $S_i[w]$ may be different from the statistics of $S_{i'}[w']$ when $i \neq i'$ or $w \neq w'$.

Decaying mean and variance. A decaying mean μ_i is computed with a decay parameter $0 < \alpha \le 1$, such that $\mu_1 = x_1$ and for i > 1, $\mu_i = \alpha x_i + (1 - \alpha)\mu_{i-1}$. We refer to the residual at time i as the difference $x_i - \mu_i$, where x_i is the measured value at time i and μ_i is the decaying mean at that point. The decaying variance at time i is the average over the squared residuals, that is, $\sigma_1^2 = 0$ and for i > 1, $\sigma_i^2 = \alpha (x_i - \mu_i)^2 + (1 - \alpha)(\sigma_{i-1}^2)$. **Point outlier.** A point outlier is a value that significantly exceeds the expected value, e.g., a value x_i that is above or below the mean by more than 2.5 standard deviations, $|x_i - \mu_i| > 2.5\sigma_i$. Outliers could indicate a volatile data quality problem. In Fig. 1, the red dots are outliers returned by the kNN outlier detection method.

Data quality. Various data quality issues can be detected based on changes in the statistical properties of a data

¹https://github.com/numenta/NAB



Figure 1: A sequence of streaming data with the occupancy of taxis in New York City. Outliers are marked by red dots.

stream. Some of the characteristics of the stream can be measured using the moments of the distribution, of the measured values, or of the delays. Commonly, for a random variable X, the *n*-th moment is $\frac{E[(X-\mu)^n]}{\sigma^n}$, i.e., the normalized expectancy of the residuals to the power of *n*. The following are measurable changes in a data stream that can be evaluated using moments.

• Level Shift in Value (first moment). A significant change in the values of measurements can be the result of a dataquality problem. For example, in a system that monitors temperatures, an unexpected lasting increase or decrease in the measured values can be the result of a calibration issue or malfunction of sensors. In Fig. 1, there is a level shift around the date of September 12.

• Level Shift in Variance (second moment). A significant change in the variance of measurements can be the result of noise. The noise could affect measurement accuracy and impact the data quality. For example, noise could be the result of partial interference to a sensor.

• Level Shift in Skewness (third moment). The skew measures the symmetry of the distribution. It can be measured as the distribution of the differences $\mu_i - \nu_i$ between the mean and median values. It may reflect bias that affects data quality.

• *Changes in Volume.* The volume is the number of measurements that arrive at each time interval. Unexpected changes in the volume may indicate that some measurements are missing, duplicated or arrive from data sources that should not be included in the stream.

• Delayed Data (first moment). The measurements may arrive one by one or in a batch. The delay is the difference between the valid time and the transaction time of the measurement. A significant increase in the difference may indicate that something is delaying the data arrival, which may lead to missing data, data points that arrive out of order or measurements that arrive too late for some online applications.

• Varying Delay (second moment). A change in the variance of the delay indicates that measurements are arriving inconsistently. This can often cause data loss or improper data processing by downstream applications. • *Skewness of Delay (third moment).* The delay may behave somewhat like an asymmetric wave and the skew will indicate whether the problem is increasing or decreasing.

• *Outlier Rate.* In many cases, the rate of point outliers is an indicator of data quality problems, e.g., jitter in a communication network. In some systems it is expected to have a few glitches and anomalies from time to time. But a major increase in the rate or concentration of point outliers is regarded as a data quality issue.

The goal is to apply data quality measurements in an effective and modular way and raise an alert when there are significant changes in the stream for the relevant data quality measures.

4. Detecting Changes in a Stream

Data streams and their statistical properties vary and depend on the application. In this paper we suggest a modular approach for anomaly detection over data streams. Each module receives a stream of data items and returns a stream of data items. A modular architecture is achieved by combining different modules such that the output stream of one module is the input of the next module. In this section we define some of the modules and their composition.

Value extraction. Given the initial stream S, the first module extracts the statistical values that we want to measure. For example, we can extract from the stream of delays $\delta_1, \delta_2, \ldots$, a stream of values x_1, x_2, \ldots , a stream of delays $\delta_1, \delta_2, \ldots$, a stream of measured values or delays, a stream of point outliers, and so on. The residuals for computing the mean, the variance or the skew can be based on a sliding window or a decaying mean.

Smoothing and imputation. In some cases, we may want to apply smoothing or convolution to emphasize certain features of the stream. Smoothing can be done in different ways, e.g., by replacing values with smoothed values s_1, s_2, \ldots based on a moving average and a trend factor β , where $s_1 = x_1, l_1 = x_2 - x_1, s_i = \alpha x_i + (1 - \alpha)(s_{i-1} + l_{i-1})$ and $l_i = \beta(s_i - s_{i-1}) + (1 - \beta)l_{i-1}$ for some $0 < \alpha < 1$ and $0 < \beta < 1$. Seasonality can also be included in the smoothing using Holt Winters smoothing [24]. Smoothing can also be executed using Kernel Density Estimation (KDE) [25], by applying a kernel function to the stream.

Predicted values using a moving average, Holt Winters exponential smoothing, ARIMA and other forecasting methods can be used for imputation of missing values to create a stream that is more complete if the next step of the processing is by a method that does not cope well with missing values.



Figure 2: Consecutive windows for distribution comparison.



Figure 3: The Earth Mover's Distance for sliding consecutive windows over the sequence in Fig. 2.



Figure 4: Applying rolling Z-score to the Earth Mover's Distance in Fig. 3. Outliers are depicted as red dots.

Distribution comparison with moving windows. A comparison of the underlying distribution is executed for two consecutive moving windows. By measuring the distance between the distributions, we get a new stream of values. Formally, given the stream S, let $x_{i+1-w}, x_{i+2-w}, \ldots, x_i$ be w values of window $S_i[w]$, and let $x_{i+1}, x_{i+2}, \ldots, x_{i+w}$ be w values of window $S_{i+w}[w]$. Note that $S_i[w]$ and $S_{i+w}[w]$ are consecutive windows. The distributions $D_i[w]$ and $D_{i+w}[w]$ of the values in the windows $S_i[w]$ and $S_{i+w}[w]$ are compared by computing the distance between them. It can be done using Earth Mover's Distance (EMD), also known as Wasserstein distance, Jensen-Shannon divergence, Kullback–Leibler divergence, etc. For every i, the difference between the distributions $D_i[w]$ and $D_{i+w}[w]$ yields a value d_i , and the result is a sequence $d_i, d_{i+1}, d_{i+2}, \ldots$, that is, a stream of differences between the distributions. Extreme values in this stream indicate a significant



Figure 5: Change in variance for the taxi occupancy stream around September 12.



Figure 6: The kNN outliers over the rolling variance for the sequence in Fig. 5.



Figure 7: EMD of sliding windows over the sequence in Fig. 5.

change in the distribution, i.e., a change point.

Rolling Z-score. In each stream, including the stream that is produced by the comparison of distributions over the sliding windows, we can find extreme values by using Z-score with respect to the moving average, or by some other anomaly detection method. The extreme values are clustered, to prevent a burst of alerts. In Fig. 4 we see the rolling Z-score as the blue line and the extreme values as a cluster of red dots.

Combining modules. In figures 5-8 we see how a composition of modules is applied to detect a level shift in the variance. Fig. 7 shows the stream that is produced by applying EMD to the two rolling consecutive windows. Note that there are two large peaks or elevated parts of the sequence. One is at the beginning of the change and the other is at the end of it. Fig. 8 shows the rolling Z-score and the extreme values when applied to the sequence in Fig. 7. We can see the effectiveness of detecting the change point in comparison to ordinary anomaly detection, e.g., kNN anomaly detection as depicted in Fig. 6.

Early detection. The comparison of two windows of size w may lead to a delay in detection. For measurement $m_i = (t_i, x_i)$, the comparison of the window $S_i[w]$ of w values that precede m_i and the w values of $S_{i+w}[w]$ that



Figure 8: Change point detection by computing rolling Z-score over the EMD sequence of Fig. 7.



Figure 9: Composition of components into a chain to discovers change points in the data stream.

follow m_i , requires waiting for w measurements to be delivered in the stream after seeing m_i . This delay can be mitigated by computing an estimation of the distance between the distributions and issuing a warning if the estimation indicates high likelihood for a change point.

Let $f_d(W_1, W_2)$ be a function that computes the difference in distribution for two windows. To assess the distance early, we define function e(i, j) that estimates $f_d(S_i[w], S_{i+w}[w])$ after seeing measurements m_i, \ldots, m_j , for i < j < i + w. The estimated value is the distance between the window $S_i[w]$ and the window $W_{i,j} = S_{i+j}[j]$ that contains the values x_{i+1}, \ldots, x_j . Earlier estimations are based on fewer values so they are less accurate. But they may provide an early indication of the change and trigger a warning that there is high likelihood for a change point.

5. Modular Architecture

In this paper we suggest a modular architecture for change point detection. In a modular architecture, the components receive a stream of values and produce a stream of values, so components can be composed in different ways, dynamically. Typically, processing is in a chain-like structure where the first component receives a stream of measurements as the input and the last component yields a stream of alerts, as illustrated in Fig. 9 and Fig. 10.

There are several benefits to the modular approach.



Figure 10: Examples of chains of components, as in Fig. 9.

One is reusing components, e.g., in Section 4, modules for computing EMD or rolling Z-score were applied to measurement values and to variance values. Hence, the same modules can be reused in different change point detection tasks.

Another benefit of the modular approach is dynamic composition of components. Modules can be added, adjusted or removed from a chain to accommodate changes in the streaming data. For example, consider two chains C_1 and C_2 of components. Chain C_1 designed for detecting level shift comprises (1) extracting measurement values x_1, x_2, \ldots from the stream, (2) applying EMD to the extracted values, and (3) using rolling Z-score for finding change points. Chain C_2 is the same as C_1 except that in the first step it extracts the residual values $|x_1 - \mu_1|, |x_2 - \mu_2|, \ldots$ and finds change points for the variance. In this case, if a significant increase in the variance is detected by C_2 , the system can add an initial component to C_1 for smoothing the values x_1, x_2, \ldots before applying level-shift detection. This reduces the noise caused by the large variance to prevent an undesirable effect on the level-shift detection. If a detection of missing values is applied, a detected increase in missing values may lead to adding an imputation module to chain C_2 so that the missing values will not affect the monitoring of the variance.

In some cases, we could have trees instead of linear chains, where the stream of a component can be directed to two or more branches (sub-chains). A composition may form a DAG when some components aggregate or combine the results of two or more streams.

Selecting the components and the order in which they are composed can be done based on a labeled ground truth. The system architect will examine typical data quality issues in the use case the system is built for and will try different combinations of modules, to find the combination that provides the best detection accuracy. This process can be automated so that the system could check the detection chains periodically against the ground truth and the best combination of modules will be selected and used.

We implemented the modules in Python on top of Databricks, to utilize the large distributed storage and computation capacity of Spark and have the flexibility of Python and Databricks notebooks. The modular ap-



Figure 11: EMD and Z-score over the CPU utilization stream.



Figure 12: kNN over the CPU utilization stream.

proach can also be implemented over stream processing systems like Apache Flink [26] by leveraging the stream processing API they provide. This would automatically add data-quality capabilities to these systems.

6. Experimental Evaluation

We conducted an experimental evaluation to (1) show the effectiveness of our method for change point detection, in comparison to ordinary outlier detection, and (2) demonstrate the benefits of the modular design when combining and reusing components.

Data. In the experiments we used real data from Numenta Anomaly Benchmark and streamed the measurements. To have ground truth, we inserted data-quality issues into the time series, like adding to selected regions a level shift, noise, outliers, gaps, delays, etc. This gave us the ability to distinguish between true positive cases, at a change point, and false positive cases, not near a change point. We present experiments with two data sets. (1) Taxi is real taxi occupancy data collected in 2015, in the Twin Cities Metro area, Minnesota. (2) CPU Util. is CPU utilization at an AWS cluster.

Methods. We used different combinations of components. As a baseline we used kNN - the kNN unsupervised outlier detection method. It finds the closest *k* nearest neighbors for every data point and measures the average

distance. The points with the largest distance from the population are the selected outliers. We executed kNN with a contamination rate of 0.05, that is, under the assumption that about 5% of the points are outliers. EMD and JSD are Earth Mover's distance and Jensen Shannon Divergence. They were executed with two sliding windows of size w = 100. Z-score is a rolling of Z-score based on a moving average. ARIMA is an ARIMA prediction model trained on the first 15% points of the data. As an outlier detection method, ARIMA returns the points where there is a large distance between the prediction and the observed value.

Evaluation. We computed for the different methods their precision (the percentage of correct detection cases out of all detection cases), recall (the percentage of correct detection cases out of all the true cases), percentage of false positive cases out of all positive cases, and the number of false positive cases. This shows how many false alerts could be raised. Note that too many alerts can lead to a case where alerts are ignored [27], i.e., an *alert fatigue*, so we want to avoid false alerts.

Results. The results show the effectiveness of detecting change points using the combined components. Table 1 shows that by executing EMD combined with Z-score on the modified CPU stream (Fig. 11) the detection has much higher accuracy than kNN (Fig. 12). Note that kNN has a large number of false detection cases, because it detects point outliers that are not part of a change point.

In most of our tests all the change points were detected, i.e., an alert was raised at or near the change point. In these cases the recall was 1. Note that change points are noticeable in the time series that we explored, so preventing false positive cases in these tests is a greater challenge than preventing false negative cases. The modular approach can be used to create chains with varying sensitivity to false positive or false negative, according to the application and the features of the data stream.

The results for detecting variance shifts are presented in Table 2. Note that for variance level shift, kNN generates too many false alerts. When using EMD combined with Z-score, the detection has high precision and high recall, however, with JSD the combined method does not detect the level shift and has low recall, because JSD is designed for categorical data and not for metric data.

Table 3 presents the results of detection of a shift in the frequency of point outliers. We can see in Table 3 that applying a rolling window for counting the frequency of outliers detected by kNN combined with Z-score does not have high accuracy. This is because kNN generates too many anomalies, not just near the change point. When executing ARIMA as an outlier detection, the accuracy is still low. However, when executing a rolling window that counts the outlier frequency detected by ARIMA and applying Z-score to the result we get a precision of 0.85.

Table 1Detection of a level shift.

Data set	Method	Precision	FP rate	Recall	False Alerts
CPU Util.	kNN	0.52	0.04	1	95
CPU Util.	EMD/Z-score	1	0	1	0
Taxi	kNN	0.86	0.02	1	28
Taxi	EMD/Z-score	1	0	1	0

Table 2

Detection of a variance level shift.

Data set	Method	Precision	FP rate	Recall	False Alerts
Taxi	var/kNN	0.05	0.06	1	145
Taxi	var/EMD/Z-score	1	0	1	0
Taxi	var/JSD/Z-score	0	0	0	0

7. Discussion

Detecting data quality issues in streaming data is challenging because (1) the data can frequently change, (2) not all of the data is available while it is streaming and (3) data quality can be affected by delays or changes in the underlying distribution of data arriving from the applications that generate the data. However, many data quality issues can be discovered as change points in the distribution of a statistical measure.

There are different types of statistical measures, data types, and data quality issues. Instead of developing a completely independent method for each case, we suggest a modular approach in which basic statistical components over streams can be combined and reused for detection of change points. We show in this paper that the combined components are much more effective than traditional methods for point outliers. We show results for kNN but we also tested other outlier detection methods, including ARIMA, Z-score, and Histogram-Based Outlier Scoring (HBOS), and got similar results. When using traditional outlier detection methods over real data there are too many outliers and creating an alert whenever an outlier is detected could overwhelm the users and make them ignore alerts ("The Boy Who Cried Wolf" effect [27]). Thus, it is essential to only raise alerts when there are significant change points. In this paper we show that our modular approach is effective at detecting change points without raising too many false alerts.

One of the limitations of change point detection is that it may miss *concept drifts* (changes over time in unforeseen ways) [28]. Detection of concept drifts may require a complementary method, so further study is needed.

While the modular method presented in this paper provides a promising direction for the detection of data quality issues over streaming data, more study is needed over a larger variety of data streams and for additional use cases. Future work includes the development of a method that could help users select the best combination of components and of parameters for their streaming data use cases. Future work also includes exploring the approach of ranking alerts based on the length and complexity of the chain used for the detection. The premise is that simpler chains may detect more noticeable changes, and thus, changes detected by simple chains should have higher priority than detections by complex chains.

Table 3

Detection of a level shift in the outlier rate.

Data set	Method	Precision	Recall	FP rate
Taxi	kNN/freq/Z-score	0.19	1	0.04
Taxi	ARIMA/Z-score	0.27	1	0.01
Taxi	ARIMA/Z-score/freq/Z-score	0.85	1	0.01

References

- L. Budach, M. Feuerpfeil, N. Ihde, A. Nathansen, N. Noack, H. Patzlaff, H. Harmouch, F. Naumann, The effects of data quality on machine learning performance, arXiv preprint arXiv:2207.14529 (2022).
- [2] S. Schmidl, P. Wenig, T. Papenbrock, Anomaly detection in time series: a comprehensive evaluation, PVLDB 15 (2022) 1779–1797.
- [3] T. Dasu, R. Duan, D. Srivastava, Data quality for temporal streams., IEEE Data Eng. Bull. 39 (2016) 78–92.
- [4] A. Klein, H.-H. Do, G. Hackenbroich, M. Karnstedt, W. Lehner, Representing data quality for streaming and static data, in: 2007 IEEE 23rd International Conference on Data Engineering Workshop, IEEE, 2007, pp. 3–10.
- [5] F. Korn, S. Muthukrishnan, Y. Zhu, Checks and balances: Monitoring data quality problems in network traffic databases, in: Proceedings 2003 VLDB Conference, Morgan Kaufmann, 2003, pp. 536–547.
- [6] B. Saha, D. Srivastava, Data quality: The other face of big data, in: 2014 IEEE 30th international conference on data engineering, IEEE, 2014, pp. 1294– 1297.
- [7] A. Klein, W. Lehner, Representing data quality in sensor data streaming environments, Journal of Data and Information Quality (JDIQ) 1 (2009) 1–28.
- [8] A. Karkouch, H. Mousannif, H. Al Moatassime, T. Noel, Data quality in internet of things: A stateof-the-art survey, Journal of Network and Computer Applications 73 (2016) 57–81.
- [9] P. E. Brown, T. Dasu, Y. Kanza, D. Srivastava, From rocks to pebbles: Smoothing spatiotemporal data streams in an overlay of sensors, ACM Transactions on Spatial Algorithms and Systems 5 (2019).
- [10] S. Sadiq, T. Dasu, X. L. Dong, J. Freire, I. F. Ilyas, S. Link, R. J. Miller, F. Naumann, X. Zhou, D. Srivastava, Data quality: The role of empiricism, ACM SIGMOD Record 46 (2018) 35–43.
- [11] S. Ahmad, A. Lavin, S. Purdy, Z. Agha, Unsupervised real-time anomaly detection for streaming data, Neurocomputing 262 (2017) 134–147.
- [12] A. Blázquez-García, A. Conde, U. Mori, J. A. Lozano, A review on outlier/anomaly detection in time series data, ACM Computing Surveys (CSUR) 54 (2021) 1–33.
- [13] A. Boukerche, L. Zheng, O. Alfandi, Outlier detection: Methods, models, and classification, ACM Computing Surveys (CSUR) 53 (2020) 1–37.
- [14] P. Boniol, J. Paparrizos, T. Palpanas, M. J. Franklin, Sand: Streaming subsequence anomaly detection, PVLDB 14 (2021) 1717–1729.
- [15] P. Boniol, M. Linardi, F. Roncallo, T. Palpanas, M. Meftah, E. Remy, Unsupervised and scalable

subsequence anomaly detection in large data series, The VLDB Journal (2021) 1–23.

- [16] S. Aminikhanghahi, D. J. Cook, A survey of methods for time series change point detection, Knowledge and information systems 51 (2017) 339–367.
- [17] D. Kifer, S. Ben-David, J. Gehrke, Detecting change in data streams, in: Proceedings of the 30-th International Conference on Very Large Data Bases, VLDB '04, VLDB Endowment, 2004, p. 180–191.
- [18] L. I. Kuncheva, Change detection in streaming multivariate data using likelihood detectors, IEEE transactions on knowledge and data engineering 25 (2011) 1175–1180.
- [19] D.-H. Tran, M. M. Gaber, K.-U. Sattler, Change detection in streaming data in the era of big data: models and issues, ACM SIGKDD Explorations Newsletter 16 (2014) 30–38.
- [20] J. Merino, I. Caballero, B. Rivas, M. Serrano, M. Piattini, A data quality in use model for big data, Future Generation Computer Systems 63 (2016) 123–130.
- [21] B. Babcock, M. Datar, R. Motwani, Sampling from a moving window over streaming data, in: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms, 2002, pp. 633–634.
- [22] E. Cohen, M. Strauss, Maintaining time-decaying stream aggregates, in: Proc. of the 22nd ACM Symposium on Principles of Database Systems, 2003, p. 223–233.
- [23] G. Cormode, F. Korn, S. Tirthapura, Exponentially decayed aggregates on data streams, in: 24th Int. Conf. on Data Engineering, IEEE, 2008, pp. 1379–1381.
- [24] S. Gelper, R. Fried, C. Croux, Robust forecasting with exponential and holt–winters smoothing, Journal of forecasting 29 (2010) 285–300.
- [25] M. Rosenblatt, Remarks on some nonparametric estimates of a density function, The annals of mathematical statistics (1956) 832–837.
- [26] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, K. Tzoumas, Apache Flink: Stream and batch processing in a single engine, The Bulletin of the Technical Committee on Data Engineering 38 (2015).
- [27] P. E. Brown, T. Dasu, Y. Kanza, E. Koutsofios, R. Malik, D. Srivastava, Don't cry wolf, in: International Conference on Data Science and Advanced Analytics (DSAA), IEEE, 2019, pp. 616–617.
- [28] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM computing surveys (CSUR) 46 (2014) 1–37.