IoTSAX: A Dynamic Abstractive Entity Summarization Approach With Approximation and Embedding-Based Reasoning Rules in Publish/Subscribe Systems

Niki Pavlopoulou and Edward Curry, *Member, IEEE*

***Abstract*—Users are interested in entity information and may use paradigms like publish/subscribe systems in Internet of Things (IoT), where entity-centric data comes from multiple sources. The IoT application demands contributed to the data integration and semantic interoperability need while maintaining high usability and low processing system resources usage. Existing approaches dealt with semantic interoperability by using/extending ontologies with strict schemata/semantics and difficult to be updated. Other approaches tackled high usability by providing sensor data abstractions that infer high-level knowledge, but they are either complex for real-time processing or have semantic/syntactic coupling making them nonflexible. Therefore, our key question is: can a dynamic entity summarization system that provides high-level abstractions while ensuring semantic/syntactic decoupling, and low use of processing system resources, be defined? In this article, we address this question by proposing the abstractive publish/subscribe summarization system that provides abstractive summaries of numerical IoT data for user-friendly subscriptions by using IoTSAX, an enhanced symbolic aggregate approximation (SAX) methodology for dynamic IoT environments, and approximate rule-based reasoning by using embedding models. Our results for two use cases, medicine and smart cities, show that although abstraction can be 2 to 3 times slower in latency, it achieves up to 98% reduction in the notifications' number. IoTSAX outperforms SAX in approximation error up to 2 to 3 times more and in compression space-saving percentage when data redundancy occurs, while it has a similar or better latency and throughput performance. Finally, concept hierarchy-based embedding models can achieve F-scores of up to 0.87 for approximate rule-based reasoning.**

***Index Terms*—Abstractive entity summarization, approximate reasoning, data approximation, Internet of Things (IoT), publish/subscribe.**

Manuscript received February 9, 2021; revised May 18, 2021; accepted June 7, 2021. Date of publication June 16, 2021; date of current version January 24, 2022. This work was supported in part by the European Union's Horizon 2020 Research Programme Big Data Value Ecosystem (BDVe) under Grant 732630, and in part by the Science Foundation Ireland (SFI) under Grant SFI/12/RC/2289_P2, co-funded by the European Regional Development Fund. *(Corresponding author: Niki Pavlopoulou.)*

The authors are with the Insight Centre for Data Analytics, National University of Ireland Galway, Galway, H91 AEX4, Ireland (e-mail: niki.pavlopoulou@insight-centre.org; edward.curry@insight-centre.org).

Digital Object Identifier 10.1109/JIOT.2021.3089931

I. Introduction

THE ASPIRATION for the future is to have a world that is fully connected so that communication, sharing, and vital actions can take place when needed. This can be accomplished by the Internet of Things (IoT) as it has played an important role in the past years in connecting different sensors and devices at large scale [1]. Nevertheless, IoT's future use poses many challenges as the world gets more complex, the data more abundant and the demands for its use in applications like smart cities and medicine increase [2]. One of these applications is that users are interested in entities, like a patient or a property, for which there is an abundance of information provided by sensors. Therefore, the implementation of dynamic IoT solutions is required that can offer more flexibility, reusability, effectiveness, and efficiency.

Today, the abundance of data and sensors hinders the analysis instead of helping it as there is a lack of semantic interoperability [3]. The sensor data cannot stand on its own and it often needs to be integrated with other data to observe its spatiotemporal dependency in order to infer a more complete knowledge to the user [4]. Also, the application needs and the data distribution might change (e.g., concept drift) [5]; therefore, data processing should be redesigned accordingly. Nevertheless, the more sensors used and the more changes in the design, the higher the possibility of having more heterogeneity [6], that is the existence of different semantics and schemata. So far most works have dealt with the semantic interoperability by using or extending different ontologies (e.g., semantic sensor network ontology) [3], [7], [8]; however, ontologies have strict schemata and semantics, they are difficult to be updated, they need domain experts who might not be available for all domains and some are domain dependent; therefore, they can be applied to limited applications [9].

On the opposite end of the spectrum, there is also more abundance of users. These users can range from experts to nonexperts; therefore, it is vital for future IoT systems to cater for the highest possible usability and ambiguity. Most graph-based systems use complex queries, like SPARQL; nevertheless, apart from them being strict in their syntax and semantics, they can also be very complex for a nontechnical user [10]. Also, different users have different perspectives according to their

expertise or lack thereof in a domain. Therefore, systems have tried with multiple approaches to create higher level abstractions from sensor data so that the user is not presented with an abundance of IoT data that may overwhelm them, but with abstractions that infer high-level knowledge [11]. However, these approaches are either very complex for real-time processing or they have semantic and syntactic coupling that makes them nonflexible.

IoT also suggests environments that will need to be efficient in real-time analysis. The abundance of sensors and the frequent sampling rate create continuous data that can be voluminous, identical, or similar. Given that challenges like memory, power, and network restrictions will always be vital in IoT [12], then, dynamic IoT solutions are needed that can cater for the highest possible system performance.

For example, in applications where users are interested in entities, like a patient or a property, there is a plethora of sensors generating information about them. Existing systems that can support a dynamic environment like IoT in terms of system performance (e.g., memory, power etc.) would either support simple queries for nonexpert users or deal with the semantic interoperability and data heterogeneity of the sensors. There is no existing system that can cater for both at the same time. For example, a system that is based on keyword-based queries (e.g., query = "patient name") is user-friendly, but it creates an abundance of unnecessary redundant information, which may overwhelm them. On the other hand, a system that is based on a filtered query, like the following SPARQL one

```
SELECT ?heartRate
FROM STREAM
WHERE {
PatientName heartRate ?heartRateValue;
FILTER (?heartRateValue > 100bpm).
}
```

would give more targeted information to the users, but at the expense of them being experts in complex queries and knowing the exact syntax and semantics used by the available sensors to create these specific filters.

Some systems have even gone beyond the typical filtered queries to deal with data heterogeneity by using rule engines and languages to provide abstract data information to the users. The rules, apart from some of them being complex and hard to extract, are based on ontologies or are defined by domain experts, therefore, they contradict their initial purpose as, in the end, they become rigid themselves. For example, the following rule taken from the M3 framework's rules[1] defines whether there is high lighting in a building based on readings by a "luminosity" sensor

```
HighLighting:
  (?measurement rdf:type m3:Luminosity)
  (?measurement m3:hasValue ?v)
  greaterThan(?v,750)
  lessThan(?v,2000)
  →
  (?measurement rdf:type home-dataset:HighLighting)
```

[1]https://github.com/gyrard/M3Framework/tree/master/war/RULES

This rule is rigid in terms of its syntax and semantics since if the system contains readings by a light or luminance sensor, then, these would not apply to the following rule and they would be missed as available information to the user.

In this article, we propose the abstractive publish/subscribe summarization system that provides abstractive summaries of numerical IoT data for user-friendly subscriptions with the use of an enhanced approximation technique and approximate rule-based reasoning. Approximation is highly used in resource-constraint environments and it provides acceptably smaller and quicker notifications provided the error range is low [13]. Reasoning is also used for reducing memory and communication traffic as well as the deduction of meaningful information from sensor data so that the users are not overwhelmed with redundant information and they are not biased based on their expertise [14]. Approximate rule-based reasoning will abstract the systems from using common formats and terminologies so that they can achieve semantic interoperability as well as release them from the limitations of ontologies, taxonomies and domain experts as they are based on statistical embedding-based models [15]. In this way, rules can be shared and reused among different systems and for different application needs.

Our main contributions are as follows.

1) A user-friendly entity-centric subscription that allows subscribers to express in a simple way whether they need to receive an entity summary of a specific window type with a desired window size.
2) IoTSAX, an enhanced dynamic approximation technique for numerical IoT data that is based on the symbolic aggregate approximation (SAX) methodology by providing dynamic segment points as well as alphabet size for symbolic representations that follow data fluctuations.
3) A novel approximate rule-based reasoning approach that is based on data approximation interpretation and embedding models that create quad-like entity-based abstractive summary notifications.
4) An evaluation methodology using synthetic and real-world IoT data as well as ontologies for examining the tradeoff among latency, throughput, compression space saving, number of forwarded messages, approximation error and F-score by using windows.

The remainder of this article is structured as follows. In Section II, we present the problem analysis that consists of motivational scenario and its challenges, whereas Section III presents the preliminaries. Section IV contains related work, whereas Section V contains an overview of the abstractive publish/subscribe summarization system. Section VI describes the IoTSAX approach, whereas evaluation and results are in Section VII. Finally, conclusions are drawn in Section VIII.

II. Problem Analysis

This section analyses the problem with the use of a motivational scenario and its challenges.

A. Motivational Scenario

A user is interested in events concerning an entity, therefore, they create subscriptions with the topic "entity name."

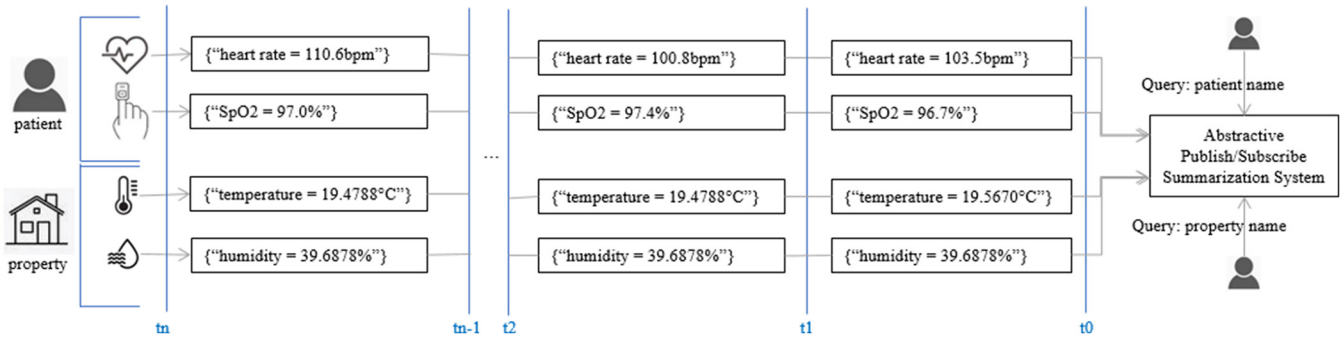


Fig. 1. User is interested in approximate information about a patient or a property. Multiple sources generate timestamped information records.

Multiple sensor readings (publishers) provide an abundance of information about this entity. For example, a doctor is interested in the medical information of one's patient, which involves body sensor readings like blood pressure, heart rate, etc. At the same time, a resident of a property is interested in the readings provided by the building like temperature, humidity, etc. The deployed sensors could belong to different manufacturers. The busy work or lifestyle of the users does not permit them to explicitly focus on all available sensor readings. Therefore, they would like a rough view of all the provided sensor data without much manual user input in order to understand if they need to take further actions (e.g., book a doctor's appointment, reduce energy consumption etc.). The motivational scenario is illustrated in Fig. 1.

B. Challenges

The challenges involved in the aforementioned motivational scenario are the following.

1) *Data-Related Challenges:*
   a) *Heterogeneity:* The sensors deployed belong to different manufacturers; therefore, different schemata and semantics are used. The users would expect to be abstracted from the lack of semantic interoperability and be presented with a notification in a common representational format.
   b) *Redundancy:* Frequent sampling rates generate identical sensor data with unchanged states for periods of time (e.g., Fig. 1: humidity = 39.6878%). The redundancy will result in voluminous unnecessary data that will overwhelm the user and it will impose resource limitations to the processing system.
2) *User-Related Challenges:*
   a) *High-Level Interpretation:* The users are not interested in specific sensor values (e.g., Fig. 1: heart rate = 100.8 bpm), but in a rough view of an entity including numerical or abstract information (e.g., Fig. 1: patient has tachycardia).
   b) *Simple Data Representation:* The notification should be presented in a suitable and understandable structure for the user.
   c) *Low User Expressibility:* The users might be nonexperts; therefore, complex queries like SPARQL are deemed unfriendly. Also, they might not be aware of the sources available nor of the semantics or schemata used in the sensor data to make more explicit queries (e.g., Fig. 1: "peripheral capillary oxygen saturation" instead of "SpO2"). The users would prefer a noncomplex contextual-aware query that provides a rough view of an entity.
3) *Performance-Related Challenges:*
   a) *Continuity:* The sources constantly update the entity's information, and the users need the most recent one.
   b) *Resource Constraints:* The heterogeneity and redundancy, mentioned above, create voluminous data that causes network overhead. Nevertheless, the data needs to be processed quickly, in real time and with as low memory, power, and network consumption as possible.

III. Preliminaries

This section provides the necessary background regarding publish/subscribe systems (PSS) and their limitations, the usefulness of knowledge graphs and abstractive summaries to the problem as well as embedding models and their use in semantic approximation.

A. Publish/Subscribe Systems

PSS provide a suitable interaction scheme for dynamic large-scale applications. Subscribers (users) express their interest in an event or pattern of events via subscriptions that are sent to the Event Engine. Publishers generate events via publications that are also sent to the Event Engine. A matcher is contained in the Event Engine that matches specific events to subscriptions, based on their conditions, resulting in subscribers being notified [16]. PSS are characterized by space decoupling (publishers and subscribers do not need to know each other), time decoupling (publishers and subscribers do not need to be active at the same time) and synchronization decoupling (publishers are not blocked during event production, and subscribers can be notified while performing another activity).

PSS have three main schemes: 1) topic based; 2) content based; and 3) graph based. In the topic-based PSS, events and subscriptions are related to a topic expressed as keywords

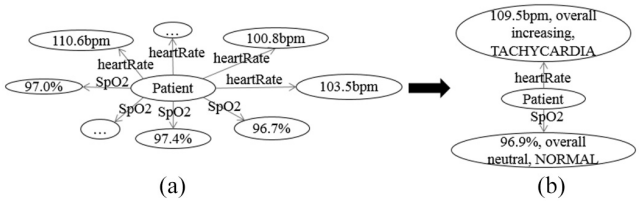


Fig. 2. Existing PSS schemes can overcome the performance-related challenges of Fig. 1. Nevertheless, the only scheme that can deal with the low user expressibility challenge is the topic-based one as it supports noncomplex keyword-based subscriptions (topic = "patient" in this example) but at the cost of creating redundant notifications with high heterogeneity (a). The aim of an Abstractive PSS is to combine the advantages of the topic-based and graph-based PSS by supporting noncomplex subscriptions and providing rich graph notifications that deal with redundancy, heterogeneity and the users' need for high-level data interpretation with the use of abstractive entity summarization (b). (a) Graph with all timestamped information of a time frame. (b) Abstractive entity summary.

(e.g., sports). In the content-based PSS, events are mostly attribute-value pairs and subscriptions are expressed as filters with the use of comparison operators ($=, <, \leq, >, \geq$) or logical combinations (and, or etc.) of individual constraints (e.g., gender = female AND age < 30). In the graph-based PSS, events are graphs and subscriptions are SPARQL-like queries that focus on specific nodes and relations between them. All three schemes have different limitations, including the need of a shared understanding of topics, the use of specific schemata and semantics, the creation of complex queries by nonexperts and the possibility of redundant or unnecessary information in notifications, suggesting that PSS need to be enhanced.

B. Knowledge Graphs and Abstractive Entity Summarization

As aforementioned, the challenges in IoT environments are many. The data in IoT comes from multiple sensors and relates to different entities. Given the sensor volume, the number of entities and the changeability of data involved, a richer but simple representation should be provided that represents the raw sensor data to conceptual entities with their associated properties. Knowledge graphs could be used as a representation since their nodes represent entities, and their directed labeled arcs constitute relations among them. In Fig. 1 the sensor readings could be represented as graphs, like in Fig. 2(a), where *patient* is an entity, all values are literals, whereas *heartRate* and *SpO2* are properties. Resource description framework (RDF) is a data modeling language that expresses these representations as triples ⟨subject, property, object⟩, where the subject is an entity, the object is a literal and the property is their relation.

Knowledge graphs by themselves are not able to tackle the challenges involved with the high data volume and the resource constraints in IoT. Entity summaries could be used in this case as a subset of the whole entity information is selected [13], [17]. There are two strategies in summarization.

1) *Extractive Summary:* The summary is a subset of the original information provided. In entity summarization, all graph nodes and their relations are taken by the original source graphs.
2) *Abstractive Summary:* The summary differentiates from the original information provided. In entity summarization, either new graph nodes and their relations are created or a high-level interpretation is provided.

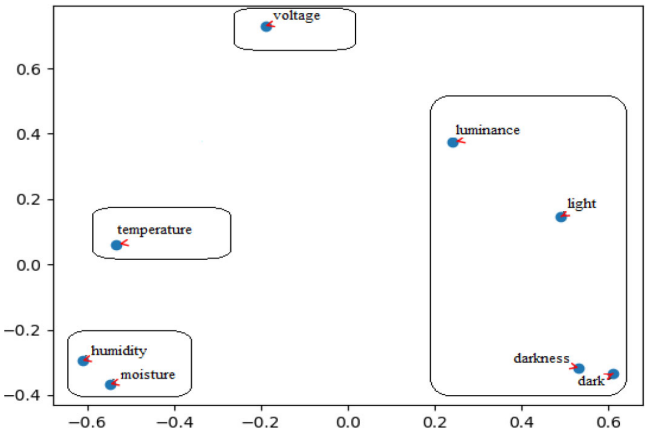


Fig. 3. Vector space depicting the position of each word based on the ConceptNet embedding model. Each partition represents the close relationship among words. For example, {luminance, light} are related words, {darkness, dark} are synonyms to each other but antonyms to {luminance, light}, therefore, all words are semantically related. We observe that related words are close in the semantic space.

An abstractive entity summary is given in Fig. 2(b), where for each distinct property we observe its aggregated value, its behavioral pattern for a time frame and its high-level interpretation with the use of approximation and reasoning methodologies.

C. Word Embedding Models and Semantic Decoupling

In environments like IoT that contain data with high heterogeneity and redundancy, it is important for the words contained in the data not to be treated as individual ones (semantic coupling). Their semantic relationship should be examined in order to discover synonymous, related or antonymous words. Word embeddings are Deep Learning models that find the semantic and contextual relationship among words. The models are trained on big corpora and they create vectors for each word that when projected in a vector space they will show related words close to one another. These models can be used for semantic interoperability since, unlike ontologies used so far, they are not bound by specific semantics, schemata or hierarchies (e.g., subClassOf) and they do not need domain experts. They are only limited to the corpora that they are trained on.

One of the first most popular and still highly used word embedding models is Word2Vec [18], where the model is trained to predict neighboring words. FastText [19] is another popular model that was based on Word2Vec. Recent research has focused on models that emphasize either on the polysemy of a word and give a separate vector for each sense/individual meaning of a word like MSSG [20] and DeConf [21] or on concept hierarchies like ConceptNet [22]. An example of a vector space is illustrated in Fig. 3.

IV. Related Work

The related work is split into two categories; approximation and contextual awareness. An overall comparison among approaches is given in Table I.

TABLE I
OVERALL COMPARISON OF DIFFERENT APPROACHES

Approximation							
Approach	Streaming/IoT	Raw Values	True Distribution	Dynamic Segments	Dynamic Alphabet Size	Extreme Values	(Sharp) Fluctuations
SensorSAX [?]	✓			✓			
Puschmann et al. [?]	✓	✓	✓				
SAX-ARM [?]			✓(normal distribution assumption)		✓(dependent on % of extreme values)	✓	
Zan et al. [?]				✓	✓		
IoTSAX	✓	✓	✓	✓	✓	✓	✓

Contextual Awareness (selected approaches)						
Approach	Semantic Decoupling in Rules	Syntactic/Semantic Decoupling in User Queries	Approximate Rules	Ontologies Dependency	Embedding Models Dependency	Domain Agnostic
Ganz et al. [?]		N/A		✓		
Puschmann et al. [?]		N/A				✓
Wei et al. [?]			✓(spatial)	✓		
Hasan et al. [?]	N/A	✓(semantic)	N/A			✓
IoTSAX	✓	✓	✓(semantic)		✓	✓

A. Approximation

1) Approximation in Time Series: Time series approximation has been widely used for reducing the dimensionality of the original data. Among the abundance of the existing techniques, a few are the most popular ones. Discrete Fourier transformation (DFT) [23] transforms a signal from the time domain to the frequency one. The original signal is represented by the first few Fourier coefficients. Discrete wavelet transformation (DWT) [24] transforms a vector into a set of smooth values and wavelet coefficients that consist of the average and the differences of every other two values of the vector. The process is iterative by considering as vector the wavelet coefficients of the previous step until only a single value remains. Principal component analysis (PCA) [25] transforms the original data representation to a new orthogonal base by calculating their covariance or their singular value decomposition (SVD) [26]. Piecewise aggregation approximation (PAA) [27] transforms a time series vector into a reduced vector of a predefined segment length by taking the average of the original vector values within each segment. SAX [28] transforms a time series into a set of letters. Initially, normalization and PAA are applied to the original vector and then the reduced vector is represented by letters based on breakpoints according to the standard normal distribution.

Among these approaches, we focus on SAX, a lossy compression method. There have been several studies that have shown the superiority of SAX and PAA, which SAX is based on, compared to other approximation techniques [14], [29]–[31]. Specifically, PAA and SAX are simple and computationally much less costly without requiring complex matrix computations, such as in PCA and SVD. They have no limitation on the time series length, whereas in the case of DFT and DWT the length should be of the power of two. Even though their parameters can affect their performance, the loss of the original data's nature and quality may be more evident in the cases of DFT, DWT, SVD, and PCA when the best number of coefficients or principal components should be defined. They do not require a global calculation if the data is not normalized; therefore, they could be implemented incrementally or in batches for quicker processing time, unlike SVD and PCA. Like the other approaches, they also satisfy the lower bounding principle, that is a distance measurement can be found on the reduced vector that is guaranteed to be less than or equal to the true distance measured on the original vector. They are the best at retaining the main characteristics of the original time series by also having high dimensionality reduction; therefore, there is a high correlation between the reduced/symbolic vector and the original one. SAX has also the added advantage that it can provide symbolic representations in order to observe patterns or occurrences of specific numerical ranges unlike the rest of the approaches.

The aforementioned SAX's advantages and the fact that it can be used to approximate massive and high-dimensional data, like in the case of IoT, made it possible to be used in sensor-based research either as an approximation approach for optimization purposes or for finding spatiotemporal correlations [4], [32]. Several works have proposed optimized ways, but most are not related to sensor data. Regarding sensor data, SensorSAX [33] proposes an adaptive segment length in PAA that is based on the streaming activity of data in order to reduce data communication and original information loss. Puschmann *et al.* [32] kept the raw values of the data (no normalization) and find its true distribution based on kernel density estimation (KDE). The distribution's equiprobable regions are examined to define the symbols to be used for the symbolic representation. Regarding some notable nonsensor data works, SAX-ARM [34] uses an inverse normal transformation for normalization to ensure normal distribution of the original time series and focuses on multivariate time series rather than univariate ones. It also defines a dynamic alphabet size for the symbolic representations based on the percentage of the deviant events that one would like to observe. Zan and Yamana [35] is the only work that focuses on enhancing all of the parameters of SAX. The dynamic segments are defined by recursive calculations and the fitting of linear regression to the data of each segment according to a threshold. The dynamic alphabet size for the symbolic representation is based on the skewness of the approximate data distribution. However, all of these works apart from Puschmann *et al.* [32], normalize

the data and assume normal distribution, which could not be the case in IoT data. Apart from Zan and Yamana [35], the works focus on enhancing either PAA or SAX, but not both. Apart from SAX-ARM, the works do not observe extreme values and no work observes sharp fluctuations of data that could prove meaningful depending on the nature of the data. Finally, the nonsensor data works are time-consuming and they are not optimized to apply on a streaming environment like IoT.

B. Contextual Awareness

1) High-Level Abstraction in IoT: There are a number of works that have tackled contextual awareness from raw numerical IoT data. The main approaches are supervised or unsupervised learning models (e.g., neural networks, clustering), manual rules, fuzzy rules, ontology-based, probabilistic models (e.g., Markov models) in order to transform low-level context to high-level one [11], [36]. Nevertheless, there is no consensus on which approach is better as each of them has its own advantages and disadvantages.

Semantic interoperability has also been tackled by rule engines that can lead to higher level annotations. El Kaed *et al.* [37] used Lua to create semantic rules for sensor decoupling when a topology change occurs (e.g., sensor being replaced or deleted). Gyrard *et al.* [7] designed the machine-to-machine measurement framework that uses data sets and sensor-based linked open rules by sharing and reusing existing ontologies/data sets/rules by domain experts. Jajaga and Ahmedi [38] proposed the C-SWRL rule language, which is an extension of SWRL but for reasoning over stream data. Ganz *et al.* [8] used a combination of machine learning and approximation techniques as well as SWRL-based rules extracted from the Web to create topical ontologies that represent domain-specific concepts. Puschmann *et al.* [32] used a combination of topic modeling and approximation techniques as well as predefined rules that describe the pattern movement of a time series for identifying underlying structures and relation in sensor data. Wei and Barnaghi [39] integrated sensor data with semantic Web and linked data for reasoning and annotation with the use of rules based on ontologies to create semantic sensor Web ontologies. Nevertheless, all these works rely on the assumption that specific rule semantics are used, complex rules will be provided by nonexpert users and that ontologies with specific schemata and semantics will either be considered or extended (e.g., semantic sensor network ontology). The only work that supports approximate rules is Wei and Barnaghi [39] but in terms of spatial proximity and not semantics, which is the scope of our work. Also, some works are domain dependent and could not be easily applied to other domains and they are too expensive for a real-time processing environment.

2) Semantic Engines in PSS: There are some works that have addressed semantic decoupling in PSS. Although, this category is not directly linked to our work; nevertheless, it is worth mentioning for completeness. Hasan and Curry [15] created an approximate semantic matcher for events coming from heterogeneous sources based on Wikipedia ESA and probabilistic models. S-TOPSS [40] use synonyms, taxonomies and mapping functions specified by domain experts for creating an approximate matcher. A-TOPSS [41] use subscriptions that match events with a degree of confidence via fuzzy set theory and probability theory. Alhakbani *et al.* [42] used approximate semantic matching in PSS for IoT based on Wikipedia ESA, a tree structure and taxonomy clustering. G-TOPSS [43] is a graph-based PSS that uses ontologies that contain synonyms, taxonomies and transformation rules to enrich data and support representational decoupling (e.g., descendant, ancestor, or direct instance of a class). Esposito *et al.* [44] enforced an RDF ontology to be dynamically built based on the incoming events in a topic-based PSS for dealing with semantic interoperability of heterogeneous events. Apart from Hasan and Curry [15], all the other works use either domain experts, ontologies or taxonomies, which infer a shared agreement of concepts, semantics and schemata, leading to the disadvantage of terminology and syntactic coupling.

V. Abstractive Publish/Subscribe Summarization System

In this section, an overview of the proposed abstractive publish/subscribe summarization system is given, like the architecture, and the event and subscription models.

A. Architecture

As aforementioned, in PSS publishers publish events and subscribers subscribe or unsubscribe to these events based on their interest. If a match is found, then, a notification is sent to the subscriber. Existing schemes cannot be applied for the scenario illustrated in Fig. 1, therefore, we propose an extension; the abstractive publish/subscribe summarization system that is based on data approximation and reasoning (IoTSAX approach) that create quad-based entity summaries (notifications) for abstractive-aware subscriptions.

The architecture of the approach is illustrated in Fig. 4. Publishers create publications concerning entities and subscribers create abstractive-aware subscriptions concerning these entities (*Subscription Model*). All publications and subscriptions enter the *abstractive publish/subscribe summarization system*, which analyses the publications and notifies the subscribers.

All publications that relate to a specific measurement of an entity are integrated into specified windows (*data integration*). As each window gets populated, dynamic PAA and dynamic SAX take place to approximate the raw data to average values and symbolic representations, respectively, (*data approximation*). Once each window reaches its full capacity based on the user-defined window size, the approximate data undergoes approximation interpretation, where the patterns and shape of the data are analyzed. It is also transformed into high-level inferences via approximate reasoning rules (*reasoning*). The abstractive summaries (*IoTSAX approach*) are, then, created in the form of quads (*event model*) that can act as notifications to the subscribers if the entity-centric matcher has found a match (*exact matching*). Then the process starts again.

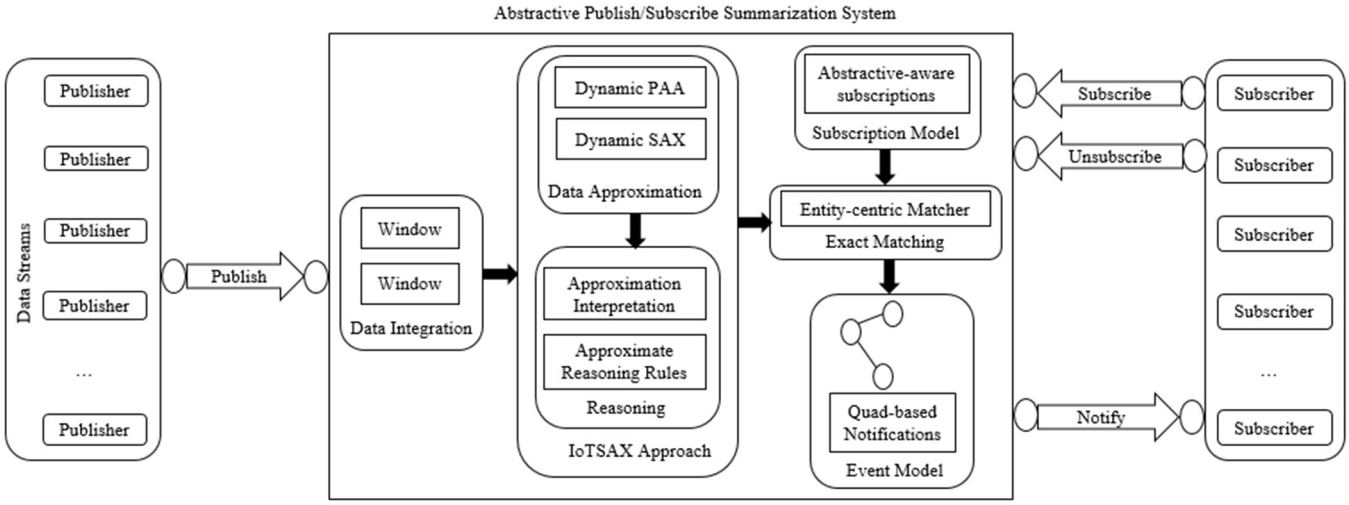


Fig. 4. Architecture of the abstractive publish/subscribe summarization system.

B. Event Model

As aforementioned, a richer but simple representation should be provided as a notification to the subscribers, in which raw sensor data turns into conceptual entities with their associated properties. Although RDF triples of the form ⟨subject, property, object⟩ could be used as a publication payload, there is no information on the high-level abstraction or pattern occurrences of the data. Therefore, we propose the adaptation of quads in the publication payload, which are of the form ⟨subject, property, object, context⟩. Below there is an example of a publication payload, where the subject is the entity in question, the object is the aggregated value within a specific window, the property is the relation between the subject and the object, and context is the pattern occurrence of raw values within a specific window and the reasoning result by the approximate rules

{<Patient> <heartRate> <103.5bpm> <overall increasing with reoccurring sharp increases and one sharp decrease, TACHYCARDIA>}

Therefore, the definition of the event model is as follows: let *EV* be the set of events, *PID* the set of publisher IDs, *PubID* the set of publication IDs, T the set of timestamps and $Q(e)$ the quad of an entity e, respectively, then

$$ev \in EV \Leftrightarrow ev = \{(\text{pID}, \text{pubID}, t, q), \ldots, : \text{pID} \in \text{PID}, \text{pubID} \in \text{PubID}, t \in T, q \in Q(e)\}. \quad (1)$$

C. Subscription Model

To support low user expressibility, the proposed subscription needs to be noncomplex and contextually aware. Therefore, the subscription payload is a set of simple attribute-value pairs and each event needs to fulfil all of its constraints so that a match occurs. Each pair consists of an attribute, an equal operator and a value. Below there is an example of a subscription payload:

{entity = "Patient," windowType = "CountTumbling," windowSize = 10, summary = "Abstraction"}

In the example above, the subscriber is interested in an abstractive summary of the entity <*Patient*> derived from the analysis of data taken from count tumbling windows of size 10. The summary's value can be either "Abstraction" or "None" should the user want no data summary.

Therefore, the definition of the subscription model is as follows: Let S be the set of subscriptions, *SID* the set of subscriber IDs, *SubID* the set of subscription IDs, T the set of timestamps, *ATT* the set of attributes, *OP* the set of operators and *VAL* the set of values, respectively, then

$$s \in S \Leftrightarrow s = \{(sID, subID, t, (att, op, val)), \ldots, : sID \in SID, subID \in SubID, t \in T (att, op, val) \in \text{ATT} \times \text{OP} \times \text{VAL}\}. \quad (2)$$

VI. IoTSAX Approach

In this section, details on the main abstractive summarization approach are given that consist of two phases: 1) IoTSAX approximation and 2) approximate rule-based reasoning.

A. Enhancing SAX Approximation

The aim of a symbolic representation of a time series can be formalized as follows.

A univariate time series $X = \langle x_1, x_2, \ldots, x_N \rangle \in \mathbb{R}^N$, that is a sequence of N data points measured at successive points in time, can be approximated into a symbolic representation $S = \langle s_1, s_2, \ldots, s_n \rangle \in \mathbb{A}^n$, where s_i is a letter of an alphabet $\mathbb{A} = \{a_1, a_2, \ldots, a_k\}$ of k letters. The approximation should:

1) boost the system performance; therefore, $n \ll N$ and $k \ll n$, meaning that the approximate sequence should have a much smaller dimension that the original one and only a small number of meaningful letters should be used;
2) satisfy the lower bounding principle;
3) provide automatically tuned parameters n and k.

1) SAX Approximation: The original SAX approximation [28] consists of three steps; z-normalization, PAA and discretization. The z-normalization ensures that all normalized values of a time series have mean and standard deviation

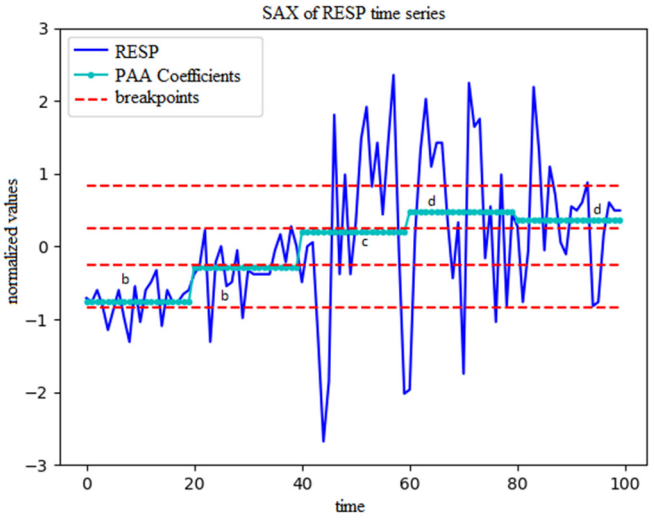


Fig. 5. SAX approximation on the first 100 respirationRate (RESP) time series samples for $n = 5$ and $k = 5$. In this case, the whole time series is approximated by the symbolic representation "bbcdd."

close to 0 and 1, respectively. PAA divides the normalized time series into user-defined equal segments and calculates the mean value of the data in each segment, which constitutes a PAA coefficient. The last step is symbolizing each PAA coefficient to a letter. This step is based on the assumption that normalized time series follow a standard normal distribution; therefore, according to a user-defined alphabet size, equiprobable regions are defined by breakpoints based on the standard normal distribution. Each PAA coefficient falls within a specific region between breakpoints that defines the letter with which it will be symbolized. An example of SAX is illustrated in Fig. 5.

2) IoTSAX Approximation: Although the original SAX approach has many aforementioned advantages compared to other approximation approaches, it is limited when it comes to real-time processing environments, like IoT. Specifically, it uses a fixed/nonadaptive segment number n and alphabet size k that is defined by the user. These parameters are data dependent; therefore, they need to be manually tuned for their best values, which is not applicable in streaming data. Also, the fixed segment number may result in poor aggregation performances as in sensor data there might be a range of data activities within a segment, from low to high data generation and small to sharp fluctuations that could indicate extreme/abnormal values. All this original information will be lost in nonadaptive segments as the average value will be defined through PAA.

Another limitation is the normalization of the raw data since the authors suggest that this is a meaningful way of comparing time series. Nevertheless, in this work reasoning and comparison among time series is based not only on their shape but on their amplitude, which is information that will be lost with normalization and it is more clearly illustrated in Fig. 6. Finally, the assumption that normalized data follows a standard normal distribution may not be the case in IoT data.

To avail of the advantages of SAX and overcome the disadvantages above, we propose IoTSAX, which is a dynamic SAX approximation that could be used in IoT environments. IoTSAX incrementally defines the best segments and alphabet size within a data window. In this way, the dynamic segments observe and retain the sharp fluctuations in the data due to extreme/abnormal values, whereas the dynamic alphabet size leads to more fine-grain representations of the approximate data based on its original skewness. Also, no data normalization takes place so that its amplitude is not lost and the true probability density function (PDF) of the data is found based on the nonparametric KDE [45]. An example of IoTSAX is illustrated in Fig. 7 and more details are given in Algorithms 1 and 2.

Algorithm 1 is the first step of IoTSAX approximation, where the best segments and PAA coefficients are found incrementally. In line 1, the DynPAA function gets as input two consecutive data points, whereas in line 3, the two points are transformed into a linear equation and the slope or derivative of the equation is calculated. The slope is later enhanced in line 4 by taking into account the amplitude between the two points. A difference between two consecutive enhanced slopes is found in line 5 and if it is higher than a user-defined threshold (line 6) then the end of a dynamic segment has been found (line 12). Nevertheless, we avoid splitting consecutive sharp enhanced slopes that are first negative and then positive and vice versa into different dynamic segments since they could be represented together by a mean value (PAA coefficient) in lines 7–11. Lines 14 and 15 are necessary updates of a possible end of a dynamic segment and enhanced slope for the next incremental rounds. Lines 17–19 dictate that if the window has reached its end, then, we have reached the end of the final dynamic segment. Lines 20–22 show that if an end of a dynamic segment has been found, then, the average value of the points within the segment constitutes its PAA coefficient.

Algorithm 2 is the last step of IoTSAX approximation, where the best dynamic alphabet size is defined as well as the PDF of the data. In line 1, the IoTSAX function gets as input the window data points X, the PAA coefficients of the window $\bar{X}$ from DynPAA function, a system-defined alphabet $\mathbb{A} = \{a, \ldots, z\}$ and $k\text{Max} = 10$ and $k\text{Min} = 3$ that define the maximum and minimum values, respectively, that the dynamic alphabet size can take. The alphabet $\mathbb{A}$ could apply to any environment. Also, according to the original SAX authors [28] and other studies [31], the range of an alphabet size is not too critical and a range of 5–8 seems to yield the best results, therefore we set the alphabet size range from 3 to 10. If the end of the window has been reached (line 2), then, KDE is called that through the calculation of mean and standard deviation of X, a necessary bandwidth is defined and the final PDF of the data is calculated based on a Gaussian kernel (line 3). Line 4 calculates the skewness of X based on its mean, mode (most frequent value) and standard deviation. The skewness shows the asymmetry of the PDF compared to its mean value. The higher the skewness, the lower the dynamic alphabet size, which is calculated in line 5 based also on the *kMax* and *kMin*. All skewness values higher than 1 are

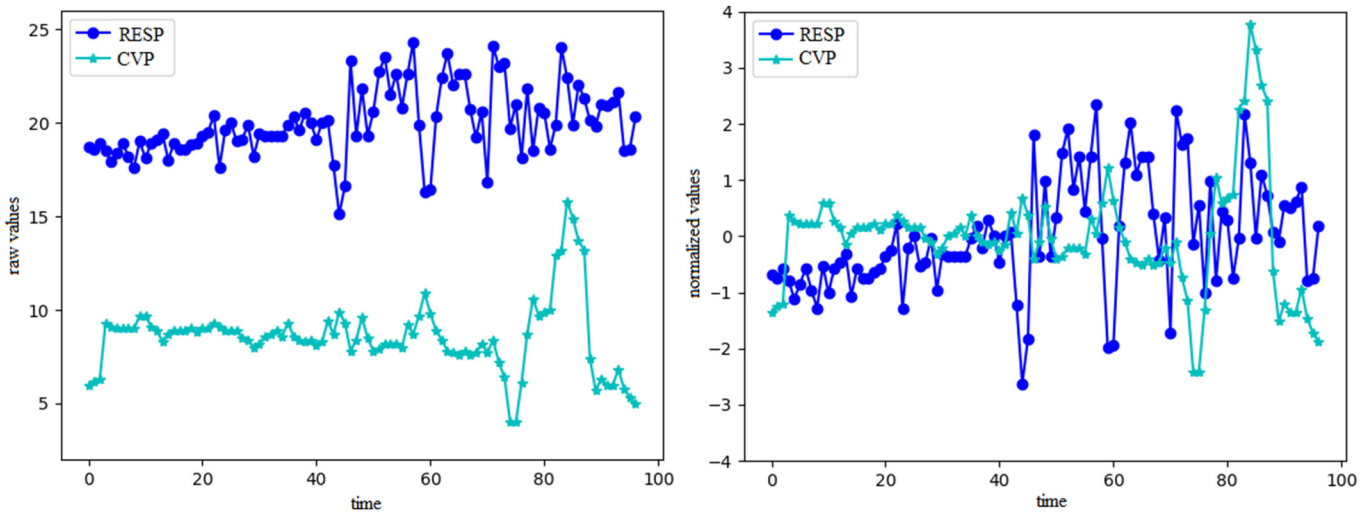


Fig. 6. Normalization maintains the shape of the original time series but loses the amplitude, which is important in sensor data comparison and reasoning. In (b) distance between the time series seems smaller compared to what it actually is as depicted in (a) and the original values that would be used in reasoning are lost. (a) Difference between raw RESP and CVP time series. (b) Difference between normalized RESP and CVP time series.

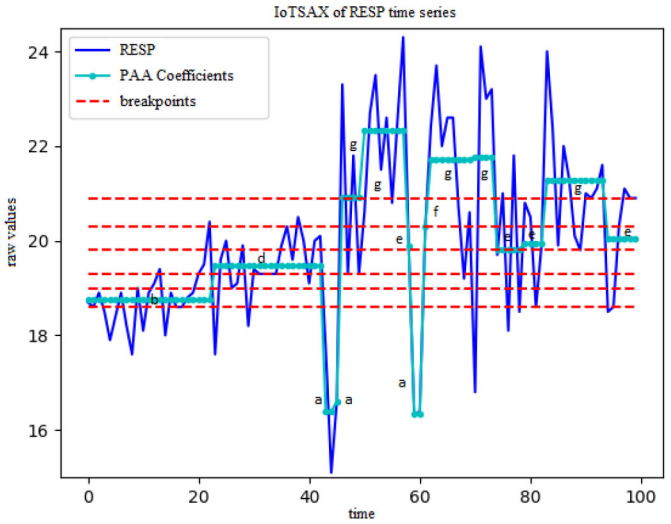


Fig. 7. IoTSAX approximation on the first 100 RESP time series for thr = 4, kMax = 10 and kMin = 3. The segments follow the sharp fluctuations of the data compared to Fig. 5 and the skewness-based alphabet size $k = 7$ leads to a fine grain representation of the approximate data, which in this case it is symbolically represented by "bdaaggeafggeege."

considered equal to 1. The skewness may change for different window data points resulting in different alphabet sizes. Lines 6–9 define the breakpoints based on the logic of the original SAX. Specifically, the β variables define the cumulative probabilities that are required if the area under PDF is split into equiprobable regions of a number equal to the alphabet size. Then, the *breakpoints* are the values where these probabilities occur. Finally, lines 10–16 also follow the logic of the original SAX, where according to which specific region between breakpoints the PAA coefficient falls within, a letter is selected for the symbolic representation S of X. An example of the PDF via KDE of Fig. 7 compared to the normal distribution is illustrated in Fig. 8.

Algorithm 1 DynPAA—Dynamic PAA

1: **function** DynPAA(x_{N-1}, x_N, thr) ▷ x_i: data point
2: $dynSegmentsPoint \leftarrow 0$
3: $slope \leftarrow (x_N - x_{N-1})/(index(x_N) - index(x_{N-1}))$
4: $enSlope \leftarrow slope * |x_N - x_{N-1}|$
5: $diffOfEnSlopes \leftarrow ||enSlope| - |prEnSlope||$
6: **if** $diffOfEnSlopes > thr$ **then**
7: **if** $index(x_{N-1}) - prDynSegmentsPoint = 1$ **then**
8: **if** $prEnSlope * enSlope > 0$ **then**
9: $dynSegmentsPoint \leftarrow index(x_{N-1})$
10: **end if**
11: **else**
12: $dynSegmentsPoint \leftarrow index(x_{N-1})$
13: **end if**
14: $prDynSegmentsPoint \leftarrow index(x_{N-1})$
15: **end if**
16: $prEnSlope \leftarrow enSlope$
17: **if** $endOfWindow$ **then**
18: $dynSegmentsPoint \leftarrow index(x_N)$
19: **end if**
20: **if** $dynSegmentsPoint \neq 0$ **then**
21: $\bar{X}_i \leftarrow \frac{1}{numOfPoints} \sum_{j=endOfPreviousSegment}^{dynSegmentsPoint} X_j$
22: **end if**
23: **return** $\bar{X}_i$ ▷ PAACoefficient (if any)
24: **end function**

B. Reasoning

Reasoning is the final step in the time series analysis in order to create the quad. It consists of two main parts, which is the interpretation of the symbolic representation and the approximate reasoning rules. More details are given in Algorithm 3.

Algorithm 3 depicts all the steps that create the final quad that is sent as a notification to the subscriber. In line 1, the Quad function gets as input the PAA coefficients of the

Algorithm 2 IoTSAX—Dynamic SAX for IoT

1: **function** IoTSAX($X, \bar{X}, \mathbb{A}, kMax, kMin$) ▷ X: window data points
2: **if** *endOfWindow* **then**
3: $PDF \leftarrow KDE(X)$
4: $skewness \leftarrow |mean - mode|/sd$
5: $k \leftarrow round(kMax - ((kMax - kMin) * skewness))$
6: **for** $i \leftarrow 0, k-1$ **do**
7: $\beta \leftarrow (i+1) * totalAreaUnderPDF/k$
8: $breakpoints \leftarrow PDF.quantileFunction(\beta)$
9: **end for**
10: **for** $i \leftarrow 0, sizeOf\bar{X}$ **do**
11: **for** $j \leftarrow 0, sizeOfBreakpoints$ **do**
12: **if** $breakpoints_j > \bar{X}_i$ **then**
13: $S_i \leftarrow \mathbb{A}_j$
14: **end if**
15: **end for**
16: **end for**
17: **end if**
18: **return** S ▷ Symbolic representation of X
19: **end function**

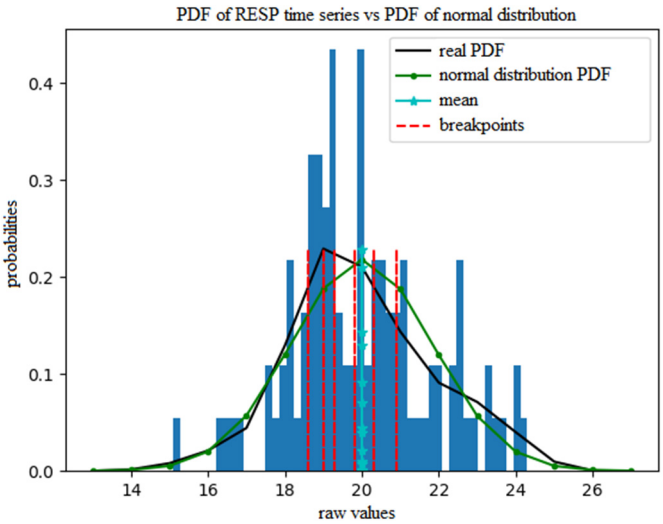


Fig. 8. PDF via KDE on the first 100 RESP time series. The histogram of the raw values and the estimated PDF show that it does not follow the normal distribution, which would be the assumption in the original SAX (here no normalization occurs, otherwise the normal distribution would have mean = 0 as in the original SAX). The distribution is skewed with *skewness* = 0.38 resulting in $k = 7$ with breakpoints that split the area under PDF in k equiprobable regions.

window $\bar{X}$ and the symbolic representation S from IoTSAX function. In lines 2–4, the main triple is created that consists of the subject and the property of the window, as well as the object, which is the mean value of the window's PAA coefficients. In lines 5–7, the distance of subsequent letters in the symbolic representation is calculated and meaning is defined. For example, for $k = 3$, if the distance is -1 then the meaning is that there is a slight increase, if it is -2 then there is a sharp increase etc. The values that *meaningOfDistance* can take are {slightIncrease, increase, sharpIncrease, slightDecrease, decrease, sharpDecrease}. In line 8, an interpretation

Algorithm 3 Quad—Reasoning

1: **function** QUAD($\bar{X}, S$) ▷ $\bar{X}$: window PAACoefficients
2: $subject \leftarrow entityOfWindow$
3: $property \leftarrow attributeOfWindow$
4: $object \leftarrow mean(\bar{X})$
5: **for** $i \leftarrow 1, sizeOf\bar{X}$ **do**
6: $meaningOfDistance \leftarrow distance(S_i, S_{i-1})$
7: **end for**
8: $interpretation \leftarrow shapeOfS(meaningOfDistance)$
9: $inference \leftarrow approximateRules(property, object)$
10: $quad \leftarrow< subject >< property >< object >< interpretation, inference >$
11: **return** *quad* ▷ quad of event
12: **end function**

of the shape of the symbolic representation is extracted based on the meaning of distances. According to the popularity of the meanings, an initial interpretation may have one of the following values {"overall increasing," "overall slightly increasing," "overall decreasing," "overall slightly decreasing," "overall neutral}." Then, based on whether they have sharp increases/decreases, their sequence and their frequency, the interpretation may additionally have some of the following values {"one sharp increase/decrease followed by one sharp decrease/increase" or "reoccurring sharp increases/decreases followed by sharp decreases/increases," "one sharp increase" or "reoccurring sharp increases," "one sharp decrease" or "reoccurring sharp decreases}." Line 9 refers to the reasoning result based on approximate manual rules. Specifically, a property contains a set of rules that define the context to be inferred depending on the property's value, which in this case is the object. These rules are approximate; therefore, if a conceptually related property to the one existing in the rules were to be used, then, the reasoner would be able to find the relevance and use the appropriate rule. Finally, in line 10, a quad is created that apart from the triple, it also contains the contextual information.

1) Approximate Reasoning Rules: Rule-based reasoning is a simple, time-efficient and low-memory way to infer high-level abstractions of sensor data. Nevertheless, one of their disadvantages is that they cannot be adapted to heterogeneous and complicated sensors. This problem can be addressed by approximate reasoning rules via word embeddings, which represent words into vectors in a vector space. Words that are more conceptually related, including semantically opposite words like antonyms, exist closer to the vector space.

The stages of finding the most relevant property existing in the rules (Fig. 9) are the following.

1) *Preprocessing:* Each property in a triple is preprocessed by lower casing, tokenising, removing stop-words and concatenating each token with an underscore.
2) *Property to Vector:* The original word, the tokens and the concatenated word are sent to a word embedding model to extract their vector. Priority is given to the

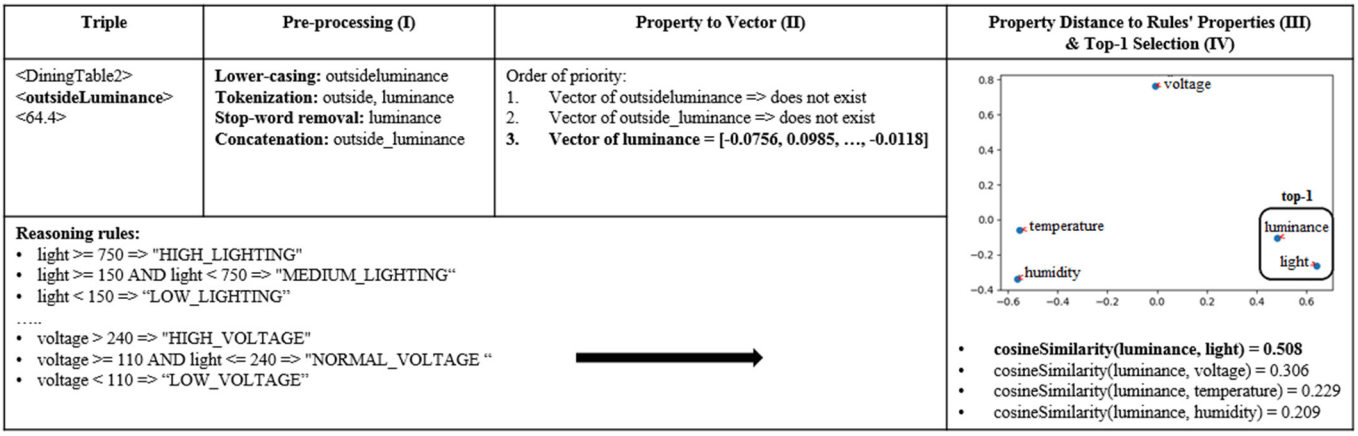


Fig. 9. Example of the stages of approximate reasoning rules. Only a subset of all the reasoning rules and their properties is given for visualization purposes. Rigid reasoning rules would not relate the property "outsideLuminance" to "light," therefore, no reasoning result would occur based on the semantically coupled reasoning rules.

original word, then, the concatenated one and finally if the previous do not exist, then, an average is taken by all the tokens' vectors. An index is created for all words that have already been examined along with their vectors.

3) *Property Distance to Rules' Properties:* A distance (cosine similarity) is calculated between the property in a triple and all properties existing in the reasoning rules. All distances are sorted from highest to lowest.
4) *Top-1 Selection:* The rules' property that has the highest similarity is picked as the most relevant property.

VII. Evaluation

This section provides the evaluation of the approach that consists of the methodology followed and the data sets used as well as the metrics and the final results.

A. Data Sets and Methodology

A combination of different real-world data sets has been used that relate to the applications of medicine and smart cities. MIMIC II database[2] contains heart-related readings (e.g., heart rate, central venous pressure etc.) of patients that developed or were at risk of developing an acute hypotensive episode. Intel Lab Data[3] contains indoor sensor readings (e.g., temperature, humidity etc.) from the Intel Berkeley Research lab. CityPulse[4] contains readings regarding the weather (e.g., dew point, wind speed etc.), pollution (e.g., ozone, carbon monoxide etc.) and road traffic (e.g., vehicle count). UCI electric consumption[5] contains electric power consumption readings in one household (e.g., global active power, voltage etc.). All readings were preprocessed into literals with properties the measurements centralVenousPressure, temperature etc., objects the original values and subjects the property/people's name. Missing values and final zero values, due to equipment disconnection or noise, were deleted. Other kinds of noise (e.g., white noise) in the original data were ignored as out of the scope of the work. The reader is directed to Vaseghi [46] on advanced signal processing techniques. Also, where possible, spatially nearby sensors were selected. An overview of the data sets and their characteristics is given in Table II.

In order to observe the consensus among the results of different abstractive publish/subscribe summarization systems, we examined ten entities, where each corresponds to a separate system. For the medicine application, we used ten patient data from MIMIC II database, whereas for the smart cities one, we synthetically combined for each entity different 24-h sets from all CityPulse and UCI electric consumption data as well as 44 spatially nearby sensor data from Intel Lab Data. We used a representative subset of the MIMIC II database and Intel Lab Data since the data contain redundancy. We also examined all the original attributes' semantics generated by the sensors. The manual rules were mostly created based on the M3 framework's rules[6] and the annotations in the MIMIC II database, which contained ranges of attributes' values when an alert occurred.

The evaluation methodology is illustrated in Fig. 10. It examines both the efficiency and effectiveness of the approach. The efficiency involves metrics like latency, throughput, etc. after the approximation and reasoning take place. The effectiveness involves metrics like F-score and approximation error. The F-score is based on a relevance ground truth (partly inspired by Hasan and Curry [15] ground truth construction). In order to show the heterogeneity involved in IoT environments, a ground truth set is constructed by choosing each original data property (e.g., light) and storing its synonyms (e.g., blaze), related words (e.g., flash) and antonyms (e.g., blackness) deriving from the Merriam-Webster[7] thesaurus as well as its hyponyms (children) observed in the BRICK ontology[8] (e.g., luminance). In the case of polysemy, the words that

[2]https://archive.physionet.org/challenge/2009/training-set.shtml
[3]http://db.csail.mit.edu/labdata/labdata.html
[4]http://iot.ee.surrey.ac.uk:8080/datasets.html
[5]https://archive.ics.uci.edu/ml/datasets/Individual+household+electric+power+consumption
[6]https://github.com/gyrard/M3Framework/tree/master/war/RULES
[7]https://www.merriam-webster.com/thesaurus
[8]https://brickschema.org/#home

TABLE II
DATA SET CHARACTERISTICS

Dataset	Domain	Sensors	Number of Sensors Used	Number of Events per Entity (Average)	Original Schema	Pre-processing
MIMIC II Database	Medicine	heart readings	11	8.8K	attribute-value	literals, missing values/final zeros deleted
Intel Lab Data	Smart Cities	temperature, voltage, light, humidity	44	262.9K	attribute-value	literals, missing light values replaced with nearby values, spatially nearby sensors selected
CityPulse	Smart Cities	weather, pollution, road traffic	13	187	attribute-value	literals
UCI Electric Consumption	Smart Cities	household electric readings	7	200.2K	attribute-value	literals

applied to the application in use were chosen (e.g., "pressure" as body measurement in the medical case and as environmental one in the smart cities case). An example of the evaluation methodology is given in Fig. 10 for a triple with property light.

The streaming rate by the publishers and subscribers follows a uniform distribution as in the real case. Each publisher generates a stream related to a measurement (e.g., temperature) of an entity. There is only one subscriber that generates subscriptions based on the entities. All runs took place in a laptop with Intel Core i7-6600U CPU@2.60 GHz 2.80 GHz and 16 GB of RAM. Regarding the implementation, for the RDF models, we use Apache Jena,[9] for the embedding model we use deeplearning4j,[10] for the KDE we modified the corresponding class by Smile,[11] for tagging and lemmatization we use Apache OpenNLP.[12]

B. Metrics

Several metrics have been used to evaluate the effectiveness and efficiency of the approach. The effectiveness evaluation focuses on the correctness, whereas the efficiency one focuses on the performance.

1) Correctness: Correctness consists of the F-score and approximation error metrics.

a) F-Score: The F-score metric calculates the effectiveness of the approximate reasoner. Specifically, each property in the ground truth is linked to an original triple's property (Fig. 10), therefore, if the reasoner finds this relevance, then, it is considered as true positive (TP), otherwise, it is a false positive (FP) as well as a false negative (FN), since the original property was found irrelevant, but was indeed relevant according to the ground truth. For example, in the case of Fig. 9 the "outsideLuminance" property is correctly found most related to the property light therefore, the F-score $= 1.0$, otherwise F-score $= 0.0$. Specifically, the F-score is defined as

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad \text{and} \quad \text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

$$\text{F} - \text{score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}. \tag{4}$$

b) Approximation Error: Based on the study of Ding *et al.* [29] the most important time series distance measures are the Euclidean distance (ED) and the full dynamic time warping (DTW). The authors also found that the two measurements complement each other as for big data sets the accuracy of ED is higher, whereas for small data sets the DTW's accuracy is higher. Therefore, these two measures are both used to define the distance between the original time series and the approximated one via PAA.

2) Performance: The performance consists of the end-to-end latency, size reduction, compression space saving and throughput metrics analyzed below.

a) End-to-End latency: The end-to-end latency is the time it takes between the publication of an event until its delivery to the subscriber as a notification. Since our summaries/notifications involve multiple publications, our end-to-end latency is the time it takes between the earliest published event in the fusion until the time of the summary's delivery to the subscriber.

b) Size reduction: This metric is the reduction in the number of messages that the subscriber receives.

c) Compression space-saving: This metric is the percentage of the reduction in data space occurring via approximation and symbolic representation.

d) Throughput: Throughput is the number of triples/events the system is able to analyze in a specific amount of time.

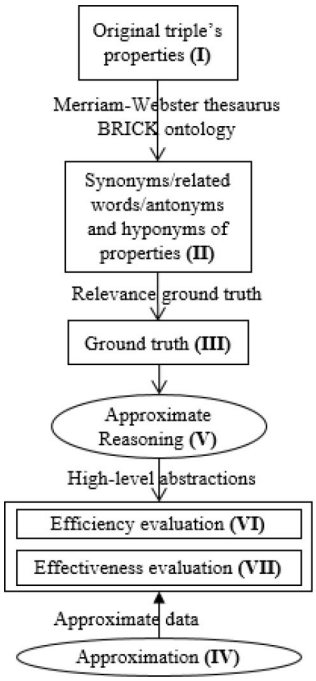
C. Results

The results of the IoTSAX approach are analyzed below. An example of the abstractive summaries produced is illustrated in Fig. 11.

1) Data Approximation: The correctness and performance of data approximation are analyzed for the following approaches: 1) the typical PSS approach (no fusion no abstraction), where all events regarding an entity are sent as a notification to the subscriber with no approximation or reasoning involved; 2) SAX approximation with reasoning (fusion SAX abstraction), where according to the original authors [28] and other studies [31] the use of $n = 5$ to 8 and $k = 5$ to 8 generally yields the best results; and 3) our proposed IoTSAX approximation with reasoning (fusion IoTSAX abstraction), where different threshold values are used as well as $k\text{Max} = 10$ and $k\text{Min} = 3$.

The results are shown below for ten entities for each use case. For the medicine's use case, a combination of 6 to 11 publishers is used depending on the availability of the data for each entity. For the smart cities use case, 64 publishers are

[9]https://jena.apache.org/
[10]https://deeplearning4j.org/
[11]https://haifengl.github.io/smile/nlp.html
[12]https://opennlp.apache.org/



Construction of ground truth

I	II	III
light	**SYNONYMS:** blaze, flare, fluorescence etc. **RELATED WORDS:** flash, glimmer, glint etc. **ANTONYMS:** blackness, dark, darkness etc. **BRICK'S HYPONYMS:** luminance, lightingSystemLuminance, outsideLuminance, lightingSystem	Map{"light": ["light, blaze, flare, …, lightingSystem"]}

Approximation & Approximate Reasoning

IV	V	VI	VII
• PAA coefficients • Symbolic representation	**Top-1:** "light" **Rules:** • light >= 750 => "HIGH_LIGHTING" • light >= 150 AND light < 750 => "MEDIUM_LIGHTING" • light < 150 => "LOW_LIGHTING"	• latency • size reduction • compression space saving • throughput	• f-score • approximation error (Euclidean distance, full Dynamic Time Warping)

Fig. 10. Evaluation methodology and an example for a triple with property "light."

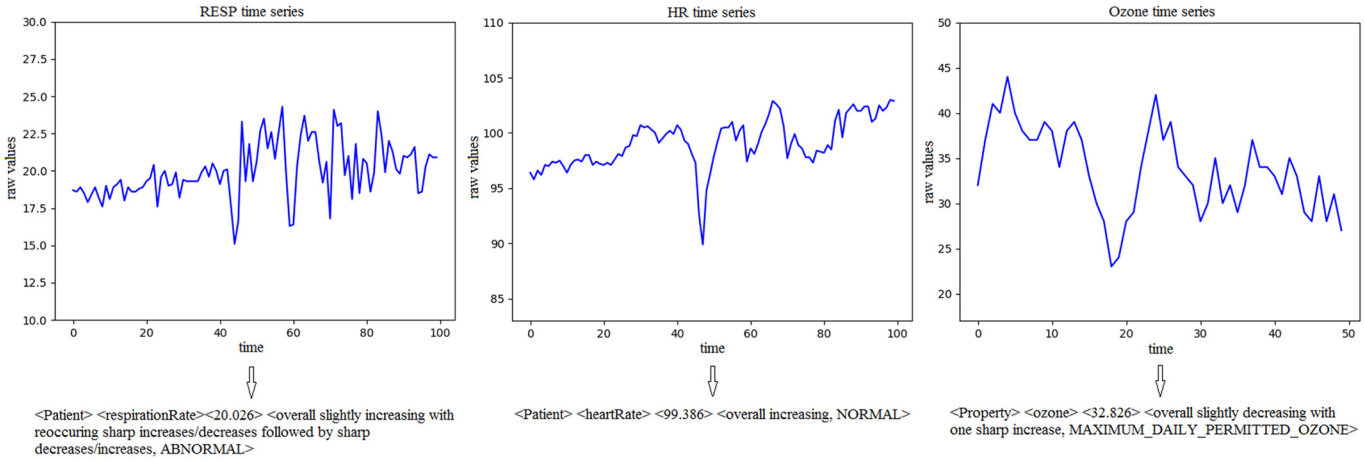


Fig. 11. Quad-based notification examples for RESP, HR, and ozone time series, respectively.

used for each entity. Each publisher is related to a measurement. The window sizes differ from 100 to 50 for medicine and smart cities, respectively. The duration of each experiment ranges from 15 to 30 min depending on the availability of the data.

a) End-to-End latency and throughput: In Tables III and IV the end-to-end latency and throughput results are illustrated, respectively. No fusion no abstraction has the best latency as no approximation or reasoning as well as waiting for a window to be populated are involved when sending notifications. Surprisingly, this has no effect on the throughput showing that a fast approach that generates an overload of notifications can have a negative effect on the performance of a system. The abstraction approaches can achieve from 2 to 3 times more the latency of no fusion no abstraction one as further processing and population of windows are involved. The medicine's use case has higher latency; therefore, lower throughput compared to the smart cities one, because fewer publishers, hence less processing windows are created (less parallelism) and the window size involved is bigger showing the effect both parallelism and window size can pose in the system performance.

In terms of the comparison among the abstraction approaches, there is no significant consensus that indicates which approach or parameter is better. Specifically, for different SAX parameters, the processing is similar time-wise; therefore, it should pose no significant system performance differentiation. This is also depicted in the throughput results showing that indeed SAX and IoTSAX are appropriate approaches to be used in a streaming environment. Nevertheless, we observe that IoTSAX achieves similar or better latencies compared to SAX. We believe that the bigger and more complex the computations are, the better the latency will be in the IoTSAX approach as it is incremental

TABLE III
AVERAGE END-TO-END LATENCY RESULTS FOR BOTH USE CASES

Use Case	Approach	Parameters	Average end-to-end latency in ms for each entity									
			E0	E1	E2	E3	E4	E5	E6	E7	E8	E9
Medicine	no fusion no abstraction		**21537**	**11143**	**22108**	**23227**	**36355**	**30171**	**20649**	**23290**	**30186**	**32119**
			Average = 25078.5, Standard Deviation = 6859.736									
	fusion SAX abstraction	n = 5, k = 5 windowSize = 100	61475	62213	62710	63954	65937	65070	63304	68543	61509	67088
			Average = 64180.3, Standard Deviation = 2295.018									
		n = 5, k = 8 windowSize = 100	61005	57264	63554	66322	63882	69938	66484	58642	62605	64939
			Average = 63463.5, Standard Deviation = 3606.05									
		n = 8, k = 5 windowSize = 100	64280	58340	66329	67186	66691	64569	62459	65641	61901	62966
			Average = 64036.2, Standard Deviation = 2564.829									
		n = 8, k = 8 windowSize = 100	70391	58308	68330	64979	61405	66293	63423	65698	67282	64106
			Average = 65021.5, Standard Deviation = 3294.044									
	fusion IoTSAX abstraction	thr = 1 windowSize = 100	69414	58590	65360	62465	66550	64522	62202	67280	63552	62015
			Average = 64195.0, Standard Deviation = 2954.864									
		thr = 4 windowSize = 100	66372	57717	67063	66220	63700	69760	63479	68776	64370	64856
			Average = 65231.3, Standard Deviation = 3185.447									
Smart Cities	no fusion no abstraction		**11684**	**11275**	**11468**	**11156**	**11257**	**10817**	**9976**	**11933**	**10854**	**10174**
			Average = 11059.4, Standard Deviation = 589.385									
	fusion SAX abstraction	n = 5, k = 5 windowSize = 50	43029	36424	39507	40429	40250	41655	40140	38725	42147	40649
			Average = 40295.5, Standard Deviation = 1760.513									
		n = 5, k = 8 windowSize = 50	44582	40787	40020	38790	40477	39157	39618	40081	39714	38403
			Average = 40162.9, Standard Deviation = 1628.99									
		n = 8, k = 5 windowSize = 50	42298	39677	41441	38651	40798	40956	40001	40450	40881	41388
			Average = 40654.1, Standard Deviation = 972.568									
		n = 8, k = 8 windowSize = 50	41452	42174	41953	39614	39451	40812	40594	41071	38856	37174
			Average = 40315.1, Standard Deviation = 1468.351									
	fusion IoTSAX abstraction	thr = 0.5 windowSize = 50	41263	43723	40376	38152	42016	39230	40947	38593	40561	40732
			Average = 40559.3, Standard Deviation = 1557.831									
		thr = 1 windowSize = 50	41643	39676	37395	39771	42256	41271	40454	39791	40475	40907
			Average = 40363.9, Standard Deviation = 1277.564									

TABLE IV
THROUGHPUT RESULTS FOR BOTH USE CASES

Use Case	Approach	Parameters	Throughput in events per second for each entity									
			E0	E1	E2	E3	E4	E5	E6	E7	E8	E9
Medicine	no fusion no abstraction		22.117	19.673	22.052	20.021	13.907	15.980	21.656	22.033	13.231	16.064
			Average = 18.673, Standard Deviation = 3.357									
	fusion SAX abstraction	n = 5, k = 5 windowSize = 100	22.026	19.673	22.089	19.917	14.069	16.099	21.651	22.077	13.231	15.999
			Average = 18.683, Standard Deviation = 3.324									
		n = 5, k = 8 windowSize = 100	21.993	19.673	21.911	20.110	14.037	15.937	21.656	21.864	13.232	15.949
			Average = 18.636, Standard Deviation = 3.311									
		n = 8, k = 5 windowSize = 100	22.068	19.673	22.006	20.094	13.976	16.013	21.646	22.063	13.231	15.994
			Average = 18.676, Standard Deviation = 3.345									
		n = 8, k = 8 windowSize = 100	22.054	19.673	21.944	19.999	13.987	16.096	21.649	22.053	13.231	16.021
			Average = 18.671, Standard Deviation = 3.323									
	fusion IoTSAX abstraction	thr = 1 windowSize = 100	22.066	19.673	22.024	19.878	12.015	16.016	21.625	22.113	13.231	15.939
			Average = 18.458, Standard Deviation = 3.657									
		thr = 4 windowSize = 100	22.042	19.673	22.089	19.987	12.044	16.057	21.648	21.956	13.231	15.963
			Average = 18.469, Standard Deviation = 3.643									
Smart Cities	no fusion no abstraction		53.070	95.209	97.319	94.058	92.289	88.802	87.993	88.742	88.849	87.766
			Average = 87.41, Standard Deviation = 11.885									
	fusion SAX abstraction	n = 5, k = 5 windowSize = 50	53.070	95.070	97.236	94.139	92.171	88.551	87.866	88.700	88.781	88.147
			Average = 87.373, Standard Deviation = 11.866									
		n = 5, k = 8 windowSize = 50	53.070	95.221	97.267	94.436	92.206	88.704	87.806	88.513	88.943	87.904
			Average = 87.407, Standard Deviation = 11.897									
		n = 8, k = 5 windowSize = 50	53.070	95.488	97.472	94.283	92.159	88.816	87.839	88.740	88.839	88.137
			Average = 87.484, Standard Deviation = 11.925									
		n = 8, k = 8 windowSize = 50	53.070	95.109	97.594	94.203	92.149	88.802	87.870	88.574	88.762	88.024
			Average = 87.416, Standard Deviation = 11.902									
	fusion IoTSAX abstraction	thr = 0.5 windowSize = 50	53.070	95.310	96.963	93.991	92.261	88.613	87.616	88.748	88.826	87.896
			Average = 87.329, Standard Deviation = 11.854									
		thr = 1 windowSize = 50	53.070	95.377	97.393	94.083	92.568	88.853	87.998	88.936	89.026	87.893
			Average = 87.52, Standard Deviation = 11.921									

compared to SAX. There might also be a possibility that KDE posed an additional computational burden in the case of IoTSAX, hence the similar latencies with SAX in some entities. Finally, we observe that stricter thresholds (lower thresholds) in IoTSAX show slightly lower latencies as more segmentation takes place.

It is important to note that we checked how the system performs if no abstraction takes place, but with fusion, meaning that no approximation or reasoning is involved but the latencies and throughput are affected by the time each window takes to be populated and we observed no significant differentiation in the results. This shows how

TABLE V
SIZE REDUCTION RESULTS FOR BOTH USE CASES

Use Case	Approach	Parameters	Forwarded messages for each entity									
			E0	E1	E2	E3	E4	E5	E6	E7	E8	E9
Medicine	no fusion no abstraction		26345	8737	26476	23911	14415	19190	21980	26340	13960	19091
			Average = 20044.5, Standard Deviation = 5810.345									
	fusion abstraction	windowSize = 50	526.90	174.74	529.52	478.22	288.30	383.80	439.60	526.80	279.20	381.82
			Average = 400.89, Standard Deviation = 116.207									
		windowSize = 100	263.45	87.37	264.76	239.11	144.15	191.90	219.80	263.40	139.60	190.91
			Average = 200.445, Standard Deviation = 58.103									
		windowSize = 150	**175.63**	**58.25**	**176.51**	**159.41**	**96.10**	**127.93**	**146.53**	**175.60**	**93.07**	**127.27**
			Average = 133.63, Standard Deviation = 38.735									
Smart Cities	no fusion no abstraction		44385	60414	62467	62363	58323	61030	58262	57138	57087	60660
			Average = 58212.9, Standard Deviation = 4977.332									
	fusion abstraction	windowSize = 50	887.70	1208.28	1249.34	1247.26	1166.46	1220.60	1165.24	1142.76	1141.74	1213.20
			Average = 1164.258, Standard Deviation = 99.547									
		windowSize = 100	443.85	604.14	624.67	623.63	583.23	610.30	582.62	571.38	570.87	606.60
			Average = 582.129, Standard Deviation = 49.773									
		windowSize = 150	**295.90**	**402.76**	**416.45**	**415.75**	**388.82**	**406.87**	**388.41**	**380.92**	**380.58**	**404.40**
			Average = 388.086, Standard Deviation = 33.182									

TABLE VI
AVERAGE COMPRESSION SPACE-SAVING PERCENTAGE RESULTS FOR BOTH USE CASES

Use Case	Approach	Parameters	Average compression space-saving in % for each entity									
			E0	E1	E2	E3	E4	E5	E6	E7	E8	E9
Medicine	fusion SAX abstraction	n = 5 windowSize = 100	**95**	**95**	**95**	**95**	**95**	**95**	**95**	**95**	**95**	**95**
			Average = 95.0, Standard Deviation = 0.0									
		n = 8 windowSize = 100	92	92	92	92	92	92	92	92	92	92
			Average = 92.0, Standard Deviation = 0.0									
	fusion IoTSAX abstraction	thr = 1 windowSize = 100	76.69	71.75	76.66	82.31	86.51	71.99	75.79	80.54	78.09	76.17
			Average = 77.65, Standard Deviation = 4.28									
		thr = 4 windowSize = 100	85.83	82.35	86.78	89.14	92.043	81.32	85.73	89.02	83.97	81.46
			Average = 85.764, Standard Deviation = 3.4									
Smart Cities	fusion SAX abstraction	n = 5 windowSize = 50	90	90	90	90	90	90	90	90	90	90
			Average = 90.0, Standard Deviation = 0.0									
		n = 8 windowSize = 50	84	84	84	84	84	84	84	84	84	84
			Average = 84.0, Standard Deviation = 0.0									
	fusion IoTSAX abstraction	thr = 0.5 windowSize = 50	92.11	93.88	93.74	94.19	93.84	92.90	92.21	92.21	93.96	92.51
			Average = 93.155, Standard Deviation = 0.801									
		thr = 1 windowSize = 50	**93.18**	**94.63**	**94.46**	**94.99**	**94.38**	**93.60**	**92.92**	**92.84**	**94.56**	**93.37**
			Average = 93.893, Standard Deviation = 0.753									

powerful SAX and IoTSAX approximation are in a streaming environment.

b) Size reduction: In Table V, we see that when abstraction is used the number of forwarded messages sent to the subscriber can be reduced from 98% to more than 99% depending on the window size. The smaller the window size, the lower the reduction. This complies with the latency results that show that although no abstraction approach can be up to three times faster than the abstraction ones, it creates an overload of notifications that can overwhelm both the system and the subscriber. We also observe that more messages are forwarded in the case of smart cities that also complies with its lower latency and higher throughput as there is more parallelism and more publishers generating events.

c) Compression space-saving and approximation error: In Tables VI and VII the average compression space saving and the overall approximation error of all time series are illustrated, respectively.

In terms of the compression space saving, we observe that in the case of SAX the compression is higher for smaller segment numbers (n) as more data is grouped in segments to be approximated. We also see that since n is static for all entities in SAX, there is no differentiation in space saving compared to the dynamic approach of IoTSAX, where the segmentation is based on the fluctuations of the data of each entity. This is highly depicted in the space-saving percentage between medicine's use case and smart cities one. In medicine, SAX performs higher compression compared to IoTSAX as more fluctuations of data exist (patient's measurements may have extreme values); therefore, more segmentation takes place in IoTSAX resulting in smaller space saving. On the other hand, in the case of smart cities, there are not many fluctuations in sensor readings (the readings may be the same for a window); therefore, less segmentation occurs in IoTSAX compared to SAX that has a static segmentation number. This means that even in redundant sensor readings, SAX will always create a bigger symbolic representation compared to IoTSAX; therefore, saving less space. We should observe though that in IoTSAX the percentage of space saving is not only dependent on the data, but also the threshold, suggesting that smaller thresholds perform stricter dynamic segments, hence less compression space saving. Finally, we see that the higher the window, the more the effect of the compression in saving space.

In terms of the approximation error, we observe how well (nonnormalized in the case of SAX) PAA coefficients behave compared to the original data. We see that ED and DTW

TABLE VII
APPROXIMATION ERROR VIA ED AND DTW RESULTS FOR BOTH USE CASES

Use Case	**Approach**	**Parameters**	**Overall Euclidean distance for each entity**									
			E0	E1	E2	E3	E4	E5	E6	E7	E8	E9
Medicine	fusion SAX abstraction	n = 5 windowSize = 100	21169	5328	31222	25435	4501	19790	16909	19923	16171	21679
			Average = 18212.7, Standard Deviation = 7794.009									
		n = 8 windowSize = 100	19594	4943	28211	24476	4182	17531	15975	18535	14855	19653
			Average = 16795.5, Standard Deviation = 7161.622									
	fusion IoTSAX abstraction	thr = 1 windowSize = 100	**11936**	**2857**	**13098**	**16270**	**2101**	**7648**	**9733**	**11318**	**10231**	10584
			Average = 9577.6, Standard Deviation = 4147.599									
		thr = 4 windowSize = 100	12331	3060	13736	16801	2455	8120	10198	11729	10264	**10290**
			Average = 9898.4, Standard Deviation = 4211.284									
Smart Cities	fusion SAX abstraction	n = 5 windowSize = 50	11162	12398	15030	11089	22488	43026	19637	20162	13203	23566
			Average = 19176.1, Standard Deviation = 9103.185									
		n = 8 windowSize = 50	8304	9523	11525	8178	17695	34960	15360	15932	9485	18839
			Average = 14980.1, Standard Deviation = 7650.237									
	fusion IoTSAX abstraction	thr = 0.5 windowSize = 50	**4033**	**5103**	**7845**	**4091**	**9443**	**21318**	**9190**	**9680**	**5910**	**10232**
			Average = 8684.5, Standard Deviation = 4772.718									
		thr = 1 windowSize = 50	4220	5297	7870	4158	9488	21450	9332	9830	6100	10361
			Average = 8810.6, Standard Deviation = 4766.821									

Use Case	Approach	Parameters	Overall dynamic time warping similarity for each entity									
			E0	E1	E2	E3	E4	E5	E6	E7	E8	E9
Medicine	fusion SAX abstraction	n = 5 windowSize = 100	53794	14772	87985	59539	11636	56912	37657	48055	59416	58855
			Average = 48862.1, Standard Deviation = 21492.064									
		n = 8 windowSize = 100	55783	15717	79014	64279	11233	57405	40690	49766	55262	57226
			Average = 48637.5, Standard Deviation = 19900.387									
	fusion IoTSAX abstraction	thr = 1 windowSize = 100	42433	**10393**	**43449**	**51357**	**7261**	**23296**	**27767**	**34773**	**40598**	**37343**
			Average = 31867.0, Standard Deviation = 13773.761									
		thr = 4 windowSize = 100	**42318**	11143	48300	54597	8162	25870	28897	37075	41118	37675
			Average = 33515.5, Standard Deviation = 14322.631									
Smart Cities	fusion SAX abstraction	n = 5 windowSize = 50	44063	50537	55845	47343	69124	137279	64767	72531	51053	78102
			Average = 67064.4, Standard Deviation = 25816.11									
		n = 8 windowSize = 50	30776	36518	43717	32159	56274	117201	47248	55230	33416	68362
			Average = 52090.1, Standard Deviation = 24633.056									
	fusion IoTSAX abstraction	thr = 0.5 windowSize = 50	**14514**	**16460**	**27582**	**13186**	**29181**	**61673**	**26368**	**31497**	**17946**	**35314**
			Average = 27372.1, Standard Deviation = 13547.938									
		thr = 1 windowSize = 50	15079	17019	27634	13407	29377	62163	26961	32094	18314	35689
			Average = 27773.7, Standard Deviation = 13564.286									

TABLE VIII
AVERAGE F-SCORE OF EACH PROPERTY AT FINDING THE CORRECT TOP-1 CONCEPTUALLY SIMILAR PROPERTY EXISTING IN THE REASONING RULES

Approach	**Memory**	**Average F-score of all properties for each use case**	
		Medicine	Smart Cities
WORD2VEC GOOGLENEWS	3.35 GB	0.697	0.740
WORD2VEC VECTORS	54.39 MB	0.677	0.789
WORD2VEC PHRASE	519.81 MB	0.606	0.722
WORD2VEC WIKIPEDIA NO LEMMATIZATION	346.60 MB	0.688	0.723
WORD2VEC WIKIPEDIA LEMMATIZATION	339.47 MB	0.656	0.723
FASTTEXT WIKIPEDIA NO LEMMATIZATION	346.54 MB	**0.750**	0.733
FASTTEXT WIKIPEDIA LEMMATIZATION	313.49 MB	0.667	0.751
MSSG WIKIPEDIA GLOBAL	363 MB	0.548	0.710
MSSG WIKIPEDIA FIRST SENSE	363 MB	0.742	0.771
DECONF FIRST SENSE	**236.91 MB**	0.727	**0.850**
CONCEPTNET	1.09 GB (English version)	**0.879**	**0.892**

sometimes may not have a similar pattern for the different approaches and use cases, which depicts why both measures should be used as each have their strengths and weaknesses. In our case, we emphasize slightly more on the ED as it depicts a straight line difference between the raw and the approximate time series values, whereas DTW observes more the general similarity between the two time series. We see that the error results follow similar observations to the compression space-saving percentage ones. Specifically, in SAX, smaller segment numbers (n) result in bigger distances for ED and the most part for DTW as higher compression takes place; therefore, more error occurs. In IoTSAX, lower thresholds result in lower approximation error as stricter dynamic segments are created, hence less compression. Generally, both measures agree that IoTSAX outperforms SAX by achieving less approximation error of up to 2 to 3 times. The difference on how much the approximation error is reduced in the case of IoTSAX according to the threshold values is more evident with data fluctuations and higher window sizes as it can be seen in medicine's use case.

2) Approximate Reasoning: Different studies have shown that the performance of the embedding models is dependent on the different parameters and the corpus that they have been trained on. Therefore, eleven embedding models are used to check their performance in finding the most conceptually relevant property existing in the rules. These models range from the ones assigning a single vector to each word (e.g., Word2Vec [18], FastText [19])

to the ones giving a vector for each sense/individual meaning of a word (e.g., MSSG [20], DeConf [21], and ConceptNet [22]). Specifically, the pretrained models are Word2Vec GoogleNews, Vectors and Phrase,[13] Word2Vec Wikipedia No Lemmatization/Lemmatization and FastText Wikipedia No Lemmatization/Lemmatization[14] [47], MSSG Wikipedia,[15] DeConf,[16], and ConceptNet.[17] For MSSG (best model is vectors.NP-MSSG.50D.30K) and DeConf only the first sense for each word is taken into account (MSSG Wikipedia First Sense and DeConf First Sense, respectively). MSSG also is examined for each word's global vector regarding all its senses (MSSG Wikipedia Global).

The correctness evaluation is done offline and the F-score results are illustrated in Table VIII. We observe that the F-scores are higher in the case of smart cities compared to the stricter terminologies in the case of medicine. Nevertheless, most models behave well with FastText Wikipedia No Lemmatization and ConceptNet being the best in medicine's use case, and DeConf and ConceptNet being the best in the use case of smart cities. An unexpected result is the third-best F-scores provided by Word2Vec Vectors for smart cities, given the fact that it is the simplest and smallest in memory model among all. It is also worth mentioning that lemmatization models behave worse or similar than no lemmatization ones and this could be attributed to possible Part-of-Speech tagging errors. The worst results for both cases are given by MSSG Wikipedia Global, making clearer the message that using senses can provide better results. Overall, we observe the superiority of the ConceptNet and DeConf models that suggest that when concept hierarchy is used to learn sense vectors for multisense words, then, the words can be better represented depending on the domain and use case. We should also observe that these models behave very well given the restrictive top-1 similarity between the sense vectors and the targeted properties existing in the reasoning rules. Nevertheless, we should be mindful of the memory regarding each model with all models apart from Word2Vec GoogleNews (3.35 GB) and ConceptNet (English-version 1.09 GB) ranging between the small values of 54.39 to 519.81 MB, making DeConf the best model regarding the tradeoff between memory and correctness.

VIII. Conclusion and Future Work

In this article, we observe that the demands in IoT applications have contributed to the need for data integration, semantic interoperability, high usability and system performance. In order to tackle these, we proposed the abstractive publish/subscribe summarization system that provides abstractive summaries of numerical IoT data for user-friendly subscriptions with the use of IoTSAX approach; an enhanced SAX approximation for dynamic IoT environments, and approximate rule-based reasoning that is based on the use of embedding models. We analyzed two use cases: 1) medicine and 2) smart cities. According to our analysis, the typical PSS approach (no fusion no abstraction) has the best latency, but it creates an abundance of notifications that could overwhelm the system and the subscriber. On the other hand, the abstractive summaries may be 2 to 3 times slower compared to the typical approach, but they can achieve up to 98% reduction in the number of notifications sent to the subscriber. In terms of abstractive techniques, IoTSAX outperforms SAX in approximation error up to 2 to 3 times more, while at the same time it has a similar or better latency and throughput performance. Also, IoTSAX may outperform SAX in compression space saving as it can detect redundant sensor readings and create a much smaller symbolic representation. IoTSAX also abstracts the user from defining the best parameters (unlike SAX) by only expecting to define the level of strictness of the compression depending on an appropriate threshold. In terms of approximate rule-based reasoning, we compared eleven embedding models and we concluded that the models that use concept hierarchy to learn sense vectors for multisense words can better represent the conceptual information of words according to the domain and use case and can achieve F-scores of up to 0.87. This results in abstracting the systems from using common formats and terminologies to achieve semantic interoperability as well as releasing them from the limitations of ontologies, taxonomies and domain experts.

Future work will focus on finding similar patterns and correlations among different time series generated by publishers and creating conceptually similar groupings of measurements. Our goal will be to create summaries that boost the system performance in PSS without sacrificing the effectiveness that could be achieved with the original raw information.

[13]https://code.google.com/archive/p/word2vec/

[14]http://vectors.nlpl.eu/repository/

[15]http://iesl.cs.umass.edu/downloads/vectors/release.tar.gz

[16]https://pilehvar.github.io/deconf/

[17]https://github.com/commonsense/conceptnet-numberbatch

Acknowledgment

For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

References

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013.

[2] R. Dautov, S. Distefano, and R. Buyya, "Hierarchical data fusion for smart healthcare," *J. Big Data*, vol. 6, no. 1, p. 19, 2019.

[3] S. D. Nagowah, H. B. Sta, and B. Gobin-Rahimbux, "An overview of semantic interoperability ontologies and frameworks for IoT," in *Proc. IEEE 6th Int. Conf. Enterprise Syst. (ES)*, 2018, pp. 82–89.

[4] Ş. Kolozali *et al.*, "Observing the pulse of a city: A smart city framework for real-time discovery, federation, and aggregation of data streams," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2651–2668, Apr. 2019.

[5] A. Tsymbal, *The Problem of Concept Drift: Definitions and Related Work*, Comput. Sci. Dept., Trinity College Dublin, Dublin, Ireland, 2004.

[6] Y. Qin, Q. Z. Sheng, N. J. Falkner, S. Dustdar, H. Wang, and A. V. Vasilakos, "When things matter: A survey on data-centric Internet of Things," *J. Netw. Comput. Appl.*, vol. 64, pp. 137–153, Apr. 2016.

[7] A. Gyrard, C. Bonnet, K. Boudaoud, and M. Serrano, "LOV4IoT: A second life for ontology-based domain knowledge to build semantic Web of Things applications," in *Proc. IEEE 4th Int. Conf. Future Internet Things Cloud (FiCloud)*, 2016, pp. 254–261.

[8] F. Ganz, P. Barnaghi, and F. Carrez, "Automated semantic knowledge acquisition from sensor data," *IEEE Syst. J.*, vol. 10, no. 3, pp. 1214–1225, Sep. 2016.

[9] P. Barnaghi, W. Wang, C. Henson, and K. Taylor, "Semantics for the Internet of Things: Early progress and back to the future," *Int. J. Semantic Web Inf. Syst.*, vol. 8, no. 1, pp. 1–21, 2012.

[10] S. Pattar, R. Buyya, K. R. Venugopal, S. Iyengar, and L. Patnaik, "Searching for the IoT resources: Fundamentals, requirements, comprehensive review, and future directions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 2101–2132, 3rd Quart., 2018.
[11] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the Internet of Things: A survey," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 414–454, 1st Quart., 2013.
[12] E. Ahmed and M. H. Rehmani, "Mobile edge computing: Opportunities, solutions, and challenges," *Future Gener. Comput. Sci.*, vol. 70, pp. 59–63, May 2017.
[13] C. C. Aggarwal, *Data Streams: Models and Algorithms*, vol. 31. New York, NY, USA: Springer, 2007.
[14] F. Ganz, D. Puschmann, P. Barnaghi, and F. Carrez, "A practical evaluation of information processing and abstraction techniques for the Internet of Things," *IEEE Internet Things J.*, vol. 2, no. 4, pp. 340–354, Aug. 2015.
[15] S. Hasan and E. Curry, "Approximate semantic matching of events for the Internet of Things," *ACM Trans. Internet Technol.*, vol. 14, no. 1, pp. 1–23, 2014.
[16] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Comput. Surveys*, vol. 35, no. 2, pp. 114–131, 2003.
[17] M. Sydow, M. Pikuła, and R. Schenkel, "DiverSum: Towards diversified summarisation of entities in knowledge graphs," in *Proc. IEEE 26th Int. Conf. Data Eng. Workshops (ICDEW)*, 2010, pp. 221–226.
[18] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 26, 2013, pp. 3111–3119.
[19] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," 2016. [Online]. Available: arXiv:1607.01759.
[20] A. Neelakantan, J. Shankar, A. Passos, and A. McCallum, "Efficient non-parametric estimation of multiple embeddings per word in vector space," 2015. [Online]. Available: arXiv:1504.06654.
[21] M. T. Pilehvar and N. Collier, "De-conflated semantic representations," 2016. [Online]. Available: arXiv:1608.01961.
[22] R. Speer, J. Chin, and C. Havasi, "ConceptNet 5.5: An open multilingual graph of general knowledge," 2016. [Online]. Available: arXiv:1612.03975.
[23] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast subsequence matching in time-series databases," *ACM SIGMOD Rec.*, vol. 23, no. 2, pp. 419–429, 1994.
[24] K.-P. Chan and A. W.-C. Fu, "Efficient time series matching by wavelets," in *Proc. IEEE 15th Int. Conf. Data Eng.*, 1999, pp. 126–133.
[25] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometr. Intell. Lab. Syst.*, vol. 2, nos. 1–3, pp. 37–52, 1987.
[26] M. Moonen and B. De Moor, *SVD and Signal Processing, III: Algorithms, Architectures and Applications*. Amsterdam, The Netherlands: Elsevier, 1995.
[27] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality reduction for fast similarity search in large time series databases," *Knowl. Inf. Syst.*, vol. 3, no. 3, pp. 263–286, 2001.
[28] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *Proc. 8th ACM SIGMOD Workshop Res. Issues Data Min. Knowl. Disc.*, 2003, pp. 2–11.
[29] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh, "Querying and mining of time series data: Experimental comparison of representations and distance measures," *Proc. VLDB Endow.*, vol. 1, no. 2, pp. 1542–1552, 2008.
[30] Ş. Kolozali, D. Puschmann, M. Bermudez-Edo, and P. Barnaghi, "On the effect of adaptive and nonadaptive analysis of time-series sensory data," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1084–1098, Dec. 2016.
[31] J. Borg and J. G. Vella, "Mining massive time series data: With dimensionality reduction techniques," in *Proc. Int. Conf. Adv. Comput. Data Sci.*, 2020, pp. 496–506.
[32] D. Puschmann, P. Barnaghi, and R. Tafazolli, "Using LDA to uncover the underlying structures and relations in smart city data streams," *IEEE Syst. J.*, vol. 12, no. 2, pp. 1755–1766, Jun. 2018.
[33] F. Ganz, P. Barnaghi, and F. Carrez, "Information abstraction for heterogeneous real world Internet data," *IEEE Sensors J.*, vol. 13, no. 10, pp. 3793–3805, Oct. 2013.
[34] H. Park and J.-Y. Jung, "SAX-ARM: Deviant event pattern discovery from multivariate time series using symbolic aggregate approximation and association rule mining," *Exp. Syst. Appl.*, vol. 141, Mar. 2020, Art. no. 112950.
[35] C. T. Zan and H. Yamana, "Dynamic sax parameter estimation for time series," *Int. J. Web Inf. Syst.*, vol. 13, no. 4, pp. 387–404, 2017.
[36] M. Serrano and A. Gyrard, "A review of tools for IoT semantics and data streaming analytics," in *Building Blocks for IoT Analytics*, vol. 6, J. Soldatos, Ed. Athens, Greece: River Publ., pp. 139–163, Nov. 2016.
[37] C. El Kaed, I. Khan, A. Van Den Berg, H. Hossayni, and C. Saint-Marcel, "SRE: Semantic rules engine for the industrial Internet-of-Things gateways," *IEEE Trans. Ind. Informat.*, vol. 14, no. 2, pp. 715–724, Feb. 2018.
[38] E. Jajaga and L. Ahmedi, "C-SWRL: SWRL for reasoning over stream data," in *Proc. IEEE 11th Int. Conf. Semantic Comput. (ICSC)*, 2017, pp. 395–400.
[39] W. Wei and P. Barnaghi, "Semantic annotation and reasoning for sensor data," in *Proc. Eur. Conf. Smart Sens. Context*, 2009, pp. 66–76.
[40] M. Petrovic, I. Burcea, and H.-A. Jacobsen, "S-TOPSS: Semantic toronto publish/subscribe system," in *Proc. VLDB Conf.*, 2003, pp. 1101–1104.
[41] H. Liu and H.-A. Jacobsen, "A-TOPSS—A publish/subscribe system supporting approximate matching," in *Proc. 28th Int. Conf. Very Large Databases (VLDB)*, 2002, pp. 1107–1110.
[42] N. Alhakbani, M. M. Hassan, M. Ykhlef, and G. Fortino, "An efficient event matching system for semantic smart data in the Internet of Things (IoT) environment," *Future Gener. Comput. Syst.*, vol. 95, pp. 163–174, Jun. 2019.
[43] M. Petrovic, H. Liu, and H.-A. Jacobsen, "G-TOPSS: Fast filtering of graph-based metadata," in *Proc. 14th Int. Conf. World Wide Web*, 2005, pp. 539–547.
[44] C. Esposito, M. Ficco, F. Palmieri, and A. Castiglione, "A knowledge-based platform for big data analytics based on publish/subscribe services and stream processing," *Knowl. Based Syst.*, vol. 79, pp. 3–17, May 2015.
[45] R. A. Davis, K.-S. Lii, and D. N. Politis, "Remarks on some nonparametric estimates of a density function," in *Selected Works of Murray Rosenblatt*. New York, NY, USA: Springer, 2011, pp. 95–100.
[46] S. V. Vaseghi, *Advanced Digital Signal Processing and Noise Reduction*. New York, NY, USA: Wiley, 2008.
[47] A. Kutuzov, M. Fares, S. Oepen, and E. Velldal, "Word vectors, reuse, and replicability: Towards a community repository of large-text resources," in *Proc. 58th Conf. Simulat. Model.*, 2017, pp. 271–276.

Niki Pavlopoulou received the B.Sc. degree in computer engineering and informatics from the University of Patras, Patras, Greece, in 2013, and the M.Sc. degree in bioinformatics and systems biology from the University of Manchester, Manchester, U.K., in 2014. She is currently pursuing the Ph.D. degree with the Insight Centre for Data Analytics, National University of Ireland Galway, Galway, Ireland.

She was a KTP Associate with Big Data Analytics for Exonar Ltd., Reading, U.K., and the University of Reading, Reading, in 2015. She is currently exploring summarization and approximation methodologies of data streams with a special interest in applications for smart cities and health informatics. Her research interests include semantic technologies, data mining, machine learning, and the Internet of Things.

Edward Curry (Member, IEEE) received the Ph.D. degree in computer science from the National University of Ireland Galway, Galway, Ireland, in 2016.

He is a Principle Investigator with the Insight Centre for Data Analytics, National University of Ireland Galway, Galway, Ireland. He has authored or coauthored over 200 scientific articles in journals, books, and international conferences. His research work is currently focused on engineering adaptive systems that are a foundation of smart and ubiquitous computing environments. His research interests are predominantly in open distributed systems, particularly in the areas of incremental data management, approximation, and unstructured event processing, with a special interest in applications for smart environments and data ecosystems.

Dr. Curry is a Vice President of the Big Data Value Association-a nonprofit industry-led organization with the objective of increasing the competitiveness of European Companies with data-driven innovation. He has presented at numerous events and has given invited talks at Berkeley, Harvard, MIT, and Stanford.