Tracking Stream Quality Issues in Combined Physical and Radar Sensors for IoT-based Data-driven Actuation

Oluwaseun Bamgboye
School of Computing
Edinburgh Napier University
Edinburgh,UK
O.Bamgboye@napier.ac.uk

Xiaodong Liu
School of Computing
Edinburgh Napier University
Edinburgh,UK
X.Liu@napier.ac.uk

Peter Cruickshank
School of Computing
Edinburgh Napier University
Edinburgh,UK
P.Cruickshank@napier.ac.uk

Qi Liu
Nanjing University of Information Sc & Tech
Nanjing, 210044, China
qi.liu@nuist.edu.cn

Yonghong Zhang
Nanjing University of Information Sc & Tech
Nanjing, 210044, China
zyh@nuist.edu.cn

***Abstract*—In this paper, a stream quality tracking for measurements from combined radar and physical sensors is developed. The authors proposed the use of RDF stream processing system and semantic rules to provide semantic reasoning for tracking erroneous data points from real time sensor readings. We demonstrated the effectiveness and the efficiency of the approach using the dataset produced from smart home project. Experiments were conducted with simulated arbitrary sensor values for inconsistent and missing sensor readings at various data points. The results was able to show the feasibility of the approach and that quality requirement of sensor streams can be verified at lower granularity level (data layer) of smart actuators.**

***Index Terms*—Actuator, C-SPARQL, Radar, IoT, Sensor, Stream Quality**

I. Introduction

In general, sensors represents components that provides certain response to specific property within its own environment. Both physical and radar sensors have recently been used by different applications, while the Internet of Things (IoT) has become a perfect enabler for their interactions and data processing operations. In recent time, the combination of radar sensor and physical sensor have been seen in applications such as wearable devices, smart buildings, and autonomous vehicles. Physical sensors usually measures physical quantity (such as temperature) and produce processed signals that can be read by observer or certain instrument. Radar sensors on the other hand are signal conversion devices that uses wireless technology to detect motion. Compared to physical sensors, radar sensors are capable to detect obstructions like glass and walls. The output produced by both type of sensors are available as data streams, which are used by smart applications to support related data-driven operations.

While the IoT focuses on infrastructure issues by identifying and connecting real-life objects, the streaming data produced from these radar and physical sensors are useful in driving data-driven actuation in smart building critical and monitoring systems. In spite of the numerous robust IoT platforms that support data streams produced from these categories sensors, there exists a number of quality issues with the data streams. Such quality problems generated from each category of the sensors often results in false actuation by certain critical or monitoring systems. For example, both radar sensor and physical sensor have been known to be source of inconsistent data streams (redundancy or noise) [1]. In other category, quality issues such as incompleteness (Missing data) and plausibility (Cross Sensitivity) [2]–[4], are known to compromise the accuracy of data streams in physical sensors. These problems can often result in false-positives (e.g. false fire alarm) or erroneous actuation [5], hence making the system's efficiency to be compromised at run-time.

This work contributes by exploring the feasibility of semantic-driven approach for error tracking in sensor streams in order to achieve effective and efficient actuation within the smart home environment. It demonstrate the use of semantic rules and RDF[1] stream processing to achieve semantic reasoning with serialised RDF sensor streams.

The organization of this paper is as follows. Section II introduces the overview of smart actuator and the significance of data stream. The proposed stream quality approach is presented in Section III. Section IV, demonstrates the implementation and deployment of proposed stream quality tracking approach. The Experimental setup and sensor data stream tracking evaluation and results are discussed in Section V. Finally, the paper is concluded in Section VI.

[1]Resource Description Framework

II. SMART ACTUATOR OVERVIEW

The process of actuation within a smart environment (e.g.smart home) are usually driven by a special entity known as the Smart actuator. The smart actuator is incorporates sensors, processors and communicators within its elements which are manipulated by programmable logic or computer software/interface [6]. The application of various types of smart actuators are based on the domain requirements and the type of actuation needed. For example, Smart actuators are used in biomedical field for converting different types of energy e.g. physical energy into mechanical work in response to different natural stimuli such as pH, heat, moisture or humidity, electric or magnetic field. Similarly, in the smart home environment the smart actuator can be used to convert switch the state of thermostat or fire alarm in response to temperature readings or other physical properties measured by the sensors. Figure 1 show the generic architecture for smart actuator as proposed by [7]. The architecture shows sensor data to be at the heart of the decision and actuation process. This indicates that every data involved in the actuation process must be void of noise or quality issues to guarantee effective and efficient system.

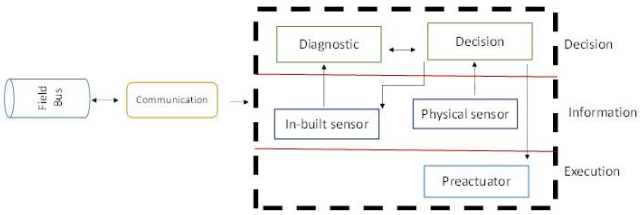

Fig. 1. Generic Architecture for Smart Actuator [7]

III. SEMANTIC-DRIVEN APPROACH FOR TRACKING SENSOR STREAMS

In this section, a description of the semantic-driven approach for the tracking of inconsistent sensor data streams within a smart home is presented. The author's view of the smart home is an intelligent environment that is equipped with both physical and radar sensors, which are connected to the IoT platform and software interfaces for smart actuation. The radar sensor in this case is a multi-channel surveillance system that is used in detecting motion within the home environment.The physical sensor represents the categories of sensors measuring the indoor properties such as temperature, pressure, humidity, CO_2, etc.

The semantic analysis of the sensor streams is conducted using the approach specified in the subsequent sub-sections with focus on the RDF serialised data formats. The semantic stream quality tracking approach is similar to that proposed in [8], [9]. The major difference when compared with the current approach is that it considers output from both radar sensors and physical sensors.

A. Stream Preprocessing

Raw streaming data are received from different physical sensor nodes within the smart home using the MQTT protocol[2]. The Data is then pre-processed for deduplication to resolve the possibility of redundant quadruples statement by using the hash table indexing key/value to provide unique identification for each data stream. Flexible data delivery between software module of the IoT platform are managed by Apache camel[3]. Apache Camel is a lightweight data/file transfer framework that allows integration between different components of a system. It also accepts streaming data in any serialised format and routes the data between modules of software. The unique individual raw sensor stream is later converted into semantic stream through an a method of semantic annotation with the help of the domain ontology. This is discussed in more detail in the following sub-section.

B. Semantic Modelling and RDF Serialisation

In an attempt to provide a suitable data model for semantic querying and reasoning, each of the sensor streaming data is annotated with the support of RDF manager using the domain ontology that describes the smart home entities. The annotation converts each data to triple statement (*Subject, Predicate, Object*) with the timestamps (Quadruple statement) by extracting the namespace including class and property types from the domain ontology. The resulting statement is read as RDF stream which is defined by the RDF model[4]. The RDF streams is further converted into RDF serialised formats using the native Jena RIOT API[5]. This allows to achieve better expressivity and faster processing of streams. The major alternative format for RDF data that has been considered as the serialisation formats for the approach are RDF/XML, N-Triple, Turtle and N3 formats. Figure 2 shows a sample of the RDF/XML serialisation format of a typical semantic stream for temperature reading from a smart home. The RDF data format is able to model both the raw sensor readings and the timestamps.

C. Semantic Reasoning Approach

Semantic reasoning describes the process of producing new knowledge or inference from an existing facts. This method has been developed to achieve the stream quality tracking approach. One way of achieving the semantic reasoning approach is by layering RDF stream processing system (RSP) with a semantic rule [9]. The advantage of this approach in dealing with quality issues provides support for multiple stream querying and data interoperability, which is currently missing in statistical approaches. Furthermore, the use of data quality rules in defining methodologies for processing and detecting anomalies in data is an approach considered in ensuring high data quality [10]. In attempt to achieve

[2]Available: http://mqtt.org/
[3]Available: http://camel.apache.org/
[4]A Standard semantic language
[5]https://jena.apache.org/documentation/io/rdf-output.html

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:smartSpace="http://localhost:8080/smartSpace#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#" >
<rdf:Description rdf:about="http://localhost:8080/smartSpace#pressureReading8">
        <smartSpace:hasPressureReading rdf:datatype="http://www.w3.org/2001/
                    XMLSchema#float">752.17</smartSpace:hasPressureReading>
      <smartSpace:pressureHasTimestamp rdf:datatype="http://www.w3.org/2001/
                    XMLSchema#dateTime">2020-01-20T09:42:12.084Z</
                    smartSpace:pressureHasTimestamp>
      <rdf:type rdf:resource="http://localhost:8080/smartSpace#pressureValue"/>
</rdf:Description>
<rdf:Description rdf:about="http://localhost:8080/smartSpace#pressureReading10">
        <smartSpace:hasPressureReading rdf:datatype="http://www.w3.org/2001/
                XMLSchema#float">753.76</smartSpace:hasPressureReading>
       <smartSpace:pressureHasTimestamp rdf:datatype="http://www.w3.org/2001/
      XMLSchema#dateTime">2020-01-20T09:42:22.085Z</smartSpace:pressureHasTimestamp>
       <rdf:type rdf:resource="http://localhost:8080/smartSpace#pressureValue"/>
</rdf:Description>
```

Fig. 2. RDF/XML Listing

an effective and efficient semantic reasoning process, the C-SPARQL[6] library is adopted as the most appropriate RSP to be combine with the Jena rule langauage[7].

The serialised data streams are selected based on continuous aggregated sliding windows using the C-SPARQL while a set of three explicit rules defined by the Jena rule language are executed over each of the sliding windows for quality tracking. Figure 3 shows these types of rules that are suitable for stream quality tracking within the smart home. The feasibility of the approach including its effectiveness and efficiency are considered in the subsequent section.

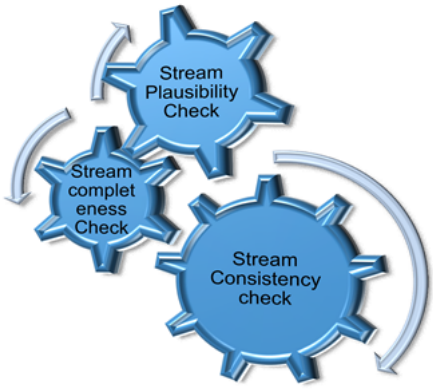

Fig. 3. Sensor Streams Tracking Rules

IV. Approach Implementation

This section describes the implementation of the stream quality approach as a software solution. The implementation is realised using the libraries of Jena, C-SPARQL and JSON[8]. The approach implemented in a case study involving the smart home with integrated sensors connected to IoT platform. The dataset generated from REFIT[9] smart home project was used for the purpose of tracking inconsistent data streams from temperature sensor. These are data specifically produced from one radar sensor (CCTV or weather satellite) and three physical sensors (temperature, pressure and humidity sensors). Each raw data stream produced from each of the sensor types are modelled based on all the four types of RDF serialised data formats. The vision of the smart home is to be able to respond appropriately to specific situation using the actuation process that is void of false alarm/response.

The semantic representation of the raw data is produced with an annotated timestamp using the domain RDF graph (derived from domain ontology) and RDF manager. The resulting RDF quadruple statement is further re-written with the RIOT API to derive the four serialisation formats. These serialised RDF formats consist of the RDF/XML, N-Triple, Turtle and Notation Formats. The new serialised RDF formats are then subjected to further processing by the framework for the semantic validation process. The serialised triples with timestamps were later subjected to semantic reasoning with rules.

```
REGISTER QUERY sensorValueOf  AS
PREFIX smartSpace: http://localhost:8080/smartSpace#
?pressureValue ?humidityReadings ?humidityValue "
  SELECT *
  FROM STREAM http://localhost:8080/smartSpace/streamTemperature [RANGE 25s STEP 7s]
  FROM STREAM http://localhost:8080/smartSpace/streamPressure [RANGE 25s STEP 7s]
  FROM STREAM http://localhost:8080/smartSpace/streamHumididty [RANGE 25s STEP 7s]
        WHERE {
         ?tempReadings smartSpace:hasValue ?tempValue.
         ?tempReadings smartSpace:hasTimestamp ?tempTime.
         ?tempReadings smartSpace:hasId ?tempId.
         ?tempReadings smartSpace:hasSeason ?tempSeason.
         ?tempReadings smartSpace:hasTimestamp ?tempTime.
         ?pressureReadings smartSpace:hasPressureReading ?pressureValue.
         ?pressureReadings smartSpace:pressureHasTimestamp ?pressureTime.
         ?humidityReadings smartSpace:hasHumidityReading ?humidityValue.
         ?humidityReadings smartSpace:humidityTimestamp ?humidityTime.
       }
      ORDER BY ASC(?tempTime)
```

Fig. 4. C-SPARQL Query for selection of Sensor Streams

The semantic reasoning for quality tracking begins with the execution of stream query in figure 4. It adopts window-based processing to support multiple RDF streams selection. During the C-SPARQL query execution stage, continuous pattern matching of concepts and properties are performed on each semantic statement describing physical properties, which includes temperature, pressure and humidity and the corresponding values and timestamps. The selection of the streaming data variable within each streaming window is ordered by timestamps and executed over a period of 25 seconds with step interval of 7 seconds for indoor temperature values and 7 seconds for other related physical properties (indoor pressure and humidity values). It is expected that using a shorter step length than the streaming window length can sometimes result in unnecessary duplication of the query result. Therefore, the results from each query processing window are received by the ActiveMQ broker which is later managed by Java Message Service (JMS) and subsequently processed concurrently by the reasoning engine.

[6]http://streamreasoning.org/resources/c-sparql
[7]Available: http://jena.apache.org/documentation/inference
[8]json-simple-1.1.1.jar
[9]https://www.refitsmarthomes.org/datasets/

```
@prefix rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
@prefix owl: http://www.w3.org/2002/07/owl#
@prefix rdfs: http://www.w3.org/2000/01/rdf-schema#
@prefix xsd: http://www.w3.org/2001/XMLSchema#
@prefix smartSpace: http://localhost:8080/smartSpace#
[consistencyCheck:
        (?humidityReadings smartSpace:hasHumidityReading ?humidityValue)
        (?humidityReadings smartSpace:humidityHasTimestamp ?humidityTime)
        greaterThan(?humidityValue,39)
        lessThan(?humidityValue,51)
        (?tempReadings smartSpace:tempHasTimestamp ?tempTime)
        (?tempReadings smartSpace:hasValue ?tempValue)
        greaterThan(?tempValue,17)
        lessThan(?tempValue,24)
        (?pressureReadings smartSpace:hasPressureReading ?pressureValue)
        (?pressureReadings smartSpace:pressureHasTimestamp ?pressureTime)
        greaterThan(?pressureValue,750.1)
        lessThan(?pressureValue,761.0)
        le(?tempTime,?humidityTime)
        le(?tempTime,?pressureTime)
        ->
        (?tempReadings smartSpace:isValid 'Consistency Check')
]
```

Fig. 5. Consistency Validation Rule for Temperature Stream

To achieve a complete reasoning process and for proper inference, each serialised format of the semantic streams extracted from the current window are executed against the three specified tracking rules and supported by the Jena inference subsystem and API. The reasoning rules in this case are defined based on specific policy in a smart office specified as health and safety regulations. Specifically, the rules are defined from the occupational health and safety[10] recommendation for indoor temperature, pressure and humidity. The order of execution of each rule set is considered in terms of firstly analysis each stream for completeness, followed by the execution of the rule in figure 5 to maintain consistency and, finally executing the plausible rule based on the output of the radar sensor. In this context, a plausible reading is a correct sensor reading that does not represent the expected reading of target property within the indoor environment. Plausible reading are estimate of the target property that has been influenced by an external property e.g. aggregating human body temperature readings instead of indoor temperature due human closeness to temperature sensor. This type of example will require data from radar sensor before such readings can be classified as plausible data.

The tracking rules executes at run-time to provide sensor values with corresponding timestamp that satisfy the conditions in the rules and, make such available to web applications through API. Figure 6 is a snapshot of the inference from the semantic reasoning process. The output of the reasoning process is a new knowledge confirming the consistency or otherwise of each data point, which is available as a quadruple statement(triple statement and timestamp). The software implementation in figure 7 contains the real time analytic interface of the approach. It is deployed as a plug-in and integrated with a cloud-based Microsoft Azure platform in order to demonstrate the feasibility of the approach and allows for continuous integration and tracking of sensor streams.

[10]http://www.ohsrep.org.au/hazards/workplace-conditions/heat

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:smartSpace="http://localhost:8080/smartSpace#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#" >
<rdf:Description rdf:about="http://localhost:8080/smartSpace#temp2Readings2">
  <smartSpace:isInconsistent>Erroneous reading</smartSpace:isInconsistent>
  <smartSpace:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#float">-27.4</smartSpace:hasValue>
  <smartSpace:tempHasTimestamp rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2020-07-28T03:32:58.992Z
</smartSpace:tempHasTimestamp>
  <smartSpace:hasId rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Sensor 2</smartSpace:hasId>
  <smartSpace:hasSeason rdf:datatype="http://www.w3.org/2001/XMLSchema#string">summer</smartSpace:hasSeason>
  <rdf:type rdf:resource="http://localhost:8080/smartSpace#tempValue"/>
 </rdf:Description>
<rdf:Description rdf:about="http://localhost:8080/smartSpace#humitidy2Readings2">
  <smartSpace:hasHumidityReading
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">43</smartSpace:hasHumidityReading>
  <smartSpace:humidityHasTimestamp rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2020-07-28T03:33:01.984Z
</smartSpace:humidityHasTimestamp>
  <rdf:type rdf:resource="http://localhost:8080/smartSpace#humidityValue"/>
 </rdf:Description>
<rdf:Description rdf:about="http://localhost:8080/smartSpace#pressureReading2">
  <smartSpace:hasPressureReading
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">756.75</smartSpace:hasPressureReading>
  <smartSpace:pressureHasTimestamp rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2020-07-
28T03:33:21.986Z</smartSpace:pressureHasTimestamp>
  <rdf:type rdf:resource="http://localhost:8080/smartSpace#pressureValue"/>
 </rdf:Description>
</rdf:RDF>
```

Fig. 6. Sample output from Sream Quality Tracking with RDF/XML serialisation

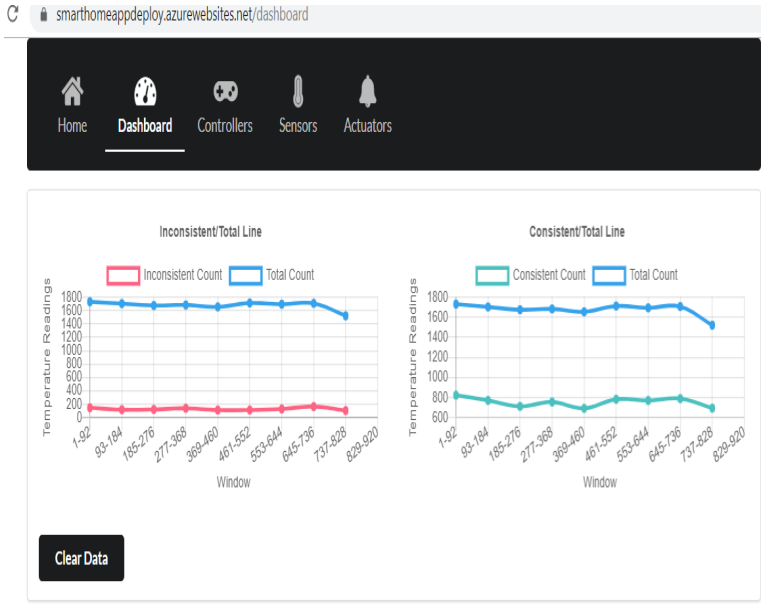

Fig. 7. Real-Time Data Validation Analytic

V. Evaluation and Results

To provide better understanding of performance of the sensor stream quality tracking approach, a simulated experiment on a single node centralized server with multiple processor. The simulated experiment involve large dataset consisting of raw sensor readings with the timestamps, which was produced from radar and physical sensors. The simulation consists of eight rounds of experiments running for six hour each and, completed in two separate experimental runs. Each of the two experimental runs is allowed to perform stream quality

tracking on the same type data set generated by streamer using the four RDF serialised formats.

In particular, outputs from ten sensor nodes that measures temperature, pressure, humidity, door open/close, window open/close, the motion of body and climate data are simulated. Specifically, the configuration of the sensors consists of 3 temperature sensors, 2 humidity sensors, 2 pressure sensors, 1 door sensor, 1 window sensor and 1 motion sensor. Correspondingly, each class of sensors generates the related streaming data for the property it measures at a different streaming rate measured in seconds. Each of the temperature sensor nodes is simulated to generate both true values and erroneous streaming values that explicitly represents typical Inconsistent, Plausible and missing readings respectively. Inconsistent streaming data values were injected into specific streaming windows at different intervals of a single (1) and ten (10) data points in separate experimental runs. In all rounds of the experiment, inconsistent and missing temperature sensor streams are data points represented explicitly by -27.4^0 *Celsius* and a pseudo-value of '8888' respectively. Plausible values are temperature readings recorded from direct interference with external or climatic/weather temperature readings.

The evaluation process specifically targets estimating the semantic tracking of Inconsistent data point produced within each processing window. The summary of semantic validation is produced over continuous validation windows with the cycle. In an attempt to get a smooth trend in interpreting the output of each validation cycle concerning the effectiveness and efficiency evaluation, the Cumulative Moving Average (CMA) is applied in all the evaluation results.In the first round of the experiment involving a single point of an inconsistent data point, the approach can perform semantic validation on average of 49 quadruples in each streaming window and a total of 3150 per cycle. The approach produced a total of 338,904 inferred quadruple statements over 2000 cycles. Similarly, in the second experimental run, the approach is able to produce an average of 103 quadruples in each streaming window with a total of 3035 quadruples statements per cycle. The reduction in the total quadruples processed by each validation cycle in the second experiment is due to the presence of the Hash function built as part of the approach. This is responsible for removing duplicate value with the same timestamps among raw streaming data. A total of 341,472 inferred quadruples are produced at the end of 2000 cycles.

The effectiveness of the approach is determined by the estimation of *Relevance score* and *Validation Accuracy* of the intervals of the processing cycles involving the serialised RDF formats. The method of deriving the effectiveness of the framework is similar to what is currently used Information Retrieval (IR). This similarity is seen in terms of classification problem, which also adopt a similar technique for pattern matching during the semantic query and reasoning process. Furthermore, efficiency Metrics further evaluate the performance of the approach as a means of its cost (time performance) on the system resources in which it is deployed. In an IoT domain that is heavily dependent on streaming data for data-driven

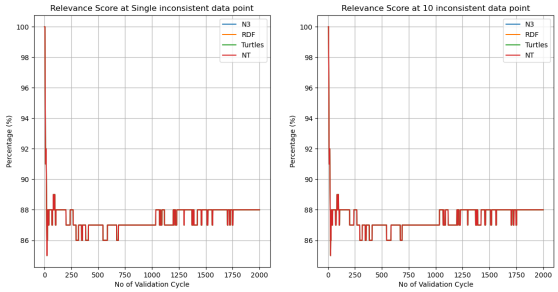

Fig. 8. Relevance Ratio of Serialised RDF Formats at two different experimental Runs with Injections of Inconsistent Data Points per Streaming window

processing such as critical system, the importance of time is inevitable. As such, systems or applications will require to operate promptly with possible limited resources. Therefore, the time-based metrics include the Semantic reasoning time of the stream quality tracking approach.

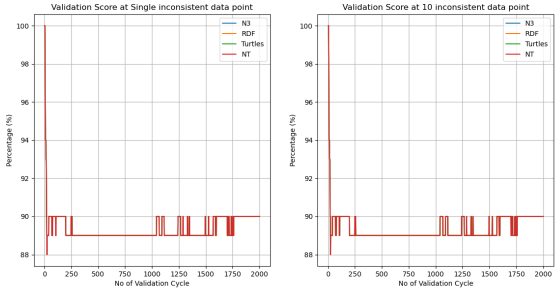

Fig. 9. Validation Score of RDF Formats at Two Experimental runs with Injection of Inconsistent Data Points per Streaming window

The two experimental runs are able to produce a total of 680,376 inferred quadruple statements (triple statement with timestamp) from the semantic validation process. The number of quadruples produced within each streaming window depends on the maximum duration of semantic stream selection and sleep duration. Figure 8 shows the results of the estimation of the Relevance score of the validation of *SISDaV* from both experimental runs. The score was stable between 86% and 88% for the two experimental runs with not much significant difference among the serialised formats. The spike in the first 10 windows is due to low plausibility count at the earlier stage of the streaming node. In addition, the drop in the relevance ratio between the 250th and 1000th validation cycle is caused by aggregated streaming windows with a significant number of Plausibility count. Similarly, Figure 9 presents the validation score showing the Validation score (accuracy) of *SISDaV* framework above 88% across all the validation cycles in both experimental run. Furthermore, the RDF serialise formats do not have any significant effect on the semantic validation process as all of them can reach a peak of 90% in both experiments from the 10TH Cycle. The implication of the results from the evaluations suggests a slight change in terms

of effectiveness, particularly in application with a high error rate. In addition, the result corresponds to the total fraction of the inconsistent data points injected into each validation cycle during both experiments.

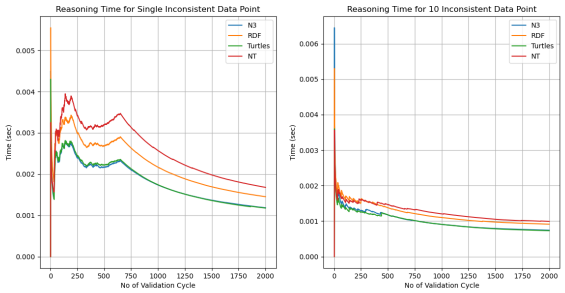

Fig. 10. Reasoning Time of RDF Formats at Two Experimental runs with Injection of Inconsistent Data Points per Streaming window

The efficiency of approach has been considered in terms of estimates of the average time required to complete the reasoning task for each processing cycle. Figure 10 shows the performance of the reasoner when the validation rules were executed in each processing window. The graph provides the estimate of the average time to complete a semantic reasoning task and provide an inference within a validation cycle.The N-Triple and RDF/XML formats require more time in seconds to perform inference in the first experiment, which is slightly lesser in the second experiment compared to counterpart serialised formats. The estimate from both experiments indicates the structure of N-Triple and RDF/XML serialised formats has effects on their expressivity. Most likely due to the resource-constrained feature, which will require more time to be processed by the semantic reasoner or semantic reasoning engine. Also, the speed of processing decreases along with the validation cycles for all the serialised format, thanks to the optimized matching technique embedded in the Jena2 reasoner [11], in which the reasoning engine was built upon.

VI. Conclusion

In this paper, a semantic approach to dealing with stream quality issues in a to minimise false actuation is proposed. The approach takes advantage of the semantic query and reasoning to achieve stream interoperability and validate each raw data point before been consumed by the actuator. The work further demonstrate the feasibility of the approach and evaluate the expressivity of serialised data formats while using the approach.The effectiveness and efficiency evaluation proves that the approach can be reasonably sustained in a temporal or time-critical software applications.

References

[1] I. D. Castro, M. Mercuri, A. Patel, R. Puers, C. Van Hoof, and T. Torfs, "Physiological driver monitoring using capacitively coupled and radar sensors," *Applied Sciences*, vol. 9, no. 19, p. 3994, 2019.

[2] A. Karkouch, H. Mousannif, H. Al Moatassime, and T. Noel, "Data quality in internet of things: A state-of-the-art survey," *Journal of Network and Computer Applications*, vol. 73, pp. 57–81, 2016.

[3] L. M. Ang, K. P. Seng, A. Zungeru, and G. Ijemaru, "Big Sensor Data Systems for Smart Cities," *IEEE Internet of Things Journal*, vol. PP, no. 99, pp. 1–13, 2017.

[4] P. Barnaghi, M. Bermudez-Edo, and R. Tönjes, "Challenges for Quality of Data in Smart Cities," *Journal of Data and Information Quality*, vol. 6, no. 2-3, pp. 1–4, 2015.

[5] P. E. Brown, T. Dasu, Y. Kanza, and D. Srivastava, "From rocks to pebbles: Smoothing spatiotemporal data streams in an overlay of sensors," *ACM Transactions on Spatial Algorithms and Systems (TSAS)*, vol. 5, no. 3, pp. 1–38, 2019.

[6] R. Agrawal, C. Koteswarapavan, N. Kaushik, and P. Matre, "Chapter 7 - smart actuators for innovative biomedical applications: An interactive overview," in *Applied Microbiology and Bioengineering*, P. Shukla, Ed. Academic Press, 2019, pp. 101–119.

[7] M. Bayart and M. Staroswiecki, "Smart actuators: generic functional architecture, service and cost analysis," in *Singapore International Conference on Intelligent Control and Instrumentation [Proceedings 1992]*, vol. 1. IEEE, 1992, pp. 642–646.

[8] O. Bamgboye, X. Liu, and P. Cruickshank, "Towards Modelling and Reasoning About Uncertain Data of Sensor Measurements for Decision Support in Smart Spaces," *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, pp. 744–749, 2018.

[9] ——, "Semantic Stream Management Framework for Data Consistency in Smart Spaces," *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 2, pp. 85–90, 2019.

[10] L. Li, T. Peng, and J. Kennedy, "A rule based taxonomy of dirty data." *GSTF Journal on Computing*, vol. 1, no. 2, 2011.

[11] J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson, "Jena: implementing the semantic web recommendations," in *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, 2004, pp. 74–83.