International Journal of Pervasive Computing and Communications

From detection to repair: an interactive framework for reliable IoT data streams

Obaid Haylan B. Alotaibi
Department of Computer Science, Shaqra University, Shaqra, Saudi Arabia, and
Eric Pardede and Sarath Tomy
Department of Computer Science and IT, La Trobe University, Melbourne, Australia

Received 14 December 2025
Revised 20 March 2026
1 May 2026
8 June 2026
Accepted 13 July 2026

Abstract

Purpose – The purpose of this study is to address the challenges of repairing errors in real-time data streams generated by Internet of Things (IoT) devices, which benefit various sectors including healthcare, business, and industry. Although these data streams are valuable, IoT readings often contain errors that lead to unreliable analysis and flawed decisions. Traditional IoT data repairing techniques rely primarily on batch processing methods, such as rule-based filtering, which introduce latency and cannot effectively handle real-time streaming data. Furthermore, these conventional approaches typically remove all anomalies without identifying their underlying causes, which can result in the loss of critical insights. Compounding these issues, the computational demands of real-time processing present significant obstacles, and the dynamic nature of data streams makes anomaly repair especially difficult.

Design/methodology/approach – The study proposes a real-time anomaly repair model for structured IoT data streams. The model classifies and repairs detected anomalies automatically. The proposed model uses statistical measurements and machine learning techniques to repair anomalies. The authors evaluated the model using four data sets, demonstrating improved data quality in real-time data streams by correctly assigning repair techniques to the detected and classified anomalies.

Findings – By eliminating manual intervention and triggering repair only upon anomaly detection, the proposed method reduces decision delays and avoids unnecessary computational overhead, making it well-suited for efficiently handling anomalies in real-time data streams.

Originality/value – This research offers an automated framework to repair anomalies in real-time data streams by applying one of three actions: delete, keep or replace, selected according to anomaly classification and type. Also, it integrates a repair toolset that uses statistical measurements and machine learning techniques to provide multiple replacement options for anomaly repair.

Keywords Real-time data stream, Data cleaning, Context-awareness, Data repairing, Data anomaly, Machine learning, Healthcare

Paper type Research paper

1. Introduction

The Internet of Things (IoT) refers to physical devices that collect and exchange data via the internet. IoT technologies simplify everyday tasks and improve comfort in our daily lives (Alotaibi *et al.*, 2024). Various sectors, such as healthcare, business and industry, gain

Disclosure statement: No potential conflict of interest is reported by the authors.

International Journal of Pervasive Computing and Communications
Emerald Publishing Limited
1742-7371
DOI 10.1108/IJPCC-12-2025-0596

significant benefits from the large volumes of data that is generated by IoT in real time because real-time data visualisation enables prompt, informed decision-making. Real-time decision-making based on high data quality offers critical advantages, including saving patients' lives, reducing operational costs in business and improving industrial productivity and safety.

However, modern domains such as healthcare, business and industry increasingly depend on digital data to guide critical decisions. The importance of data quality has grown substantially over time because decisions based on poor data quality can have serious consequences. Conversely, high-quality data streams enable organisations to make more reliable decisions and fully realise the benefits of digital transformation. For example, in healthcare, hospitals rely on accurate patient data to provide high-quality care and ensure patient safety. Low-quality data can lead to misdiagnosis, inappropriate treatment plans or dosage miscalculations that endanger patients and compromise the quality of care. High-quality data, on the other hand, supports accurate diagnoses, an appropriate treatment plan and correct medication dosages, ultimately increasing patient safety and improving overall healthcare outcomes.

The difference between using raw data and cleaned data is the level of proactive prevention across these domains. Raw data leads to incorrect analysis results and inaccurate decision-making that negatively affects these domains, whereas cleaned data enables these domains to mitigate risks in real-time by making timely decisions based on accurate analysis results.

While the field of IoT anomaly detection is still in its early stages (Chatterjee and Ahmed, 2022), anomaly repair for streaming IoT data remains even less explored. Detecting anomalies in real time is particularly challenging and critical (Cook *et al.*, 2020); however, repairing anomalies in real-time data streams is equally essential, if not more critical. Several factors make anomaly detection and repair in real-time stream processing more challenging than in batch processing, as summarised in Table 1 (Alotaibi *et al.*, 2023).

Despite the significant benefits IoT brings to different sectors, organisations face mounting challenges in maintaining the data quality necessary for reliable decision-making. There is a direct correlation between data analysis and data cleaning; therefore, cleaner data will lead to increased quality in real-time data streams, resulting in more trusted and accurate results that will help businesses make better decisions based on data analysis. Hence, data cleaning plays a major role in increasing data quality.

Traditional IoT data cleaning techniques primarily rely on batch processing methods, such as rule-based filtering, which introduce latency and fail to effectively handle real-time streaming IoT data. Moreover, traditional repairing approaches often remove all anomalies

Table 1. Comparison of stream and batch processing characteristics

Aspect	Stream processing	Batch processing
Latency	Low latency	High latency
Complexity	More complex to implement and manage	Less complex to implement
Processing mode	Processed as it arrives	Processed after being collected
Error handling	Need to be identified and managed in real time	Can be identified and corrected before execution at any time
Fault tolerance	More difficult due to the real-time nature	Easier to achieve because the entire batch can be reprocessed
Use case example	Monitoring sensor data in IoT system	Monthly reports

without identifying their underlying causes, potentially leading to the loss of critical insights. The demand for computational resources is another challenge for anomaly detection in data streams, as real-time processing needs greater computing power to meet application requirements (Schneider and Xhafa, 2022). Similarly, anomaly repair in data streams encounters comparable computational demands, as it must perform the repair operations within the same time constraints.

Detecting anomalies in a real-time data stream is not enough to make a decision or to improve data quality. In this paper, we propose a real-time anomaly cleaning model for structured IoT data streams. To improve the data quality, the model aims to repair the detected anomalies based on their classification and address these anomalies automatically. We summarise our contributions in this paper as follows:

- An automated framework to repair anomalies by applying one of three actions: delete, keep or replace, selected according to anomaly classification and type.
- An integrated repair toolset that uses statistical measurements and machine learning techniques to provide multiple replacement options for anomaly repair.

As a result, these contributions provide a scalable, adaptive and intelligent framework that ensures IoT data streams are cleaned, which enables reliable and confident decision-making in real-time. This paper is structured as follows: Section 2 reviews previous work in anomaly repair. Section 3 describes our proposed method for repairing anomalies in real-time data streams. Section 4 presents a proof of concept for the proposed method, followed by evaluation in Section 5. Section 6 provides a practical implication of the proposed method. Finally, Section 7 summarises the paper.

2. Related works

Anomaly repair techniques are used to clean detected anomalies in the data. Thus, we divide this section into anomaly-repair techniques for stream processing and batch processing. Various techniques have been proposed to handle issues that might be detected in batch processes or streaming data, such as imputing missing values or removing anomalies.

2.1 Anomaly repair techniques on batch processing

Numerous studies have proposed deep learning approaches to address data cleaning issues such as Zhang *et al.* (2020), which improved density-based clustering of applications with noise density clustering to deal with duplicated data in batch processing. Authoritative data from the University of California were collected to examine the proposed method. The results of recovery rate, correct detection and accuracy with an increment of record number, the result of recovery rate was oscillated, where the correct detection result was increased and the accuracy results was decreased when the number of record increased. Fang (2022) proposed the diversity based sample selection achieved (SDUS) algorithm based on the correlation of redundant data to deal with missing and duplication in batch processing. The data set was gathered from IoT and the accuracy of the proposed algorithm was compared with the ST-SDC algorithm. The results showed that SDUS 96% accuracy whereas the ST-SDC has an average accuracy of 87%.

Ma *et al.* (2020) combined long short-term memory with bi-directional imputation and transfer learning to deal with missing values in batch processing, using energy consumption data from facilities and campus services at Cornell University. The results were compared to existing methods. Ding and Qin (2018) proposed an updating algorithm combining data cleaning and conjugate gradient to deal with missing values in batch processing. The data

used for the experiment were from the US Census 1990 and the Thyroid Medical data sets. The results compared precision and recall in the Thyroid Medical data set using a logical regression model and accuracy in the US census data set was compared to the updating algorithm by random sampling. The proposed algorithm showed better performance than logistic regression model. Rama Satish and Kavya (2018) proposed a hybrid algorithm by combining cuckoo search optimisation with the gravitational search algorithm to tackle missing values, duplicated data and outliers in batch processing. An employee data set was used to examine the performance of the proposed algorithm in terms of precision, recall, accuracy and f-measure and the results were compared with an existing method for cleaning data without optimisation.

2.2 Anomaly repair techniques on stream processing

Turabieh *et al.* (2019) proposed a dynamic adaptive network-based fuzzy inference system that combines a fuzzy system and artificial neural networks. The proposed method was used to impute missing values that can occur in data collected from IoMT. Fountas and Kolomvatsos (2020) proposed an ensemble correlation model based on cosine similarity and Mahalanobis distance to impute missing values in the data stream. Van Zoest *et al.* (2021) focused on outlier detection and imputing missing data from data streams generated from IoT devices by applying various methods. A method based on spatio-temporal classification was adopted to detect outliers, where the missing value was estimated using the maximum likelihood estimation method.

Andreoni Lopez *et al.* (2019) merged the normalisation algorithm and feature selection algorithm to deal with irrelevant data and they used Grupo de Teleinformática e Automação/ Federal University of Rio de Janeiro data set. the results were compared with principal component analysis, sequential feature selection, support vector machine recursive feature elimination and ReliefF methods in terms of accuracy and sensitivity of detection by using three classification methods namely, decision tree, support vector machine and neural network. The principal component analysis had majority of the highest results in all classifications in terms of accuracy and sensitivity of detection. Zhao *et al.* (2022) proposed variational-based imputation for multi-modal time series. For better convergence, a two-stage isolated optimisation strategy was also proposed.

Jehlol and George (2022) proposed a method based on Pearson correlation between histograms of various data extensions to remove duplicated data. 3DLDF files and SQLite files were used for the evaluation and the results were compared to the two thresholds two divisors and basic sliding window methods in terms of data size after removing duplicated data and percentage of removing duplicated data. The proposed method had better results than other methods that were compared with it in both experiments. You *et al.* (2018) dealt with duplication and irrelevant data by proposing the online streaming feature selection window algorithm with a sliding window strategy. For the evaluation, 14 data sets were used and the accuracy and run time were compared with Alpha-investing, online streaming feature selection and scalable and accurate online approach algorithms. Kumar *et al.* (2013) detected and repair sensor drift by proposing a framework that uses spatial Kriging and Kalman filtering for IoT wireless sensor networks, demonstrating superiority in detecting and correcting both linear and nonlinear drifts in real time.

In relation to repair techniques, previous studies proposed various methods to impute missing values without discussing which technique was used to detect the missing values. Van Zoest *et al.* (2021) considered outlier detection in addition to imputing missing values, but their approach does not repair the detected extreme value. To the best of our knowledge, previous works generally do not include data quality assessment, data analysis or using

context during the process. The previous work has typically focused on a single aspect of data repair and does not address the variety of data issues that can be occur in real-time streams, such as extreme value, sensor drift and duplicate data. Therefore, there is a need for a comprehensive automated anomaly repair technique capable of handling diverse data issues as they are detected.

Beyond the methods reviewed above, online multivariate time-series imputation approaches such as Kalman filtering, robust seasonal-trend decomposition and multiple imputation by chained equations provide conceptual baselines for replacement-based repair in streaming contexts, though these methods focus on imputation rather than classification-aware repair.

To the best of our knowledge, no existing work in the stream processing literature specifically addresses the repair of other anomaly types, such as stuck values and extreme values, as existing works focus primarily on detection rather than repair. This gap highlights the need for a comprehensive automated repair framework capable of handling various types of anomaly in real time, which is the focus of the proposed work.

The proposed framework differs from the existing approaches in three key ways. Firstly, unlike streaming imputation-only methods, which focus solely on replacing missing values without considering anomaly classification, the proposed framework first classifies each detected anomaly before determining the appropriate repair action, keeping, deleting or replacing, based on anomaly type and classification. Secondly, unlike rule-based stream filters that indiscriminately remove all flagged anomalies, the proposed framework preserves anomalies classified as valid events, preventing the loss of critical insights in high-stakes domains such as healthcare. Thirdly, the proposed framework explicitly addresses the trade-off between latency and repair correctness in real-time streams by applying lightweight statistical and machine learning techniques within a sliding window, ensuring that repair operations are completed within the time constraints of the streaming pipeline without sacrificing repair accuracy.

3. Method

Repairing detected anomalies is a critical task, as there are three possible actions for handling anomalies in real-time data streams: remove, keep or replace the anomalous values (Alotaibi *et al.*, 2023; Rollo *et al.*, 2023). We follow a process-based approach that systematically handles each detected anomaly through a structured decision-making pipeline. Our methodology applies logic-based rules to determine the appropriate repair action for each classified anomaly. Our repair process uses a sliding window mechanism. We consider several types of anomalies that may occur in real-time data streams. In our methodology, we assume that these anomalies have already been detected and we subsequently classify them as errors, events or uncertain, as shown in Figure 1. Any anomaly type not covered by Figure 1 is treated as an undefined and classified as uncertain.

In the proposed framework, anomalies are assumed to have already been detected by an external detection prior to repair. The detection component identifies anomalies based on statistical thresholds and contextual rules defined by the user. It is important to note that detection errors may influence repair outcomes. Specifically, false positives may trigger unnecessary repair actions on valid data points, while false negatives allow genuine anomalies to pass through the pipeline unrepaired, potentially affecting overall data quality. In the proposed framework, context refers to the user-defined configuration established prior to initiating the data stream. This context is used by the detection stage to identify anomalies and by the repair stage to validate replacement values. It is important to acknowledge that the delete on error policy carries a misclassification risk. If an anomaly is incorrectly classified as

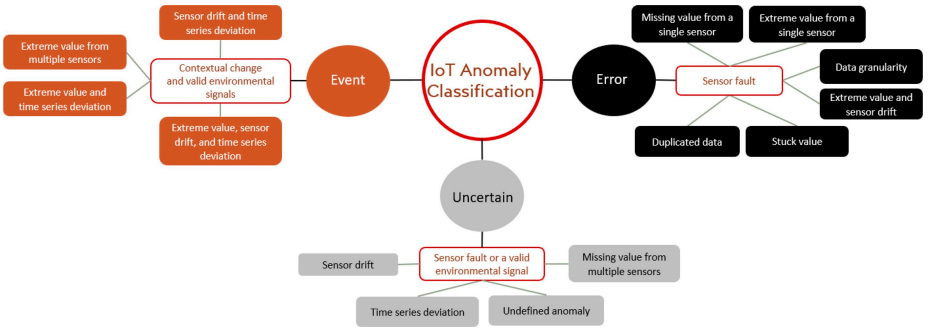

Figure 1. A summary of anomaly classification

an error, valid data points may be permanently removed. The current framework does not incorporate confidence thresholds or rollback mechanisms to mitigate this risk. Incorporating such safeguards, including human in the loop checkpoints for high stakes decisions, is identified as a direction for future work.

Anomalies such as duplicate data, granularity issues, missing values from one sensor, stuck values, extreme values from a single sensor (sudden spikes) and extreme values accompanied by drift are classified as errors because these types clearly indicate sensor faults. Anomalies categorised as real events include extreme values confirmed by other sensors, extreme values with deviation and drift with deviation because they represent contextual changes and valid environmental signals rather than sensor malfunctions. Anomalies involving missing values confirmed by other sensors, isolated drift, isolated deviation, undefined anomalies or extreme values exhibiting both drift and deviation are categorised as uncertain because they could indicate either sensor faults or valid signals.

As time is a critical factor when anomalies are classified, the system repairs anomalies automatically. Depending on the anomaly classification, the system decides whether to keep, replace or remove the detected anomaly. Keeping an anomaly means taking no action and allowing the streaming data to remain unchanged. When the anomalies are classified as events, the system automatically keeps them, as these anomalies may represent a valid environmental signal or contextual change. Deleting anomaly data means removing it from the streaming context. Anomalies classified as errors are removed automatically because they typically result from sensor faults.

Replacing the anomaly means substituting it with an appropriate value. The system provides both statistical measurement techniques and machine learning methods to repair anomalies classified as uncertain. For statistical measurements, the arithmetic average is used to replace the missing values from multiple sensors, whereas the middle value is used to replace anomalies labelled as undefined anomalies, because this technique is resistant to outliers. Algorithms 1 and 2 implement these replacement strategies using the arithmetic average or the middle value. The arithmetic average is computed using the formula:

$$\text{arithmetic average} = \frac{\sum x_i}{n}$$

where $\sum x_i$ is the sum of normal observations and n is the number of observations.

The middle value is computed as follows. If the sliding window contains an odd number of normal observations:

$$\text{middle value} = \left(\frac{n+1}{2}\right)^{th}$$

If the sliding window contains an even number of normal observations:

$$\text{middle value} = \frac{1}{2}\left[\left(\frac{n}{2}\right)^{th} + \left(\frac{n}{2}+1\right)^{th}\right]$$

Algorithm 1. Replace anomaly with arithmetic average

```
Require: sensor_id, field, anomalous_value
Ensure: Replace anomaly with arithmetic average
normal_data ← self.normal_data_for_repair(sensor_id, field)
if normal_data.length ≥ 1 then
return arithmetic average(normal_data)
end if
return anomaly_data
```

Algorithm 2. Replace anomaly with middle value

```
Require: sensor_id, field, anomalous_value
Ensure: Replace anomaly with middle value
normal_data ← self.normal_data_for_repair(sensor_id, field)
if normal_data.length ≥ 1 then
return middle value(normal_data)
end if
return anomaly_data
```

For machine learning techniques, we use a KNN imputer to replace anomalies caused by sensor drift or time-series deviation. The KNN imputer operates as a univariate temporal nearest-neighbour method within the current sliding window. When an anomaly occurs in a target feature, the imputer identifies the K most temporally similar observations of that feature within the sliding window and computes their average to replace the anomalous value. The KNN imputer is configured with a maximum of $K = 5$ neighbours, selected based on empirical observation as a balance between repair stability and computational efficiency. When the number of available normal data points in the sliding window is fewer than 5, the number of neighbours is automatically reduced to match the available data. If fewer than 5 normal observations are available in total, the KNN imputer falls back to the middle value technique to ensure a valid replacement is always produced. This fallback strategy prevents repair failures in the early stages of the data stream, where insufficient historical data may exist to support KNN-based imputation.

The rules and logic used to repair detected anomalies in real-time IoT data streams are summarised in Algorithm 4, while Appendix provides the full set of rules and logic used in the repair process.

Algorithm 3. Repair anomaly with KNN imputation

```
Require: sensor_id, field, anomalous_value
Ensure: Replace anomaly with KNN imputation
normal_data ← self.normal_data_for_repair(sensor_id, field)
if len(normal_data) ≥ 5 then
    data ← normal_data + [np.nan]
    data_array ← np.array(data).reshape(−1, 1)
    imputer ← KNNImputer(n_neighbors = min(5, len(normal_data)))
    imputed ← imputer.fit_transform(data_array)
    return imputed[−1][0]
else
    return self.repair_with_median(sensor_id, field, anomalous_value)
end if
```

Algorithm 4. Automated decision for classified anomaly

```
Require: anomaly_classification, anomaly_type
Ensure: Return action and technique for anomaly handling
if anomaly_classification == "Error" then
    return {action : "delete", technique: None}
else if anomaly_classification == "Event" then
    return{action: "keep", technique: None}
else
    if anomaly_type ∈ {"Drift", "Deviation"}then
        return{action: "replace", technique: "knn"}
    else if anomaly_type == "Undefined anomaly" then
        return{action: "replace", technique: "middle value"}
    else
        return{action: "replace", technique: "arithmetic average"}
    end if
end if
```

The replacement techniques operate on the current sliding window; therefore, the repaired values may change over time, as each replacement is computed using only the most recent observations. Figure 2 illustrates the anomaly repairing process in real-time data streams.

By incorporating context-awareness, we propose a comprehensive pipeline for real-time anomaly repairing in structured IoT data streams. This pipeline is designed to support real-time decision-making by improving repair accuracy while reducing both processing time and

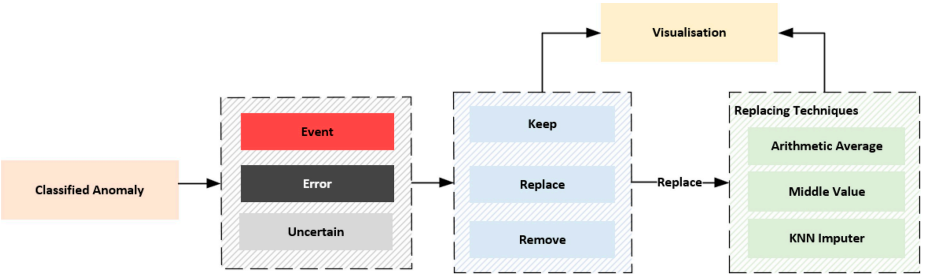

Figure 2. A comprehensive real-time anomaly repairing framework

computational overhead. The effectiveness of the proposed method is demonstrated in the implementation section that follows.

4. Implementation

4.1 Experiment setup

The experiments were conducted on a Windows 10 machine equipped with an i7 8th Gen CPU @ 3.20 GHz, 32 GB of RAM. We assume that the user has defined the context, and we apply a sliding window size of 20 to the streaming data, enabling anomaly repair in real time. A sliding window size of 20 was chosen based on empirical observations across the four data sets used in this study. The window size primarily influences the replacement-based repair techniques, namely, arithmetic average, middle value and KNN imputation, as these techniques rely on the number of recent data points available in the window to compute replacement values. For the keep and delete action, the window size has no effect on the repair outcome. A window of 20 observations was found to provide a sufficient number of recent data points to produce stable and accurate repair estimates while remaining computationally efficient for real-time processing. Windows smaller than 20 were found to produce unstable replacement values due to insufficient contextual data, particularly in the early stages of the data stream. Therefore, the window size of 20 represents a practical trade-off between repair quality and processing efficiency in real-time IoT data streams.

The proposed method was implemented as a user interface and the following section illustrates the methodology and process using a scenario.

To demonstrate the effectiveness of the proposed methodology, we used four data sets collected from IoT devices (Health Monitoring System, 2020; Health Monitoring, 2021; Patient Health Monitoring Data, 2020; Indoor Environment Data, 2020): three from health monitoring systems and one from indoor environments. Following the methodology described in (Ubani *et al.*, 2023), we expanded the data sets to 5,100 records in total. As a proof of concept, the experiment was stopped after 428 records had been streamed, at which point the framework had demonstrated correct and consistent behaviour. We assume that the user has defined the context. The proposed method was implemented as a user interface using the Streamlit library in Python.

4.2 Use case in healthcare: monitoring patients' vital signs

To demonstrate the implementation, we present a healthcare scenario that shows the implementation of anomaly repair in real-time data streams. For example, a clinician monitors a patient's vital signs, which are collected in real time via IoT devices. The clinician has already configured the context and entered the data link to initiate the streaming process.

Once initiated, the system presents the current context in a statistical table that displays the minimum, maximum, average and current value of each field in real time. Data quality is calculated once the model is trained. The system displays incoming data as a line chart and shows the cumulative record count from stream initiation to the current moment. When the model is trained, incoming real-time data first passes through the detection stage and the clinician sees the number of anomalies detected and their classification (event, error or uncertain), as shown in Figure 3.

When an anomaly is detected, the system automatically applies the appropriate repair technique based on anomaly classification. If the anomaly is classified as uncertain, then the system determines which repair technique to apply according to the anomaly type. If it is classified as an error, the system will remove it. Conversely, if the anomaly is classified as an event, the system retains it. After each repair action is applied, the repair table is updated accordingly and the line chart reflects only the cleaned data streams after repair techniques have been executed. The data quality percentage is displayed to the clinician in the detection table before the repair process and again in the repair table after the repair technique has been applied.

Since the impact of anomaly repair should be evaluated not only functionally but also quantitatively, we assessed data quality before and after the repair process. There are multiple dimensions for computing data quality and each dimension refers to a specific aspect of data quality (Ehrlinger and Wöß, 2022). To compute the data quality, we used the following dimensions, each assigned an equal weight of 25%:

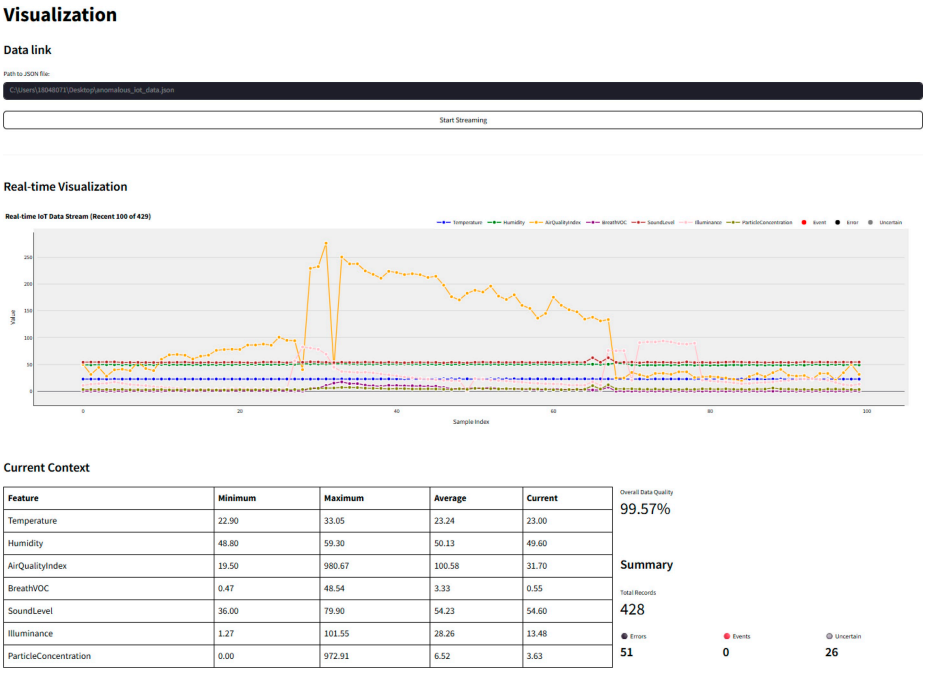

Figure 3. Anomaly repair user interface

$$\text{Accuracy} = 1 - \frac{\text{Data units in error}}{\text{Total data units}} \quad (\text{Lee } et\ al.,\ 2006) \tag{1}$$

$$DQ_{\text{dupl}} = 1 - \frac{\text{Repeated Timestamps}}{\text{Collected Values}} \quad (\text{Buelvas } et\ al.,\ 2023) \tag{2}$$

$$\text{Timeliness} = \max\left(0,\ 1 - \frac{\text{number of data granularity}}{\text{total\_records}}\right) \times 100 \tag{3}$$

$$\text{Completeness} = 1 - \frac{\text{Incomplete elements}}{\text{Total elements}} \quad (\text{Lee } et\ al.,\ 2006) \tag{4}$$

For the timeliness equation, we use the formulation defined by Scholl *et al.* (2023), shown as follow:

$$\text{Timeliness} = t_i - t_{i-1} \tag{5}$$

In this equation, timeliness is expressed as the difference between consecutive timestamps. Based on this definition, we calculated the timeliness percentage using equation (3), where the number of data granularity represents the count of intervals that deviate from the expected periodic rate (detected using equation (5)) and the total records refers to the total number of consecutive timestamp pairs analysed. We use the max operator in equation (3) to ensure that the resulting value remains within a valid range and that the timelines score never falls below zero. Figure 4 shows the data quality before and after applying the repair actions using the equations above. Figures 3 and 4 are displayed on a single screen, allowing the clinician to simultaneously observe the detected anomalies and the data quality before and after the repair process.

While the overall data quality is computed using the formulas proposed by Alotaibi *et al.* (2024):

$$\text{Average Data Quality} = \frac{DQ_{\text{detection}} + DQ_{\text{repair}}}{2} \tag{6}$$

As the data stream continues, the clinician continues monitoring the patient's vital signs in real time. At a later stage, the model detects an extreme value from a single sensor and again classifies it as an error. In response, the system removes this anomaly, as it is a

Detected Classifications (per feature)

Feature	Errors	Events	Uncertain
Temperature	7	0	2
Humidity	7	0	3
AirQualityIndex	8	0	4
BreathVOC	7	0	3
SoundLevel	8	0	5
Illuminance	7	0	4
ParticleConcentration	7	0	5
Overall Data Quality	**99.4%**		

Repair Actions

Feature	Removed	Replaced	Kept
Temperature	7	2	0
Humidity	7	3	0
AirQualityIndex	8	4	0
BreathVOC	7	3	0
SoundLevel	8	5	0
Illuminance	7	4	0
ParticleConcentration	7	5	0
Overall Data Quality	**100.0%**		

Figure 4. Data quality measurement before and after the repair process

single erroneous point. This ensures that the clinician views a smoother and more accurate chart of the patient's vital signs, enabling immediate intervention when necessary, ultimately saving the patient's life through timely access to high-quality data. This scenario highlights the critical importance of repairing detected anomalies. Effective anomaly repair supports timely and informed decision-making and demonstrates the essential role of real-time data cleaning in high-stakes environments such as healthcare.

5. Evaluation and discussion

To validate the robustness of the proposed framework, we used two evaluation methods. Firstly, we assessed the framework's performance using data quality measurements across four data sets collected from IoT devices (Health Monitoring System, 2020; Health Monitoring, 2021; Patient Health Monitoring Data, 2020; Indoor Environment Data, 2020). A total of 428 records were processed during the proof of concept experiment, representing the portion of the 5,100 expanded records streamed before the experiment was stopped. Artificial anomalies of four types, duplicate data, granularity issues, missing values and extreme values, were inserted into this portion to validate the framework. Secondly, we conducted functionality testing by examining whether the repair techniques were correctly mapped to the corresponding detected anomalies in real-time data streams. While it is technically possible to evaluate latency, throughput and resource consumption during anomaly repair, such metrics are not meaningful in our context because anomaly repair is not executed continuously for every incoming record. Instead, repair is triggered only when an anomaly occurs. For this reason, our evaluation focuses on data-quality improvement and functionality testing rather than system-level performance metrics. It should be noted that computational efficiency, as referenced in this paper, pertains to the event-driven nature of the repair mechanism and the elimination of manual intervention, rather than formally benchmarked throughput or latency. Since the framework simulates a real world streaming scenario in which anomaly occurrence is not known in advance, repair time cannot be measured independently of the pipeline, as the repair step is triggered automatically and immediately upon detection within the stream. A formal per-operation microbenchmark reporting average repair time for each technique is identified as a direction for future work.

For performance evaluation, we compared data quality metrics before and after applying the repair techniques. As shown in Figure 4, the data quality before repairing was 99.4%. After the repair techniques were applied, the data quality improved to 100%. The repair table reflects 100% data quality because detected anomalies are either removed or replaced based on their classification. Consequently, when the above formulas [equations (1)–(6)] are applied, the results show 100% data quality, as no anomalies remain. However, if the anomaly is classified as an event, it is kept in the data stream and the data quality will change accordingly. Overall, the improvement in data quality confirms that the proposed method is both effective and valid.

For functionality testing, we examined the proposed method with different scenarios, such as keeping anomalies, deleting anomalies and replacing anomalies with various replacement techniques. We also evaluated the accuracy of assigning repair techniques to detected anomaly types based on their classification. The proposed method achieved 100% accuracy for all these scenarios. These results confirm that anomaly repair techniques are correctly assigned to detected anomalies, thereby enhancing the quality of real-time data streams.

The focus of our experiment was to automate the cleaning process in real-time and to apply appropriate repair techniques according to the current state of the data streams. The

proposed method uses various data cleaning strategies for different anomaly types based on their classification. The proposed approach enables users to make informed real-time decisions with confidence based on high data visualisation. Both evaluation methods, performance assessment and functionality testing, demonstrate the effectiveness and correctness of the proposed method. A quantitative fidelity assessment comparing repaired values against pre-injection ground truth values, such as mean absolute error or root mean square error, would provide additional evidence of reconstruction quality and is identified as a direction for future work.

6. Practical implications

The importance of data quality has grown considerably across various domains. The proposed framework provides a useful tool for improving data quality in real-time data streams through automated repair mechanisms and continuous quality assessment, enabling better decisions across multiple smart environments. We focus on three key application domains: smart healthcare, smart factories and smart cities.

In smart healthcare environments, the proposed framework enhances data quality through automated, real-time repair without requiring manual intervention. Once anomalies in patient vital signs are detected and classified, the framework's automated repair mechanism determines the appropriate action, whether to keep, replace or remove the anomalous values. When a heart rate reading is classified as a sensor error, the system automatically removes it, maintaining data integrity without requiring clinical staff to manually intervene. Conversely, when vital sign anomalies are classified as events, the system preserves them, ensuring that critical medical changes reach clinicians immediately. This automated repair approach eliminates the need for manual data quality interventions, allowing clinical staff to focus on patient care while ensuring diagnostic accuracy.

In smart manufacturing environments, the framework's automated data quality enhancement capabilities directly impact operational efficiency and product quality. Once equipment sensor anomalies are detected and classified, the system automatically applies the appropriate repair action. This reduces the need for engineers to manually review large volumes of sensor readings while ensuring that critical faults are identified promptly. Automated data quality enhancement reduces operational costs by eliminating unnecessary maintenance triggered by false positives. The system's ability to repair anomalies in real-time prevents catastrophic equipment failures that can cost millions in repairs and lost production, while the automated nature of the framework ensures these benefits are achieved without additional labour costs for data quality management.

In smart city environments, the framework enables real-time anomaly repair in weather monitoring systems, environmental sensors and surveillance infrastructure without human intervention. The automated repair mechanism continuously assesses data quality and applies appropriate actions based on anomaly classification. When sudden weather events occur, such as flash floods, the system preserves them, enabling immediate public alerts for real threats while avoiding false alarms that erode public trust and waste emergency response resources. Smart city utilities benefit from automated data repair in water distribution and waste management systems. The framework automatically repairs anomalies classified as sensor errors, preventing unnecessary maintenance dispatches and ensuring that genuine faults are not masked.

While the framework was evaluated using healthcare and indoor environment data sets, the classification to action mapping and sliding window mechanism are domain agnostic by design. The framework can be extended to additional anomaly types by updating the

classification rules and repair mappings and can be applied to other domains such as smart agriculture, transportation and energy management, provided that appropriate context is defined by the user.

The framework can also be integrated with existing streaming platforms such as Apache Kafka or Apache Flink by embedding the repair module as a processing layer within the data pipeline. Similarly, it can complement existing data quality monitoring tools by providing automated repair capabilities alongside continuous quality assessment.

It should be noted that the practical implications described in this section are based on experimental evaluation using simulated real-time streams. Real world deployment may introduce additional challenges such as network latency, sensor heterogeneity and varying anomaly distributions that may affect framework performance. Further evaluation in live deployment settings is therefore recommended to fully validate these implications. In high-stakes deployments, the risk of over-cleaning should also be considered specifically, the possibility that rare but genuine events may be misclassified as errors and permanently removed from the stream. The absence of uncertainty quantification and human in the loop verification check points are open considerations for real world deployment and addressing these limitations is identified as a direction for future work.

7. Conclusion

This paper has demonstrated the significance of anomaly repair in real-time data streams and exposed the limitations of traditional techniques in addressing this challenge. We proposed a method that repairs the detected anomalies in real-time data streams by applying a range of data cleaning techniques and data quality assessment. The goal is to support informed, confident decision-making in dynamic environments through clear and intuitive real-time data quality management. This method contributes to improving the user's decision-making accuracy in real-time environments.

As a proof of concept, we conducted experiments to validate the effectiveness of our proposed method using healthcare data. We used three healthcare data sets and an indoor environments data set to demonstrate the applicability of the proposed framework. Furthermore, we discussed the practical implications of the proposed method across several sectors, including smart healthcare, smart manufacturing and smart city.

By automating the repair process and triggering corrective actions only upon anomaly detection, the proposed method eliminates manual intervention delays and avoids unnecessary computational overhead, making it well-suited for efficiently handling anomalies in real-time data streams. In future work, we aim to extend the evaluation to additional sectors to further validate the generalisability and robustness of the proposed framework.

Funding

This research received no external funding.

Data availability

The original data presented in the study are openly available: https://thingspeak.mathworks.com/channels/1041003, https://thingspeak.mathworks.com/channels/1490300, https://thingspeak.mathworks.com/channels/940553, https://thingspeak.mathworks.com/channels/1014791

References

Alotaibi, O., Pardede, E. and Tomy, S. (2023), "Cleaning big data streams: a systematic literature review", *Technologies*, ISSN: 2227-7080, Vol. 11 No. 4, p. 101, doi: 10.3390/technologies11040101, available at: www.mdpi.com/2227-7080/11/4/101

Alotaibi, O., Tomy, S. and Pardede, E. (2024), "A framework for cleaning streaming data in healthcare: a context and User-Supported approach", *Computers*, ISSN: 2073-431X, Vol. 13 No. 7, p. 175.

Andreoni Lopez, M., *et al*., (2019), "A fast unsupervised preprocessing method for network monitoring", *Annals of Telecommunications*, Vol. 74 Nos 3-4, pp. 139-155, doi: 10.1007/s12243-018- 0663- 2, available at: www.scopus.com/inward/record.uri?eid=2-s2.0-85053375010&doi=10.1007%2fs12243-018-0663-2&partnerID=40&md5=c133a89b683a5840c269366aa95004a2

Chatterjee, A. and Ahmed, B.S. (2022), "IoT anomaly detection methods and applications: a survey", *Internet of Things*, ISSN: 2542-6605, Vol. 19, p. 100568, doi: 10.1016/j.iot.2022.100568.

Cook, A.A., Misirli, Göksel. and Fan, Z. (2020), "Anomaly detection for IoT time-series data: a survey", *IEEE Internet of Things Journal*, ISSN: 2327-4662, Vol. 7 No. 7, pp. 6481-6494, doi: 10.1109/JIOT.2019.2958185.

Ding, X. and Qin, S. (2018), "Iteratively modeling based cleansing interactively samples of big data", *Cloud Computing and Security*, Vol. 11063, pp. 601-612, doi: 10.1007/978-3-030-00006-6_55. LNCS, available at: www.scopus.com/inward/record.uri?eid=s2.0-85061261224&doi1007%2f978-3-030-00006-6_55&partnerID=md57f8127424933e1a29137758b55c4705

Ehrlinger, L. and Wöß, W. (2022), "A survey of data quality measurement and monitoring tools", *English Frontiers in Big Data*, ISSN: 2624-909X, Vol. 5, doi: 10.3389/fdata.2022.850611, available at: www.frontiersin.org/articles/10.3389/fdata.2022.850611

Fang, J. (2022), "Research on automatic cleaning algorithm of multi-dimensional network redundant data based on big data", *Evolutionary Intelligence*, Vol. 15 No. 4, pp. 2609-2617, doi: 10.1007/s12065-021-00620-y, available at: www.scopus.com/inward/record.uri?eid=2-s2.0-85107361782&doi=10.1007%2fs12065-021-00620-y&partnerID=40&md5=3a78c88cc595647b5df8ba65145dc1b6

Fountas, P. and Kolomvatsos, K. (2020), "A continuous data imputation mechanism based on streams correlation", *2020 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1-6, doi: 10.1109/ISCC50000.2020.9219548.

Health Monitoring (2021), "Health monitoring data", available at: https://thingspeak.mathworks.com/channels/1490300

Health Monitoring System (2020), "Health monitoring system data", available at: https://thingspeak.mathworks.com/channels/1041003

Indoor Environment Data (2020), "Indoor environment data", available at: https://thingspeak.mathworks.com/channels/1014791

Jehlol, H.B. and George, L.E. (2022), "Big data De-duplication using classification scheme based on histogram of file stream", *2022 International Conference on Intelligent Technology, System and Service for Internet of Everything (ITSS-IoE)*, pp. 1-7, doi: 10.1109/ITSS-IoE56359.2022.9990942.

Lee, Y.W., *et al*. (2006), *Journey to Data Quality*, The MIT Press, Cambridge, MA, ISBN: 9780262256544, doi: 10.7551/mitpress/4037.001.0001.

Kumar, D., Rajasegarar, S. and Palaniswami, M. (2013), "Automatic sensor drift detection and correction using spatial Kriging and Kalman filtering", *2013 IEEE International Conference on Distributed Computing in Sensor Systems*, pp. 183-190, doi: 10.1109/DCOSS.2013.52.

Ma, J., *et al*., (2020), "A bi-directional missing data imputation scheme based on LSTM and transfer learning for building energy data", *Energy and Buildings*, Vol. 216, p. 109941, doi: 10.1016/j.enbuild.2020.109941, available at: www.sciencedirect.com/science/article/pii/S0378778819333717

Patient Health Monitoring Data (2020), "Patient health monitoring data", available at: https://thingspeak.mathworks.com/channels/940553

Rama Satish, K.V. and Kavya, N.P. (2018), "Hybrid optimization in big data: error detection and data repairing by big data cleaning using CSO-GSA", *Cognitive Computing and Information Processing*, Vol. 801, pp. 258-273, doi: 10.1007/978-981-10-9059-2_24, available at: www.scopus.com/inward/record.uri?eid=2-s2.0-85045314251&doi=10.1007%2f978-981-10-9059-2_24&partnerID=40&md5=538b46e09bec8cae5a0e33f21486208d

Rollo, F., Bachechi, C. and Po, L. (2023), "Anomaly detection and repairing for improving air quality monitoring", *Sensors*, ISSN: 1424-8220, Vol. 23 No. 2, doi: 10.3390/s23020640, available at: www.mdpi.com/1424-8220/23/2/640

Schneider, P. and Xhafa, F. (2022), *Anomaly Detection and Complex Event Processing over Iot Data Streams: With Application to EHealth and Patient Data Monitoring*, Academic Press, London, ISBN: 0128238194.

Scholl, C., *et al*. (2023), "An integrated framework for data quality fusion in embedded sensor systems", *Sensors*, ISSN: 1424-8220, Vol. 23 No. 8, doi: 10.3390/s23083798, available at: www.mdpi.com/1424-8220/23/8/3798

Turabieh, H., Mafarja, M. and Mirjalili, S. (2019), "Dynamic adaptive Network- Based fuzzy inference system (D-ANFIS) for the imputation of missing data for internet of medical things applications", *IEEE Internet of Things Journal*, Vol. 6 No. 6, pp. 9316-9325, doi: 10.1109/JIOT.2019.2926321, available at: www.scopus.com/inward/record.uri?eid=2-s2.0-85076778114&doi=10.1109%2fJIOT.2019.2926321&partnerID=40&md5=3b5bd4c0960ab89bab923611e47f67d6

Ubani, S., Polat, S., Olcay and Nielsen, R. (2023), "Zeroshotdataaug: generating and augmenting training data with ChatGPT", arXiv Preprint, *arXiv:2304.14334*.

Van Zoest, V., Liu, X., Ngai, E (2021), "Data quality evaluation, outlier detection and missing data imputation methods for IoT in smart cities", *Machine Intelligence and Data Analytics for Sustainable Future Smart Cities*, Ghosh, U. *et al*. (Eds), Springer International Publishing, Cham, ISBN: 978-3-030-72065-0, pp. 1-18, doi: 10.1007/978-3-030-72065-0_1.

You, D., *et al*. (2018), "Online feature selection for streaming features with high redundancy using Sliding-Window sampling", *2018 IEEE International Conference on Big Knowledge (ICBK)*, pp. 205-212, doi: 10.1109/ICBK.2018.00035.

Zhang, X., Lin, R. and Xu, H. (2020), "An adaptive parameters density cluster algorithm for data cleaning in big data", *Artificial Intelligence and Security*, LNCS, Vol. 12239, pp. 543-553, doi: 10.1007/978-3-030-57884-8_48, available at: www.scopus.com/inward/record.uri?eid=2-s2.0-85091303046&doi=10.1007%2f978-3-030-57884-8_48&partnerID=40&md5=06ec31f6cd114dbfe8b0e883aec835f0

Zhao, X., *et al*. (2022), "VIMTS: variational-based imputation for multi-modal time series", *2022 IEEE International Conference on Big Data (Big Data)*, pp. 349-358, doi: 10.1109/BigData55660.2022.10020834.

Corresponding author
Eric Pardede can be contacted at: e.pardede@latrobe.edu.au

Appendix

Algorithm 5. `Anomalies repairing`

Require: stream data with detected and classified anomalies
Ensure: cleaned data stream

```
cleaned_data ← []
for each new_data in stream_data do
  sensor_id ← new_data.sensor_id
  timestamp ← parse timestamp from new_data.created_at
  for each sens_id in history do
    history[sens_id] ← [record for each record in history[sens_id]
      if (timestamp − record.timestamp) ≤ 2 minutes]
  end for
  anomaly_type_dict ← new_data.anomaly_type
  anomaly_classification_dict ← new_data.anomaly_classification
  record_modified ← False
  if anomaly_type_dict.timestamp ∈ {Duplicated, Granularity} then
    new_data.delete_record ← True
    anomaly_type ← anomaly_type_dict.timestamp
    for each field in new_data do
      if field ∉ EXCLUDED_FIELDS and field ≠ delete_record_flag then
        repair_tracker.append({
          field: field, action: delete,
          anomaly_type: anomaly_type, reason: timestamp anomaly
        })
      end if
    end for
    Append new_data to cleaned_data
    Append new_data to history[sensor_id]
    continue
  end if
  for each field in new_data do
    if field ∈ EXCLUDED_FIELDS then
      continue
    end if
    if field ∉ anomaly_type_dict then
      if new_data[field] is a valid numeric value then
        v ← numeric value of new_data[field]
        if v is not undefined then
          Update normal data window(sensor_id, field, v, timestamp)
        end if
      end if
      continue
    end if
    anomaly_type ← anomaly_type_dict[field], or Undefined anomaly if nc
    anomaly_classification ← anomaly_classification_dict[field], or Uncerta
    if new_data[field] is a valid numeric value then
      current_value ← numeric value of new_data[field]
    else
      current_value ← raw value of new_data[field]
    end if
    original_value ← current_value
    decision ← get_automated_decision(anomaly_classification, anomaly_t
    action ← decision.action
    if action = delete then
```

```
      repair_tracker.append({
       field: field, action: delete,
       anomaly_type: anomaly_type, original_value: original_value
      })
      Remove field from new_data
      record_modified ← True
     else if action = keep then
      repair_tracker.append({
       field: field, action: keep,
       anomaly_type: anomaly_type, original_value: original_value
      })
     else if action = replace then
      technique ← decision.technique
      if technique ∈ available repair techniques then
       num_value ← undefined
       if current_value is a valid numeric value then
        num_value ← numeric value of current_value
       end if
       repaired_value ← apply technique(sensor_id, field, num_value)
       if num_value is valid and repaired_value is numeric
        and repaired_value = num_value then
        No update required
       else
        new_data[field] ← repaired_value
        record_modified ← True
        repair_tracker.append({
         field: field, action: replace,
         anomaly_type: anomaly_type,
         original_value: original_value, repaired_value: repaired_value
        })
        if repaired_value is a valid numeric value then
         v ← numeric value of repaired_value
         if v is not undefined then
          Update normal data window(sensor_id, field, v, timestamp)
         end if
        end if
       end if
      end if
     end if
    end for
    if record_modified then
     Mark new_data as repaired
    end if
    Append new_data to cleaned_data
    Append new_data to history[sensor_id]
end for
return cleaned_data
```