PoSSUM: An Entity-centric Publish/Subscribe System for Diverse Summarization in Internet of Things

NIKI PAVLOPOULOU and EDWARD CURRY, Insight Centre for Data Analytics, National University of Ireland Galway, Ireland

Users are interested in entity information provided by multiple sensors in the Internet of Things. The challenges regarding this environment span from data-centric ones due to data integration, heterogeneity, and enrichment, to user-centric ones due to the need for high-level data interpretation and usability for non-expert users, to system-centric ones due to resource constraints. Publish/Subscribe systems (PSSs) are suitable schemes for large-scale applications, but they are limited in dealing with the data and user challenges. In this article, we propose PoSSUM, a novel entity-centric PSS that provides entity summaries for user-friendly subscriptions through data integration, a novel Density-Based VARIance Clustering (DBVARC) for diverse entity summarization that is parameter-free and partly incremental, reasoning rules, and a novel Triple2Rank scoring for top-k filtering based on importance, informativeness, and diversity. We introduce a novel evaluation methodology that creates ground truths and metrics that capture the quality of entity summaries. We compare our approach with a previous dynamic approach and a static diverse entity summarization approach that we adapted to dynamic environments. The evaluation results for two use cases, Healthcare and Smart Cities, show that when users are provided with less information, their data diversity desire could reach up to 80%. Summarization approaches achieve from 80% to 99% message reduction, with PoSSUM having the best-ranking quality for more than half of the entities by a significant margin. PoSSUM has the highest conceptual clustering F-score, ranging from 0.69 to 0.83, and a redundancy-aware F-score up to 0.95, with cases, where it is almost two times better than the other approaches. PoSSUM takes 50% or less clustering processing time and it performs scoring significantly faster for larger windows. It also has comparable end-to-end latency and throughput values, and it occupies a third of the memory compared to the second-best approach.

CCS Concepts: • **Software and its engineering** → **Publish-subscribe/event-based architectures**; • **Information systems** → **Summarization**; *Data stream mining*; • **Theory of computation** → **Unsupervised learning and clustering**; *Semantics and reasoning*;

Additional Key Words and Phrases: Diverse entity summarization, triple ranking, Top-k filtering, Internet of Things

ACM Reference format:

Niki Pavlopoulou and Edward Curry. 2022. PoSSUM: An Entity-centric Publish/Subscribe System for Diverse Summarization in Internet of Things. *ACM Trans. Internet Technol.* 22, 3, Article 73 (March 2022), 30 pages. <https://doi.org/10.1145/3507911>

This publication has emanated from a research grant from the European Union's Horizon 2020 research programme Big Data Value ecosystem (BDVe) grant No 732630 and in part from Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289_P2, co-funded by the European Regional Development Fund.

Authors' address: N. Pavlopoulou and E. Curry, Insight Centre for Data Analytics, National University of Ireland Galway, Galway, Ireland; emails: {niki.pavlopoulou, edward.curry}@insight-centre.org.

This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2022 Copyright held by the owner/author(s).

1533-5399/2022/03-ART73

<https://doi.org/10.1145/3507911>

1 INTRODUCTION

The **Internet of Things (IoT)** consists of several physical devices connected to the Internet [47]. It is used for many applications ranging from Smart Cities to Healthcare [12]; therefore, analysis and interpretation of the data deriving from the devices are of the utmost importance for users or applications to gain valuable insight. However, the interest drawn in the past years to the IoT has created many challenges that have not yet been efficiently and effectively tackled.

IoT devices cover a wide range of data types that concern multiple entities (e.g., buildings, patients, etc.) [47]. Often this raw data cannot stand on its own and needs to be integrated with other raw data [24] or enriched by external databases to provide richer and complementary contextual information. Therefore, multiple heterogeneous data is generated from various sensors that relate to the characteristics of one or more entities. This poses a complexity in data analysis not only for integrating the data that concern each entity but also for creating semantic abstractions that translate to high-level interpretations that could involve an entity or its relation to others [5].

Humans perform queries to reach a contextual interpretation of the IoT data. For example, Publish/Subscribe is an Event-based System, where publishers generate sensor data and humans create subscriptions (like queries) to apply filters to the data based on their interests [16]. Nevertheless, the interpretation and specific query creation derive from humans' subjective nature and may be affected by personal experiences or biases [49]. This may lead to false knowledge inferences or even missing information. Additionally, the high heterogeneity in IoT data, the number of sources and entities, and the high streaming rate of the sensors will create an overloading amount of information for a human to comprehend [23]. According to George Miller, people can only receive, process, and remember a limited amount of information at a time; otherwise, they get overloaded [31]. Therefore, humans are responsible for creating complex technical queries to find narrower and more specific information, even though they might not have the capacity to do so.

IoT architectures are characterized by dynamic streams, resource constraints, and real-time requirements [43]. Nevertheless, the high amount of data generated by multiple sources makes any possible data analysis cumbersome in terms of memory, power, and network interfaces [32]. Hardware and communication can ease these challenges, but the way the data analysis methodologies are designed can prove highly efficient. Therefore, the IoT analysis needs to be quick, versatile, resource efficient, and able to integrate information coming from multiple dynamic devices.

In this article, we propose PoSSUM, a novel entity-centric **Publish/Subscribe System (PSS)** that provides entity summaries for user-friendly diversity-aware queries through data integration, diverse entity summarization, reasoning rules, and top-k filtering that could resolve the aforementioned IoT challenges.

The contributions of this articles are the following:

- PoSSUM, an entity-centric PSS for diverse entity summarization in IoT that contains methodologies for entity graphs with object-based and data-type (numerical) properties
- A user-friendly entity-centric subscription that allows users to receive top-k diverse filtered information using a specific window type with the desired window size
- A novel **Density-Based VARIance Clustering (DBVARC)** that is parameter-free, is partly incremental, and can be used for general streaming applications
- A novel data ranking approach (Triple2Rank) that is based on importance, informativeness, and diversity
- A synthetic dataset based on real-world data and ground-truth sets deriving from a novel evaluation methodology for entity summarization.
- An extensive evaluation using real-world IoT and DBpedia data among our approach and existing thesauri/ontology-based and embedding-based approaches by examining their correctness and system performance

Fig. 1. Healthcare use case: A doctor is interested in a patient's summarized health status. Sensors generate timestamped information records ranging from the patient's health sensors to external static sources deriving from the patient's home or the hospital. An example of a top 5 summary is presented.

The rest of this article is structured as follows. In Section 2 the problem is analyzed, Section 3 presents the necessary background, Section 4 contains related work, Section 5 describes PoSSUM on a high level, Section 6 presents the main approach, evaluation is given in Section 7, and conclusions are made in Section 8.

2 PROBLEM ANALYSIS

In this section, the problem is analyzed with motivational scenarios and their challenges. Also, the research questions and requirements are provided.

2.1 Motivational Scenario

As a motivational scenario, two use cases are explored: Healthcare and Smart Cities.

2.1.1 Healthcare Use Case. A doctor observes multiple patients daily, who are either in the hospital or at home. The patients' medical information ranges from body sensor reading (e.g., heart rate) medical history to living conditions. The body sensors could belong to different manufacturers deployed in different hospital rooms or purchased by the patients themselves at home. Due to the high patient number and limited time, the doctor is not interested in explicit patient readings and does not want to manually fuse the available medical data and be overwhelmed by viewing its high volume. The doctor is interested in a summarized view of the patient's health status to know if intervention is needed through hospital visits or appointment bookings or to detect any possible epidemic outbreaks with patients in different hospital rooms. The use case is illustrated in Figure 1.

2.1.2 Smart Cities Use Case. A real estate agent helps customers to buy, sell, or rent commercial or residential smart properties. A smart property will generate data ranging from sensor readings (e.g., temperature) to nearby social events. These properties will be located in different areas within a building or its surroundings; therefore, they will have sensors from different manufacturers and different or similar events nearby. The real estate agent is interested in a property's summarized information like its energy consumptions or the popularity of the nearby social events so that one

Fig. 2. Smart Cities use case: A real estate agent is interested in a property's summarized information. Sensors generate timestamped information records ranging from spatially nearby kitchen sensors to external static sources deriving from the property or the nearby venues. An example of a top 5 summary is presented.

is aware of the property value to potential buyers or tenants and to compare the value of different properties in different regions. The use case is illustrated in Figure 2.

2.2 Challenges

2.2.1 Data Challenges. These challenges involve heterogeneity, redundancy, and data enrichment.

Heterogeneity (C1): The IoT data contains various patient/property information in different schemata and semantics depending on their manufacturers. The subscribers would expect a notification that covers as much unique diverse patient/property information as possible in a common representational format.

Redundancy (C2): Redundancy is caused by (1) duplication due to frequent sampling rates that generate identical data from sensors that have unchanged states for time periods (e.g., Figure 2: humidity = 39.6878%), (2) conceptual similarity due to heterogeneous data deriving from multiple sensors regarding the same thing or sensors located nearby (e.g., Figure 1: "heart rate" vs. "pulse," different types of blood pressure including the mean value, "birthPlace" vs. "placeOfBirth" and Figure 2: "temperature" vs. "inside air temperature," "humidity" vs. "atmospheric humidity," country vs. part of region or municipality). The redundancy will result in voluminous unnecessary data that will overwhelm the subscribers and impose resource limitations on the processing system.

Data Enrichment (C3): The raw sensor health/property data by single sources (e.g., Figure 1: the patient's heart rate, pulse, blood pressure, SpO2, and Figure 2: room's temperature, humidity) will not lead to any satisfactory notifications for the subscribers. An automatic combination of raw data from multiple sources and enrichment of data from external static sources (e.g., Figure 1: living conditions like placeOfBirth, medical history like previous hospital visits, and Figure 2: cultural events nearby, location information) will lead to a notification with complementary contextual information [34] concerning a patient/property.

2.2.2 User Challenges. These challenges involve high-level interpretation and non-technical users.

High-level Interpretation (C4): The subscribers are not interested in numerical sensor data of specific values (e.g., Figure 1: "heart rate = 100.8bpm" and Figure 2: temperature = 19.567°C)

as they do not depict a meaningful message. The subscribers are interested in a summarized view of multiple patient/property data including numerical data or automatic data interpretations (e.g., Figure 1: a patient has tachycardia and in Figure 2: warm temperature) that will help them infer their own knowledge and high-level interpretations (e.g., patient is in critical condition or property is of high quality).

Non-technical Users (C5): The doctor and the real estate agent are not technology experts; therefore, they find complex queries like SPARQL to be unfriendly [34]. Also, they are not aware a priori of the available IoT sources [45] to manually select which ones are appropriate [34] nor of the semantics or schemata used in the data coming from sensors of different manufacturers to make more explicit queries (e.g., Figure 1: “peripheral capillary oxygen saturation” instead of “SpO2” and Figure 2: “kitchen” instead of “kitchen entrance,” “Fahrenheit” instead of “degrees Celsius”). The doctor and the real estate agent prefer a query that is neither too simple as to provide partial patient/property information nor too abstract to overload them or the system. They need a contextually aware query that covers as much complete information provided by the sources as possible in a user-friendly data representation (e.g., the doctor wants to obtain health status information based only on a patient’s name, which is presented in an understandable structure).

2.2.3 Performance Challenges. These challenges involve timeliness and resource constraints.

Timeliness and Resource Constraints (C6): The high number of sensors and patients/properties as well as the data heterogeneity involved in IoT create voluminous data that causes network overhead. Also, the sources are dynamic and create unbounded data. Nevertheless, the data needs to be processed quickly, in real time, and with as low memory consumption as possible to satisfy the subscribers’ needs that may include the most recent information.

2.3 Research Questions and Requirements

Based on the motivational scenario and challenges above, the **research questions (RQs)** and **requirements (Rs)** of this work are the following:

- **RQ1:** Can we create a PSS that offers usability (**R1**) while maintaining a user’s expressibility (**R2**) effectively (**R4**) and efficiently (**R5**)? (**C1–C6**)
- **RQ2:** Can we create a PSS that offers expressiveness of heterogeneous data (**R3**) effectively (**R4**) and efficiently (**R5**)? (**C1–C6**)

3 BACKGROUND

3.1 Event-based Systems

In Event-based Systems, users or applications use conditional rules to express their data requests (events), and sources generate data that is analyzed and notify the users or applications when these events have been observed. Therefore, these systems abstract users from the underlying analysis of the data and are efficient due to their scalable and distributed nature. These advantages make Event-based Systems excellent candidates for the challenging world of IoT [22].

An Event-based System is Publish/Subscribe [16]. In IoT, sensor data derives from publishers, and queries come from subscriptions (conditional rules) defined by users (subscribers). All sensor data and subscriptions are processed by the engine, which is responsible for matching the rules of subscriptions to the publications that satisfy them. Once a match is observed, a notification of the result (event) would be sent to the subscriber. PSSs are a suitable interaction scheme for IoT dynamic large-scale applications since they are decoupled in three ways [16]: (1) space decoupling since publishers and subscribers do not need to know each other, (2) time decoupling since publishers and subscribers do not need to be active at the same time, and (3) synchronization decoupling

since publishers are not blocked during event production, and subscribers can be notified while performing another activity.

There are three main schemes in PSSs: Topic-based, Content-based, and Graph-based PSSs.

3.1.1 Topic-based PSSs. Publishers publish events on specific topics expressed as keywords (e.g., Sports), and subscribers that have subscribed to these topics get notified whenever there is a match. Topics are good candidates for non-complex entity queries or concept-based queries that could cover different data interpretations by humans. Nevertheless, publishers and subscribers need to have a shared understanding of the topics. Also, the topic included in the publications is assumed to have the exact schema for matching to occur. This is a rigid way of matching without considering a more relaxed, approximate entity-based matching that integrates data from multiple sources. Furthermore, an abstract topic query like this of an entity along with its sub-topics (topic hierarchy) may lead to an abundance of information that would overwhelm both users and the system.

3.1.2 Content-based PSSs. It improves the expressibility of users compared to the topic-based one by adding event content filtering on the subscription side. This filtering typically involves comparison operators ($=$, $<$, $\leq$, $>$, $\geq$) on attribute-value pairs derived from the events. Complex subscription rules can also be created by logical combinations (and, or, etc.) of individual constraints. For example, an event could be (gender = female, age = 20) and a subscription that matches it could be (gender = female, age < 30). Although these systems are used in the IoT, they assume that users are aware of the schemata and semantics of the heterogeneous data involved to make specific conditional queries without also considering the users' expertise in query languages. Additionally, users might form strict queries based on their intuition on an entity's concept that might lead to partial entity information within a notification. This makes a high-level interpretation of results more difficult for users. Furthermore, data integration would pose a problem as it would require users to perform complex join queries and be aware of all the sources involved in an entity's data generation [34]. Finally, conditional queries are more specific than a topic query, so they might not be so memory heavy since they result in filtered notifications; nevertheless, a strict query might lead to unnecessary duplicate results while an approximate, relaxed query [4, 22] might lead to conceptually similar results that could be deemed redundant by the user.

3.1.3 Graph-based PSSs. It could be considered a sub-category of Content-based PSSs, but for graph-based publications instead of attribute-value pairs [8]. In this scheme, points of interest are the graph's nodes and the relations between the nodes that are the graph's edges. Subscriptions can be SPARQL-like queries of specific nodes and relations among them. The notifications are those graphs that match the subscriptions. Graphs are richer and more dynamic structures of representing IoT data; therefore, graph-based PSSs would be good candidates in the IoT. Nevertheless, they have not been explored much and they inherit all of the limitations of content-based PSSs. Also, to the best of our knowledge, there is no possibility of an approximate, relaxed query in graph-based PSSs at the moment [43]. This contradicts the ability to represent IoT data in rich graph structures compared to narrow, rigid attribute-value ones but not being able to also perform flexible matching.

We propose PoSSUM, an Entity-based PSS that combines the advantages of Topic-based and Graph-based PSSs. In this scheme, subscribers can perform non-complex diversity-aware queries concerning an entity, therefore abstracting them from the need of using exact schemata and semantics. The graph-based notifications received contain approximate non-redundant entity information integrated from multiple sources and cover the subscribers' different data interpretations. Also, the notifications refrain from overwhelming the subscribers as the latter can define the desired notification size within the queries.

3.2 Knowledge Graphs and Current Summarization Approaches

IoT data information comes from multiple sensors concerning different entities; therefore, representing the raw sensor data to conceptual entities with their associated properties or background information will result in richer data analysis and querying [33]. Knowledge graphs could be used as a representation since their nodes represent entities, and their directed labeled arcs constitute relations among them. The **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 either an entity (object-type properties) or a number/string (data-type properties), and the property is their relation. An example of a triple is (Patient, heartRate, 100.8) from Figure 1.

IoT data is often so changeable and heterogeneous that focusing on a single entity and its individual or shared properties could be more meaningful to a user [49]. However, simple entity-based keyword queries result in abundant information generated continuously by the sensors, so it will be difficult for users to handle [23]. Entity summaries can assist as they are a subset or a high-level inference of all the available entity information. Summaries, in general, can be used for data integration purposes and can assist in the definition of a mapping between different sensor schemata and semantics [25]. They can also reduce the information sent to the user since only a representative subset is created, and hence users will not be overwhelmed. Query processing over summaries can improve the efficiency of a resource-constraint processing system when it comes to memory, network overhead, or further processing speed, assuming the error range is small [1]. The summary's size can act as a tradeoff between effectiveness and efficiency [23]. Specific types of summaries, like diverse ones, may also help with the subjective contextual interpretations of humans as they create a representative subset that includes a diverse coverage of the whole entity information. Therefore, summaries can resolve the aforementioned IoT challenges.

There are two strategies in entity summaries [49]: (1) extractive summary, where all graph nodes and their links/relations are taken by the original source graphs, and (2) abstractive summary, where either new graph nodes and their links/relations are created or a high-level interpretation message (as in this article) is provided. Diverse entity summarization covers a wide range of the available entity information, and repetitions are avoided and unique information pieces on an entity are preferred. We propose an approach that involves only diverse entity summarization in PSSs with a combination of extractive and abstractive strategies. Current diverse entity summarization approaches could be split into two categories: Thesauri/Ontology based and Embedding based.

3.2.1 Thesauri/Ontology-based Diverse Summarization Approach. This category involves methodologies that rely on ontologies or a combination of thesauri and ontologies to create diverse entity summaries. Ontologies and thesauri are domain dependent and create knowledge representations about concepts, their properties, and relations with other concepts [22]. Their domain and application dependency make them effective for processing data sources of one kind [42]. Therefore, these limitations, along with the specific schemata and semantics used, make them ineffective for the heterogeneous nature of IoT data that comes from multiple sources [4] and need to work on a global scale [5]. Also, although using these approaches as annotations for further processing could work for big static data, they could prove inefficient in the dynamic nature of IoT streams [33]. The lack of an ontology or experts to define one for specific domain-related data generated by IoT sensors could also prove fatal when using an ontology-dependent approach as there is no sufficient information. Also, the presence or expansion of an ontology or the occurrence of errors within it might create additional problems in the data, like more complexity in terms of the number of concepts and heterogeneity [11] that already exist in IoT data.

The most representative work in this category is FACES [18], which uses the WordNet thesaurus (<https://wordnet.princeton.edu/>) for data enrichment, a modified Cobweb [17] hierarchical conceptual clustering algorithm, and a ranking based on the DBpedia ontology (<https://wiki.dbpedia.org/>). The aim is to create entity summaries of static data based on diversity, uniqueness, and popularity. Works belonging to this category could be adapted for a dynamic environment like IoT, depending on their restrictions, to examine their performance as we did in a previous work [36] for FACES by applying it to dynamic Linked Data.

3.2.2 Embedding-based Diverse Summarization Approach. This category involves methodologies that rely on word embeddings to create diverse entity summaries. Word embeddings are models trained on text-corpora and represent words into vectors. In this way, a word can be visualized in a semantic vector space, where words that are semantically synonymous, related, or even antonymous will be closer together. Word embeddings are more flexible models compared to the strict ontologies or thesauri in terms of domains, concepts, schemata, and semantics as they are based on context. For example, semantically opposite words (antonyms) but conceptually similar (e.g., death place - birthplace) are represented closely in the vector space, whereas in an ontology or a thesaurus they would not be so closely linked, if linked at all. Also, word embeddings can find relations among phrases, whereas phrases might be completely absent in the case of ontologies or thesauri. Another advantage of word embeddings is that they do not rely on experts to construct them as they are created automatically based on text-corpora that cover a range of topics. These advantages make word embeddings better candidates for efficiently analyzing heterogeneous IoT data that comes from multiple sources. On the other hand, word embeddings are limited in terms of the text-corpora they have been trained on; therefore, some words or phrases might not exist.

The most representative work in this category is a previous work of ours [35], which for the rest of the article will be referred to as PubSum. This prior work uses a combination of Word2Vec [30] embedding models, the DBSCAN [15] algorithm for conceptual clustering, and a ranking based on cosine and Euclidean similarity metrics. The purpose is to create entity summaries on dynamic and heterogeneous data with the use of windows based on relevance, diversity, and importance. In this article, we propose a new approach, PoSSUM, and we compare it to our previous one.

4 RELATED WORK

4.1 Summarization

4.1.1 Summarization in PSSs. Summarization has been examined in PSSs by several works. Some notable works are Triantafyllou et al. [50], which uses subscription subsumption on attribute-value constraints (content-based PSSs) to merge them into summary structures; Doblander et al. [13], which compresses notifications via sampling and creates dictionaries that link to a topic (topic-based PSSs) to which a subscriber subscribes; G-ToPSS [40], which subsumes subscriptions of queries for RDF-like data (graph-based-like PSSs); and Liu et al. [27], which uses subscription subsumption by storing only the most general subscriptions based on semantics or subscription merging by checking the frequency that subscriptions are matched together by the same event (graph-based-like PSSs). Our work differentiates from these works as it focuses on entity-centric publication summaries and not subscription summaries or attribute-value pair-based publication approximation. Our work is a combination of topic-based and graph-based PSSs and it does not pre-assume that the subscriber is aware of the structure and semantics of graph-like publications. To the best of our knowledge, no work has tackled summarization in graph-based PSSs, which is the focus of our work.

4.1.2 Static Diverse Entity Summarization. Entity summarization has been examined by multiples approaches [28]. Most of these approaches are relevance-based entity summaries and

focus on the structure of the graphs, unlike our approach that only refers to star-like graphs. The most notable related approaches to ours are the unsupervised diversity-based summaries like the aforementioned FACES, as well as DIVERSUM [46] and FACES-E [19]. DIVERSUM focuses on a per-property basis summarization based on novelty, importance, popularity, and diversity by adapting the document-based Information Retrieval to the knowledge graphs, whereas FACES-E improves on FACES by considering data-type properties (strings containing entity references) instead of only object-type properties for entity summarization. Another notable work is ES-LDAext [41], which uses Word2Vec embedding models for data enrichment, a modified **Latent Dirichlet Allocation (LDA)** [6] topic modeling methodology, where each entity is a multinomial distribution over its properties, and a ranking based on probability distributions in the DBpedia ontology. Recent approaches, like DeepLENS [29] and ESA [51], also use embedding models for entity summarization. However, these approaches are supervised and may need pre-defined tuned parameters; therefore, they could not be adapted in a streaming environment. We focus the evaluation of our approach only on FACES since it is an unsupervised diverse entity summarization approach that is the most relevant to our work. All approaches belonging to this category examine only static Linked Data; therefore, they could not be directly applied to streaming environments or temporal numerical IoT data.

4.2 Clustering

Clustering is an unsupervised learning methodology with several approaches [14] ranging from K-means to density-based algorithms like DBSCAN to hierarchical algorithms like Agglomerative Hierarchical Clustering. Nevertheless, all these approaches are related to static data and they could not be directly applied to streaming data as in the case of the IoT. The most related sub-categories of clustering approaches to our work are conceptual clustering and stream clustering.

4.2.1 Conceptual Clustering. This clustering involves placing instances (attribute-value pairs) into disjoint clusters that each correspond to a concept. The most influential works are the initial ones like Witt [21] and COBWEB (which FACES is based on) that create conceptual clusters based on the co-occurrence between pairs of features or how discriminative one category/concept is from others based on common features. These approaches are schematically and semantically coupled as they involve attribute-value pairs and exact semantics in values among different features, respectively. More approaches are discussed by Pérez-Suárez et al. [39], who mention that most of them are computationally expensive and memory heavy, they contain many parameters including a pre-defined number of clusters to be manually tuned for best results, some need corpora or knowledge bases to be trained on, some are not incremental, some need user input/feedback, and some cannot be applied directly to triples (FACES adapted COBWEB for triples). All of these constraints would not scale well in an IoT environment.

4.2.2 Stream Clustering. This clustering involves the on-the-fly clustering of streams. The most important approaches are CluStream [3], DenStream [9], ClusTree [26], and DBSTREAM [20]. CluStream creates micro-clusters (a tuple that contains statistics of a set of data points) in an online manner and performs K-means offline for refining the clusters (macro-clusters). DenStream extends DBSCAN to create micro-clusters in a streaming fashion and creates final clusters in an offline manner. It involves two offline stages, one where the original DBSCAN is applied to create an initial set of clusters, and another one where the final clusters are defined based on density. ClusTree creates a hierarchical tree of micro-clusters inspired by the structures of R-trees based on data distributions and Euclidean distances. Its main characteristic is that it does anytime clustering in different time intervals, and as with DenStream, it has a non-optimized initialization phase. DBSTREAM extends the micro-clustering structure by incorporating the density between the area

Table 1. Overall Comparison of Different Approaches/Systems

Approach/System	R1	R2	R3	R4	R5
Topic-based PSS	✓				✓
Content/Graph-based PSS		✓		✓ (subscription)	✓
Clustering & related work	N/A	N/A	✓ (approach/parameter)	✓ (approach/parameter)	
FACES & related work	✓	✓	✓ (ontology/thesaurus)	✓ (summary)	
PoSSUM	✓	✓	✓ (embeddings model)	✓ (summary)	✓

R1: Usability, R2: User Expressibility, R3: Data Expressiveness, R4: Effectiveness, and R5: Efficiency.

of two micro-clusters to be used in the offline re-clustering phase. Nevertheless, when it comes to IoT environments, these approaches pose many limitations. The offline initialization phase and final clustering are computationally inefficient, the approaches mostly apply to numerical data and not triples, some approaches assume the data follow a Gaussian distribution that might not always be the case, the creation of only spherical clusters by some approaches (excluding DenStream and DBSTREAM) is rather limited, and the pre-determination of the number of clusters (excluding DenStream and DBSTREAM) as well as other finely tuned parameters (e.g., threshold values in DenStream and DBSTREAM, the amount of data stored in each tree node in ClusTree, etc.) cannot be applicable for real-time data. Carnein et al. [10] discuss more approaches and their comparison.

4.3 Stream Approximation and Contextual Reasoning in IoT

Stream approximation on numerical data involves aggregation, synopsis, or frequent patterns [2], whereas contextual awareness entails supervised or unsupervised models (e.g., neural networks, clustering), manual rules, fuzzy rules, ontologies, or probabilistic models (e.g., Markov models) to transform a low-level context to a high-level one [38]. In this work, aggregation and manual rules are used for approximation and reasoning, respectively, since they are simple, time-efficient, and low-memory ways to infer abstractions of the numerical IoT data in resource-constrained environments. Advanced methodologies like our proposed IoTSAX [37], involving an enhanced SAX approximation and approximate reasoning rules, could also be applied in this work.

A comparison among the related approaches concerning this work's requirements is shown in Table 1.

5 ENTITY-CENTRIC PUBLISH/SUBSCRIBE SUMMARIZATION SYSTEM (POSSUM)

An entity-centric Publish/Subscribe summarization system (PoSSUM) needs to be defined with the solutions/techniques shown in Table 2.

5.1 Architecture

The architecture of the proposed PoSSUM is illustrated in Figure 3. Publishers generate entity-centric publications and subscribers create Diversity-aware Subscriptions concerning these entities (Subscription Model). All publications and subscriptions enter PoSSUM, which analyzes the publications and notifies the subscribers. The Entity-centric Matcher will find a match only if the publications and the subscriptions concern the same entity (Exact Matching). If a match is found, then, the publications of the matched entity will be integrated into a window depending on their data type (Data Integration), one for object-type properties and one for data-type (numerical) properties. As each window gets populated, each element (triple) is transformed into a vector through an embedding model (Embedding-based Triple Vectors), and an initial clustering takes place (DBVARC Phase 1). In the case of numerical data, an additional incremental Aggregation occurs for

Table 2. Overall Map of Challenges (Cs), Requirements (Rs), Research Questions (RQs), and Solutions/Techniques

Rs	Cs: Solutions/Techniques
R1	C5: Simple, expressive, and semantically/schematically decoupled subscription model that
R2	needs only the entity name from the users (RQ1)
R2	C1: Diversity-aware subscription model (RQ1) C2: Diversity-aware subscription model with user-defined summary size (RQ1) C3: Subscription model with windowing policy (RQ1) C4: Diversity-aware subscription model (RQ1)
R3	C1: Diverse top-k summaries and high-level abstractions of numerical data (extractive and abstractive summaries) (RQ2) C2: Diverse summaries by conceptual similarity and duplicates elimination through word embedding models, conceptual clustering, and top-k ranking (extractive summary) (RQ2) C3: Data integration from multiple sources/databases to entity-based windows (RQ2) C4: High-level abstractions of numerical data through aggregation and reasoning (abstractive summary) (RQ2) C5: Graph-based event model that represents conceptual entities with their associated properties or background information and it is understandable to the users (RQ2)
R4	C1 & C2: Ground truths and correctness evaluation (RQ1, RQ2) C4: Correctness evaluation (RQ1, RQ2)
R5	C6: PSS that integrates unbounded and dynamic data to entity-based windows, which provide entity summaries and undergo performance evaluation (RQ1, RQ2)

Fig. 3. Architecture of an Entity-centric Publish/Subscribe Summarization System (PoSSUM).

each incoming value. Once the windows reach their full capacities based on a subscription-defined window size, a refined clustering takes place (DBVARC Phase 2). At this point, the numerical data undergoes an additional step of Reasoning, where high-level inferences are extracted based on their aggregated value so far. Then, all triples within each conceptual cluster are ranked based on their importance (Triple2Rank). The top-k triples for each window are then selected based on their overall ranks (Top-k Selection). Once all top-k triples are available, a Global Top-k Selection

occurs to pick a diverse set of the most important triples from both windows. These top-k triples form the summary, which is the payload of the Graph-based Notifications that will be sent to the subscribers (Event Model). Then the process starts again as long as the relevant subscription is active.

5.2 Subscription and Event Model

The subscription model is of the format $\langle \text{subscriberID}, \text{subscriptionID}, \text{timestamp}, \text{payload} \rangle$. The subscription payload is a set of simple attribute-value pairs and each event needs to fulfill all of its constraints so that a match occurs. For example, in the payload $\langle \text{entity} = \text{"Patient"}, k = 5, \text{windowType} = \text{"CountTumbling"}, \text{windowSize} = 10, \text{summary} = \{\text{"Extractive"}, \text{"Abstractive"}\} \rangle$, the subscriber is interested in the top 5 information concerning the entity "Patient" via an extractive and abstractive summary deriving from the analysis of data taken from count tumbling windows of size 10, that is, 10 publications. The summary's value can be either "Extractive," "Abstractive," or "None," should the user want no data summary.

The notifications (events) sent to the subscribers are of the format $\langle \text{notificationID}, \text{timestamp}, \text{payload} \rangle$. The payload is the resulting summary, which consists of a graph structure (collection of triples) with the k most important data coming from different sources concerning an entity. An example of a summary for the subscription above is given below (based on the publications of Figure 1), where the subject is the entity in question, the object is either the original value (extractive summary) or the reasoning result (abstractive summary) deriving from the aggregated value within the specific window and the rules, and the property is the relation between the subject and the object: $\{ \langle \text{Patient}, \text{SpO2}, \text{NORMAL} \rangle; \langle \text{Patient}, \text{pulse}, \text{NORMAL} \rangle; \langle \text{Patient}, \text{birthPlace}, \text{Russia} \rangle; \langle \text{Patient}, \text{bloodPressure}, \text{NORMAL} \rangle; \langle \text{Patient}, \text{medicalRecord}, \text{DIABETES MELLITUS W/OUT} \rangle \}$.

6 POSSUM APPROACH

In this section, details of the PoSSUM approach are given. Its main steps are (1) Embedding-based Triple Vectors, where a triple is transformed into a vector based on an embedding model; (2) DBVARC, a novel parameter-free and partly incremental conceptual clustering based on density and variance; and (3) Triple2Rank, a novel methodology for measuring the importance of a triple within and among all conceptual clusters. More details are given below.

6.1 Embedding-based Triple Vectors

The first step of the approach is to turn a triple into an embedding-based vector. This could be achieved by using either knowledge graph entity embeddings (e.g., TransE [7], RDF2Vec [44]) that focus on the graph structure or word embeddings (e.g., Word2Vec [30]) that focus on the semantic importance of words. In this article, the work focuses only on star-like graphs, and due to a diversity-oriented summarization goal, it is deemed that word embeddings are more suitable for representing triples as vectors. Nevertheless, word embeddings cannot be used directly on triples; therefore, a different approach needs to be defined as in our previous work PubSum [35].

This is an incremental step of PoSSUM, where for each triple of a generated publication that enters the corresponding window, the property and the object of the triple are extracted. Then, they are pre-processed. Specifically, the property undergoes tokenization, stop-words removal, lower-casing, and concatenation of tokens with an underscore and a dash, and all words ranging from original to tokens to concatenated ones are stored into an index (*propertiesIndex*). Similarly, the object's type/types undergo the same pre-processing, and all words are stored into another index (*objectsIndex*). An embedding model transforms the indices' words to their equivalent vectors and they are all stored in another index (*word2VecIndex*). Specifically, if the model does not contain

Object-type property	Triple information	Pre-processing	Word Vector Extraction
<Patient> <areaServed> <United_States>	Property: areaServed Object: United_States Object's types: Country, ..., SpatialThing	propertiesIndex for "areaServed": areaserved, area_served, area-served, area, served objectsIndex for "United_States": country, ..., spatialthing, spatial_thing, spatial-thing, spatial, thing	word2VecIndex for "areaServed": average of the vectors "area" and "served" word2VecIndex for "area": vector of "area" word2VecIndex for "served": vector of "served" word2VecIndex for "country": vector of "country" word2VecIndex for "spatialThing": vector of "spatial_thing" word2VecIndex for "spatial": vector of "spatial" word2VecIndex for "thing": vector of "thing"
Final triple vector: average of the vectors areaServed, area, served, country, ..., spatialThing, spatial, thing			

Fig. 4. The transformation of the triple ⟨Patient, areaServed, United_States⟩ to an embedding-based vector. In the example, the original word in the form of “areaserved,” “area_served,” or “area-served” was not contained in the embedding model; therefore, the average of the vectors of its tokens (area, served) was considered. This is not the case for “spatialthing” as the “spatial_thing” word existed in the model.

the original word (original property/type), then priority is given to the concatenated ones and, finally, to the average of the vectors of the word’s tokens. The final triple vector is the average of the vectors of the original words and their tokens. For data-type (numerical) properties, this process is only done for the property and the object is aggregated incrementally, where its mean value is calculated. The indices are used for optimization purposes so that already processed words are not examined again. An example of this process is given in Figure 4.

6.2 DBVARC: Density-based VARIance Clustering

We propose DBVARC with the notion that triples are partitioned based on similarity, resulting in clusters with elements that are similar to their cluster’s elements and dissimilar to the other clusters’ elements. In the case of DBVARC, this is translated to finding the densely connected regions in the semantic space derived from the Embedding-based Triple Vectors step.

In general, density-based algorithms are more suitable for evolving data streams due to:

- *No pre-defined number of clusters parameter.* Knowing a priori the correct value of this parameter is not possible, especially in streaming and evolving data that could lead to a change of the number of clusters. In static data, this parameter could be tuned for its best value, but this is not applicable in streaming data.
- *No restrictions on the size and the shape of the clusters.* In streaming data, an area with similar or related characteristics in a vector space could be of any shape and population.

Nevertheless, there is no density-based algorithm even for streaming data that is completely parameter-free, meaning that tuning needs to take place, which is not possible in streaming environments. DBVARC does not demand any pre-defined parameters and it is partly incremental, making it suitable for streaming environments. The algorithm is split into two phases: an incremental phase 1, where initial clusters are created, and a batch phase 2, where clusters are examined for further partitioning if they contain elements that are densely intra-connected and loosely inter-connected. More details are given in Algorithms 1 and 2.

Algorithm 1 is the incremental phase 1 of the DBVARC, where each triple is put in an initial cluster. In line 1, the DBVARC function gets as input the *statementVector*, which is the time-stamped triple vector derived from the Embedding-based Triple Vectors step. The function aims to create or update micro-clusters (*microClusters*), which are tuples that contain not only the elements of a cluster but also their statistics involving the linear sum of the elements’ vectors and the cluster’s centroid. Specifically, in line 4, the first micro-cluster is created. Its linear sum of the elements’ vectors is calculated in line 14, its centroid is the average of all the elements’ vectors in line 15, and its elements contain the vectors in line 16. The created micro-cluster is added to the list

ALGORITHM 1: DBVARC Phase 1 - Density-based VARiance Clustering (Initial Clusters)

```

1: function DBVARC(statementVector)                                ▷ statementVector: timestamped triple vector of window
2:   if  $\neg \text{endOfWindow}$  then                                       ▷ phase 1 - initial clusters
3:     if microClusters  $\leftarrow \emptyset$  then
4:       createNewCluster(statementVector)
5:     else
6:       for microCluster  $\leftarrow$  microClusters do
7:         closestCluster  $\leftarrow$  maxCosineSimilarity(microCluster.centroid, statementVector.vector)
8:         if maxCosineSimilarity > 0.5 then
9:           updateCluster(closestCluster, statementVector, addition)
10:        else
11:          createNewCluster(statementVector)
12:   return microClusters                                           ▷ initial set of clusters
13: function CREATENEWCLUSTER(statementVectors)
14:   sumOfElements  $\leftarrow \sum$  statementVectors.vector
15:   centroid  $\leftarrow \frac{\text{sumOfElements}}{\text{sizeOfStatementVectors}}$ 
16:   clusterElements  $\leftarrow$  statementVectors
17:   microClusters  $\leftarrow$  MicroClusterTuple(centroid, sumOfElements, clusterElements)
18: function UPDATECLUSTER(cluster, statementVector, status)
19:   if status  $\leftarrow$  addition then
20:     sumOfElements  $\leftarrow$  cluster.sumOfElements + statementVector.vector
21:     clusterElements  $\leftarrow$  cluster.clusterElements.add(statementVector)
22:   if status  $\leftarrow$  deletion then
23:     sumOfElements  $\leftarrow$  cluster.sumOfElements - statementVector.vector
24:     clusterElements  $\leftarrow$  cluster.clusterElements.remove(statementVector)
25:   centroid  $\leftarrow \frac{\text{sumOfElements}}{\text{sizeOfClusterElements}}$ 
26:   microClusters  $\leftarrow$  MicroClusterTuple(centroid, sumOfElements, clusterElements)

```

of *microClusters* in line 17. If a new micro-cluster involves only one vector as in the case of line 4, then the tuple's elements are the vector itself. In line 5, each new *statementVector* is examined for its closest micro-cluster. Specifically, in line 7, cosine similarity is calculated between the new vector and the centroids of the micro-clusters. The most similar micro-cluster is the closest one. If the similarity is greater than a good starting point threshold that suggests strong similarity (line 8), the new vector will be merged/added to the closest micro-cluster (line 9), leading to the update of the latter (lines 18–26); otherwise, a new micro-cluster will be created that contains the new vector (line 11). All micro-clusters are stored in the list *microClusters* for the duration of a window and the list is returned in line 12 for further use. An example of DBVARC phase 1 is given in Figure 5.

Algorithm 2 is the batch phase 2 of the DBVARC, where the existing clusters are refined. In line 1, the DBVARC function gets as input the *microClusters* and *statementVectors* of the previous steps. The function aims to check the variance of the elements assigned in clusters, and if it exceeds a dynamic threshold dependent on the data, then the elements are either merged with the next closest cluster or split into their own cluster. Specifically, in line 5, each element of a micro-cluster is checked for its variance as calculated in lines 13–18. In line 14, the mean of the cosine similarities between the cluster centroid and the cluster's elements' vectors is used to define a dynamic threshold (line 4). If the cosine similarity of a cluster element is lower than the threshold (line 16), then the element has high variance and is stored in *elements* (line 17) for further examination. In line 8, each element with high variance is examined for its neighboring clusters defined by lines 19–21, where all cosine similarities between the element's vector and the clusters' centroids are sorted in descending order and the 30% neighboring clusters are extracted (line 21) since the other clusters would be much further away to be even considered as neighbors. Line 9 defines whether the element will be merged to one of the neighboring clusters or split into a new one. This decision follows the logic of DBVARC phase 1 and is made in lines 22–29, where the closest triple in a neighboring cluster is defined based on its cosine similarity with the element's

ALGORITHM 2: DBVARC Phase 2 - Density-based VARiance Clustering (Refinement of Clusters)

```

1: function DBVARC(microClusters, statementVectors)
2:   if endOfWindow then                                     ▷ phase 2 - refinement of clusters
3:     for microCluster  $\leftarrow$  microClusters do
4:       threshold  $\leftarrow$  round(mean) - 0.1
5:       elements  $\leftarrow$  highVarianceElements(microCluster, statementVectors, threshold)
6:       for microCluster  $\leftarrow$  microClusters do
7:         for statementVector  $\leftarrow$  elements do
8:           sortedClusters  $\leftarrow$  neighbourClusters(microClusters, statementVector)
9:           triplesToBeSplit  $\leftarrow$  mergeOrSplit(statementVector, sortedClusters, microCluster)
10:          if  $\neg$ triplesToBeSplit  $\leftarrow$   $\emptyset$  then
11:            split(microCluster, triplesToBeSplit)
12:          return microClusters                                     ▷ final set of clusters
13: function HIGHVARIANCEELEMENTS(microCluster, statementVectors, threshold)
14:   mean  $\leftarrow$  meanCosineSimilarity(microCluster.centroid, statementVectors.vector)
15:   for clusterElement  $\leftarrow$  microCluster.clusterElements do
16:     if cosineSimilarity < threshold then
17:       elements  $\leftarrow$  clusterElement
18:   return elements
19: function NEIGHBOURCLUSTERS(microClusters, statementVector)
20:   sortedClusters  $\leftarrow$  sort(cosineSimilarity(microClusters.centroid, statementVector.vector))
21:   return 30% of sortedClusters
22: function MERGEORSPLIT(statementVector, sortedClusters, oldCluster)
23:   for microCluster  $\leftarrow$  sortedClusters do
24:     closestCluster  $\leftarrow$  maxCosineSimilarity(microCluster.vectors, statementVector.vector)
25:     if maxCosineSimilarity > 0.5 then
26:       merge(closestCluster, oldCluster, statementVector)
27:     else
28:       triplesToBeSplit  $\leftarrow$  statementVector
29:   return triplesToBeSplit
30: function MERGE(closestCluster, oldCluster, statementVector)
31:   updateCluster(oldCluster, statementVector, deletion)
32:   updateCluster(closestCluster, statementVector, addition)
33: function SPLIT(microCluster, triplesToBeSplit)
34:   createNewCluster(triplesToBeSplit)
35:   threshold  $\leftarrow$  round(mean - 0.02, 2)
36:   elements  $\leftarrow$  highVarianceElements(microCluster, triplesToBeSplit, threshold)
37:   if  $\neg$ elements  $\leftarrow$   $\emptyset$  then
38:     updateCluster(oldCluster, elements, deletion)
39:   createNewCluster(elements)

```

vector (line 24). The cluster that this triple belongs to is the closest cluster to the element. In this way, we examine not only the distance between cluster elements in regards to their centroid but also to elements that could lie on the edge of a cluster. If the cosine similarity with the closest triple is higher than a threshold that suggests strong similarity (line 25), then the element is merged to its closest cluster (line 26), leading to the updates of the old cluster and the cluster to be merged (lines 30–32). Otherwise (line 27), the element will be split and stored in *triplesToBeSplit* for further use (line 28). In line 11, the elements belonging to a micro-cluster that are soon to be split are examined. Specifically, a new cluster with all of the elements is created in line 34. Once again, the variance of the elements is examined in line 36, and according to a dynamic threshold (line 35), the elements will either remain in the new cluster or the elements with high variance will form their own cluster (lines 38–39). All new or updated micro-clusters are stored in the list *microClusters* and the list is returned in line 12 for further use. An example of DBVARC phase 2 is given in Figures 5 and 6.

6.3 Triple2Rank

We propose Triple2Rank, which is the final step of the approach, where each triple is ranked according to its importance. The ranking is based on the concept of taxonomies, where a hierarchical structure of words is created starting from words with basic-level concepts (parents)

Fig. 5. An example of the two phases of DBVARC for the DBpedia entity “Usain Bolt.” The squares represent a cluster (C1–C10) and the stars are their centroids. The bold squares are pure clusters (all elements are clustered correctly) based on the ground truth. We observe that in phase 1 two clusters are found correctly (C1, C5), whereas in phase 2 the refinement of three clusters (C2, C3, C4) led to the creation of four more pure clusters (C2, C7, C8, C9). In clusters C2, C3, and C4, some elements are loosely inter-connected; therefore, they either form a separate common cluster (C7, C10) or are split into more clusters (C8, C9). The process is further detailed in Figure 6.

that are gradually split into words of refined-level concepts (children) that could be subsumed by their parents. The ranking is based on the following rules:

- Basic-level concepts are characterized by shorter and more polysemous words [48]. Therefore, the fewer tokens a word contains and the shorter it is, the higher it should be in the hierarchy.
- Values that are not only more popular (frequent) but also more informative (rare) than others should be higher in the hierarchy.
- Values that belong to an abnormal/extreme low or high range are more important than others and should be higher in the hierarchy.

Algorithm 3 is a ranking methodology that measures the importance of a triple within and among all conceptual clusters of an entity. In line 1, the Triple2Rank function gets as input the *microClusters* of Algorithm 2 and the *Reasoner*, which contains the reasoning rules. The function aims to score the triples of the clusters based on significance, but also to define a selection order that chooses which triples will be selected first for the final summary. Line 2 proceeds with the scoring of each triple that is explained in lines 8–14. For each triple of a cluster, a separate scoring is calculated for the property (lines 15–16) and the object (lines 17–22). Since the property is an actual word, a penalty is given to it according to the number of tokens that it contains and its length. The final property score is given in line 16, meaning that the more tokens a word contains or the longer it is, the higher the penalty. On the other hand, the object is either an entity or a number. If

DBVARC Phase 1 Initial Clusters: C1: {t1, t2, t3, t10, t11, t12, t13, t15, t17}, C2: {t4, t6, t18}, C3: {t5, t16, t27, t32, t33}, C4: {t7, t8, t9, t19, t20, t22, t23, t24, t25, t26, t28, t29, t30}, C5: t14, C6: {t21, t31}			
DBVARC Phase 2 - High Variance Elements Step			
Cluster: C1	Cluster: C2	Cluster: C3	Cluster: C4
Distance(t1, c1) = 0.918 ... Distance(t17, c1) = 0.901 Mean = 0.946 High Variance Elements: -	Distance(t4, c2) = 0.880 Distance(t6, c2) = 0.792 Distance(t18, c2) = 0.908 Mean = 0.860 High Variance Elements: t4, t6	Distance(t5, c3) = 0.792 Distance(t16, c3) = 0.787 Distance(t27, c3) = 0.910 Distance(t32, c3) = 0.910 Distance(t33, c3) = 0.892 Mean = 0.858 High Variance Elements: t5, t16, t33	Distance(t7, c4) = 0.601 Distance(t8, c4) = 0.779 Distance(t9, c2) = 0.758 Distance(t19, c4) = 0.982 ... Distance(t30, c4) = 0.982 Mean = 0.920 High Variance Elements: t7, t8, t9
Updated Clusters: C2: {t18}, C3: {t27, t32}, C4: {t19, t20, t22, t23, t24, t25, t26, t28, t29, t30}			
DBVARC Phase 2 - Find neighbor clusters & Merge or Split Decision Step			
Elements: t4, t6	Elements: t5, t16, t33	Elements: t7, t8, t9	
Closest cluster to t4: C1 Closest point: t12 Closest distance = 0.248 < 0.5 => split	Closest cluster to t5: C1 Closest point: t10 Closest distance = 0.364 < 0.5 => split	Closest cluster to t7: C6 Closest point: t21, t31 Closest distance = 0.220 < 0.5 => split	
Closest cluster to t6: C5 Closest point: t14 Closest distance = 0.231 < 0.5 => split	Closest cluster to t16: C6 Closest point: t21, t31 Closest distance = 0.306 < 0.5 => split	Closest cluster to t8: new C3 Closest point: t27, t32 Closest distance = 0.248 < 0.5 => split	
-	Closest cluster to t33: C6 Closest point: t21, t31 Closest distance = 0.387 < 0.5 => split	Closest cluster to t9: new C3 Closest point: t27, t32 Closest distance = 0.224 < 0.5 => split	
DBVARC Phase 2 - Split Step			
Cluster: C7	Cluster: C8	Cluster: C9	
Distance(t4, c2) = 0.883 Distance(t6, c2) = 0.850 Mean = 0.867 High Variance Elements: -	Distance(t5, c3) = 0.868 Distance(t16, c3) = 0.895 Distance(t33, c3) = 0.789 Mean = 0.851 High Variance Elements: t33 Put t33 in another cluster and END	Distance(t7, c4) = 0.968 Distance(t8, c4) = 0.986 Distance(t9, c2) = 0.958 Mean = 0.971 High Variance Elements: - END	
Final Clusters: C1: {t1, t2, t3, t10, t11, t12, t13, t15, t17}, C2: {t18}, C3: {t27, t32}, C4: {t19, t20, t22, t23, t24, t25, t26, t28, t29, t30}, C5: t14, C6: {t21, t31}, C7: {t4, t6}, C8: {t5, t16}, C9: {t33}, C10: {t7, t8, t9}			

Fig. 6. The process of the “Usain Bolt” example in Figure 5. After the initial clusters from DBVARC phase 1 (C1–C6), we show how clusters with size >1 or mean <1.0 are examined for elements with high variance. These elements are deleted from the existing clusters (updated C2, C3, C4), their neighbor clusters are found, and their closest element from each neighbor cluster decides whether they will be merged to this cluster or split into their own. The existing elements are all split into their own clusters (C7, C8, C10) apart from t33, which presented high variance to the new cluster; therefore, it formed its own cluster (C9).

the object is numerical (aggregated value), then the Reasoner is called and its high-level inference is deducted that replaces the numerical value (lines 18–19). The score given is based on the value range from which the inference was taken (line 20). If the object is an entity, the score given is based on a cluster-based tf-idf. In this modified tf-idf, the cluster the object belongs to is the document and all clusters are the corpus. Therefore, the tf is the number of the object’s occurrence within the cluster and the idf is based on the uniqueness of the object among all clusters (line 22). The triple score is the combination of its property’s and object’s score in line 11. A *scoredStatement* is then created in line 12 that contains the timestamped scored triple based on the publication’s original

ALGORITHM 3: Triple2Rank

```

1: function TRIPLE2RANK(microClusters, Reasoner)                                ▷ Reasoner: reasoning rules
2:   scoredStatementsOfClusters ← score(microClusters)
3:   averageScorePerCluster ← averageScore(scoredStatementsOfClusters)
4:   orderedClusters ← sort based on averageScore or sizeOfCluster if averageScore is tied
5:   while ¬scoredStatementsOfClusters ← ∅ do
6:     scoredStatements ← startSelectionRound(orderedClusters, scoredStatementsOfClusters)
7:   return scoredStatements                                                    ▷ ordered timestamped scored triples of window
8: function SCORE(microClusters)
9:   for microCluster ← microClusters do
10:    for triple ← microCluster.clusterElements do
11:      finalScore ← propertyScore(propertyOfTriple) + objectScore(objectOfTriple)
12:      scoredStatementsOfCluster ← ScoredStatement(timestamp, triple, finalScore)
13:      scoredStatementsOfClusters ← sort(scoredStatementsOfCluster)
14:   return scoredStatementsOfClusters
15: function PROPERTYSCORE(propertyOfTriple)
16:   return 100 + (1 − numberOfTokensOfProperty) + (1 − lengthOfProperty)
17: function OBJECTSCORE(objectOfTriple)
18:   if objectOfTriple is numerical then
19:     objectOfTriple ← Reasoner(objectOfTriple)
20:     return score based on value range of reasoning rules
21:   else
22:     return frequencyOfObjectInCluster * log  $\frac{\text{numOfClusters}}{\text{numOfClustersThatContainObject}}$ 
23: function STARTSELECTIONROUND(orderedClusters, scoredStatementsOfClusters)
24:   for cluster ← orderedClusters do
25:     for scoredStatement ← scoredStatementsOfCluster do
26:       if objectOfTriple is from Reasoner then
27:         if objectOfTriple not selected or end is reached then
28:           scoredStatements ← scoredStatement
29:           scoredStatementsOfCluster.remove(scoredStatement)
30:       else
31:         if propertyOfTriple not selected or end is reached then
32:           scoredStatements ← scoredStatement
33:           scoredStatementsOfCluster.remove(scoredStatement)
34:   return scoredStatements

```

timestamp and it is stored in the list *scoredStatementsOfCluster*. The list is sorted in descending order in line 13 and it is returned in line 14 for further use. This list is then used for defining the average score of the triples within each cluster (line 3). The average scores are used in line 4 for defining the order in which each cluster will be selected for choosing the best every time triple for the final summary. Priority is given to clusters with a higher average score, but if the score is tied, then the bigger cluster will be selected first. In lines 5–6, the selection round is taking place, described in lines 23–34. This is a continuous process until all triples have been selected. In this process, a cluster is visited based on its order (line 24) and the highest-scored *scoredStatement* within the cluster is selected (line 25). If the object of this triple contains a high-level inference (line 26), then, if it has not already been selected in another round or if all objects of a cluster have already been selected once in other rounds (line 27), the *scoredStatement* is chosen as the next to be selected for the final summary in line 28 and it is removed from the collection of *scoredStatements* in line 29; otherwise, the next-highest-scored *scoredStatement* within the cluster is selected, and so on. On the other hand, if the object is an entity, then the same selection applies but for the properties instead of objects (lines 30–33). This provides more diversity in the final summary. The final order of the selected *scoredStatements* is given in line 7 and it is used for top-k selection. An example of Triple2Rank is given in Figure 7.

7 EVALUATION

The datasets, methodology, metrics, and results regarding two use cases, Healthcare and Smart Cities, are analyzed below.

Triple2Rank		
Triple & Rules	Triple Scoring	Selection (from Fig. 5)
<Usain_Bolt, birthPlace, Trelawny_Parish_Jamaica>	Number of tokens = 2, Length = 10 Property score = 90	Average score per cluster: {C1: 88.766, C2: 93.041, C3: 96.521, C4: 96.104, C5: 86.041, C6: 96.521, C7: 87.021, C8: 96.521, C9: 97.041, C10: 84.347} Ordered clusters: {C9, C3, C6, C8, C4, C2, C1, C7, C5, C10}
	tf = 2, idf = 1.0 (from Fig. 5) Object score = 2.0	Round: 1 C9: top triple: t33 => property not selected, select C3: top triple: t32 => property not selected, select ... Round: 4 C1: top triple: t2 => property "birthPlace" selected, skip top triple: t13 => property not selected, select ... Order of triples: t33, t32, t31, t16, t30, t18, t15, t4, t14, t8, t27, t21, t5, t29, t10, t6, t9, t28, t3, t7, t26, t13, t25, t17, t24, t2, t23, t1, t22, t12, t20, t11, t19 Top-5: t33, t32, t31, t16, t30
<Usain_Bolt, heartRate, 100.8> Aggregated value = 100.8 Rules: heartRate > 100 => TACHYCARDIA 60 ≤ heartRate ≤ 100 => NORMAL heartRate < 60 => BRADYCARDIA	Number of tokens = 2, Length = 9 Property score = 91	
	Inference = TACHYCARDIA (top value range) Object score = 30 (max score)	

Fig. 7. An example of Triple2Rank. The triple scoring is shown for both cases, where the object is an entity “Trelawny_Parish_Jamaica” or an aggregated number “100.8.” In the case of the numerical object, the reasoning result based on the rules is TACHYCARDIA, which belongs to the top value range of the rules [min-60, 60-100, 100-max]; therefore, the maximum score is given to the object. The triple selection is also presented for the triples of Figure 5. C3, C6, and C8 have tied average scores, but they also have the same cluster sizes; therefore, the order among the three clusters is not important. In terms of selection rounds, in Round 4, the “birthPlace” property has already been chosen before (t3) from C1; therefore, even if t2 is the top-scored triple in the cluster, the next one (t13) will be examined, and so on.

7.1 Datasets

A combination of different real-world datasets has been used (Table 3). Each entity dataset has been synthetically generated through either different real-world entity DBpedia data (FACES), patient data (MIMIC II Database), or 24-hour sets of readings (Intel Lab Data, CityPulse, and UCI Electric Consumption). The manual reasoning rules were mostly created from the M3 Framework (<https://github.com/gyrard/M3Framework/tree/master/war/RULES>) and the annotations of the MIMIC II Database, which contained ranges of attributes’ values when an alert occurred.

7.2 Methodology

The evaluation methodology is illustrated in Figure 8. The effectiveness evaluation is based on three types of ground truth: relevance, conceptual, and user defined. The relevance ground truth was created by choosing each original property (word) (step I in Figure 8) and storing its synonyms, related words, and antonyms deriving from multiple thesauri as well as its hyponyms (children) observed in ontologies (step II). In the case of polysemy, we chose the words that applied to the application in use (e.g., “pressure” as body measurement in the first use case and as an environmental one in the second case). If the original word did not exist in the thesauri, then its separate tokens were considered. The conceptual ground truth proved more challenging as most summarization approaches rely only on users to evaluate their results; nevertheless, this type of evaluation is dependent on the users at hand and their experience, resulting in a more subjective view. Therefore, we created a conceptual ground truth that is based on context coherence; that is, we observed the number of common words between each original property pair (minimum common words’ number was 3) (step III) as well as if words were directly linked to other words with a relatedness rank via thesauri (e.g., Roget’s thesaurus) (step IV). Also, we checked in taxonomies if any words belonged to the same category so that they could be clustered together (step V). In both ground-truth types, we used multiple thesauri to provide a more reliable gold standard. The user-defined ground truth (step VI) is provided by FACES, in which 15 human judges were asked to select ideal triples for specific entities (for $k = 5$ and $k = 10$). This ground truth was used in our evaluation for

Table 3. Datasets Used for the Healthcare and Smart Cities Use Cases

Dataset	Use Case	Data/Sensors	Pre-processing
FACES ¹	All	DBpedia (people, places, and buildings)	Deletion of data-type properties (textual or numerical) and duplicates.
MIMIC II Database ²	Health-care	Heart readings, annotations, medical history	The readings and medical history turned to triples (e.g., ⟨Patient, centralVenousPressure, 19.800⟩). Noise and missing values were deleted. The annotations were used for reasoning rules.
Intel Lab Data ³	Smart Cities	Temperature, voltage, light, humidity	The readings turned to triples (e.g., ⟨DiningTable1, humidity, 40.399⟩). Missing values replaced with nearby ones.
CityPulse ⁴	Smart Cities	Weather, pollution, road traffic, cultural events	The readings turned to triples (e.g., ⟨Amsterdam, particulateMatter, 67⟩). The cultural events were translated from Danish to English, turned to triples (e.g., ⟨Amsterdam, culturalEvent, Chamber_concert⟩) and their types were used as typing information.
UCI Electric Consumption ⁵	Smart Cities	Electric readings	The readings turned to triples (e.g., ⟨Kitchen, activeEnergy, 36.000⟩).

providing a more complete effectiveness evaluation (step VII). On the other hand, the efficiency evaluation (step VIII) is based on metrics like latency, throughput, and so forth. It is taking place after the final ranking has occurred for all triples.

The thesauri that have been used for the methodology described above for both use cases are Merriam-Webster (<https://www.merriam-webster.com/thesaurus>), Thesaurus (<https://www.thesaurus.com/>), and Roget's Thesaurus (<http://www.roget.org/>). The taxonomy IPTC Subject Codes (<https://docs.aylien.com/newsapi/search-taxonomies/#search-labels-for-iptc-subject-codes>) also acted as a reference. For the Healthcare use case, the SNOMED CT taxonomy (<https://browser.ihtsdo.org/?>) was used, which is an organized collection of medical terms (e.g., central venous pressure etc.), whereas for the Smart Cities use case the BRICK ontology (<https://brickschema.org/#home>) was examined for selecting synonyms/related words up to level 0 to 2 deep, which contains words regarding the measurements of buildings (e.g., pressure, humidity, luminance etc.).

7.3 Metrics

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

¹<http://wiki.knoesis.org/index.php/FACES>.

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>.

Fig. 8. The evaluation methodology and an example for the properties temperature, humidity, dewPoint, and pressure (I). After their synonyms, related words, antonyms, and hyponyms from thesauri have been found (relevance ground truth II), their most common words are analyzed. There is a high commonality between temperature, humidity, and pressure; nevertheless, pressure has some commonality with other properties unrelated to temperature and humidity like globalReactivePower, globalActivePower, and windDirection (III). To define to which conceptual cluster pressure belongs, we observe their direct links among them from thesauri. It is shown that temperature, humidity, and dewPoint have high relevance, and pressure is also linked to them but not so strongly; nevertheless, the link between temperature, humidity, and dewPoint is higher than the one with globalReactivePower and globalActivePower (IV). Therefore, the final conceptual cluster is {temperature, humidity, dewPoint, pressure}. The other properties are analyzed in the same way.

7.3.1 Correctness. Correctness consists of the agreement, diversity consensus, and quality metrics by using the three aforementioned types of ground truth: relevance, conceptual, and user defined.

Agreement and Diversity Consensus. The FACES agreement Agr defines how consistent the ideal summaries are among the judges in the user-defined ground truth, whereas the proposed diversity consensus $DivCon$ shows the degree of the selected triples by the judges that belong to different conceptual clusters in the conceptual ground truth. These are defined as

$$Agr = \frac{2}{n(n-1)} \sum_{i=1}^n \sum_{j=i+1}^n |S_i^I(e) \cap S_j^I(e)| \quad \text{and} \quad DivCon = \frac{1}{n} \sum_{i=1}^n \begin{cases} \left| \frac{CC(S_i^I(e))}{S_i^I(e)} \right|, & \text{if } |S_i^I(e)| \leq |CC(e)| \\ \left| \frac{CC(S_i^I(e))}{CC(e)} \right|, & \text{otherwise,} \end{cases} \quad (1)$$

where n is the number of summaries, $S_i^I(e)$ is the i th ideal summary for an entity e , and $CC(S_i^I(e))$ as well as $CC(e)$ are the distinct conceptual clusters belonging to the i th ideal summary and an entity e , respectively.

Quality. The quality refers to either the final summary/notification (after ranking) that takes place or the conceptual clustering (before ranking). The former involves the FACES QUD and proposed $RS_F - score$ metrics, whereas the latter involves the proposed $RCC_F - score$ metric.

QUD is the quality user-defined metric that is based on the commonalities between the approach's summaries and the ideal ones in the user-defined ground truth defined as

$$QUD(S(e)) = \frac{1}{n} \sum_{i=1}^n |S(e) \cap S_i^I(e)|, \quad (2)$$

where $S(e)$ is the approach's summary.

$RS_F - score$ and $RCC_F - score$ are the redundancy-aware F-score of the summary and the conceptual clustering, respectively. Regarding the $RS_F - score$, we adopt the metric defined in Zhang et al. [52] and adapt it to cater for conceptually similar information (not only duplicates). All words that were selected by the approach to belong to the same conceptual cluster and that are indeed in the same cluster based on the conceptual ground truth are considered as **True Positives** (TP). On the other hand, if they do not belong to the same cluster based on the ground truth, then they are considered as **False Positives** (FP). The words that the approach put in different conceptual clusters but that belong to the same cluster based on the ground truth are considered as **False Negatives** (FN). Similarly, regarding the summary, the TP could refer to the set of non-delivered redundant triples (R^-), the FP to the set of non-delivered non-redundant triples (N^-), and FN to the set of delivered redundant triples (R^+). Therefore, the $RS_F - score$ is

$$precision = \frac{R^-}{R^- + N^-} \quad \text{and} \quad recall = \frac{R^-}{R^- + R^+} \quad \text{and} \quad RS_F - score = 2 \times \frac{precision \times recall}{precision + recall}. \quad (3)$$

For the $RCC_F - score$, each conceptual cluster of the ground truth is checked against each conceptual cluster of the approach's summary and then, based on the commonality percentage of the attributes, each ground-truth cluster is mapped to its most representative one of the approach's summary (the one with the highest commonality percentage). In this way, we take into account non-pure clusters. The $RCC_F - score$ is defined similarly to $RS_F - score$ by replacing R^- with TP , N^- with FP , and R^+ with FN . The difference between the two metrics is that $RCC_F - score$ shows the effectiveness of the approach on detecting conceptually similar words, whereas the $RS_F - score$ shows the loss of important information that took place via ranking and sending only a subset of the whole available information (top-k).

7.3.2 Performance. The system performance consists of the processing time of individual processes like clustering or scoring as well as the end-to-end latency, which covers the time from the generation of the earliest publication that belongs to a summary to the time this summary has been sent as a notification to the subscriber. Also, the throughput is examined, which corresponds to the number of triples/events the system receives (throughput in) and the one that it extracts in the form of summaries (throughput out) in a specific amount of time. The final performance metrics are the message reduction, which is the reduction in the number of triples within the graph-based notification that is sent to the subscriber, and the memory used by the different approaches by either embedding models or ontologies.

7.4 Results

PoSSUM is compared against the aforementioned FACES (COBWEB cut-off = 5, path level = 3 from original paper) and our previous work PubSum ($\epsilon = 1$, minPts = 1, GoogleNews Word2Vec model from original paper). Also, the naive No top-k approach is acting as a baseline, where no summary occurs and all publications are sent to the subscriber as notifications. The correctness evaluation has been done offline; that is, the results are examined on the assumption that all available data belongs in a window, whereas the performance evaluation is done online. All experiments were run five times and the average result was taken. The runs took place on a laptop with Intel(R) Core(TM) i7-6600U CPU@2.60GHz 2.80GHz and 16GB of RAM.

7.4.1 Agreement and Diversity Consensus. The average agreement among all entities regarding people, places, and buildings within FACES ideal summaries is $Agr = 1.917$ for $k = 5$ and $Agr = 4.587$ for $k = 10$. This means that approximately 2 out of 5 and 5 out of 10 triples were identical among different judges when their top choices for an entity summary were concerned. On the

other hand, the diversity consensus is $DivCon = 0.780$ for $k = 5$ and $DivCon = 0.651$ for $k = 10$. This shows that in almost 80% and 65% of the top 5 and top 10, respectively, ideal information of the judges is diverse, meaning that it belongs to different conceptual clusters.

These results prove that judges might not highly agree on which triples are important (humans' subjective nature), especially when less information is considered (the top 5 agreement is smaller than the top 10 one). Nevertheless, they also prove that judges do agree that the less information they are provided with, the more diversity they require. This is a good indication that a diverse entity summarization approach is deemed essential in heterogeneous and data-voluminous environments like IoT since it would not only resolve resource constraints of the environment but also not overwhelm the users with redundant information.

7.4.2 Quality. The quality of the final summary and the conceptual clustering is shown in Tables 4 and 5. Table 4 shows the overlap between the user-defined ground truth and the approach's summary for each entity. It is shown that on average, the best approach is PoSSUM for both top 5 and top 10 summaries. This does not indicate that it always performs the best for all entities since this happens for 17 out of 30 entities in top 5 summaries and 13 out of 30 in top 10 ones. The second-best approach is FACES with the best results for 9 out of 30 entities in top 5 summaries and 11 out of 30 in top 10 ones, leaving as last PubSum with 5 out of 30 and 7 out of 30, respectively. This shows that PoSSUM behaves the best by a large margin for shorter summaries (top 5) by capturing better the most ideal triples by the judges. In the case of larger summaries (top 10), PoSSUM still performs the best but very closely to FACES in regards to the number of best-performing entities. Nevertheless, especially for top 10 summaries, we observe that when PoSSUM is better than FACES, the quality may be even twice better. This is not the case when FACES is better, indicating that even though the number of the best entities for PoSSUM and FACES for the top 10 is close, the quality of PoSSUM is much better in total. In general, we observe that for all approaches there is never a very high quality, with some entities having a low one. This occurs as summarization creates a representative subset of the original information, meaning that some important information to the users will always be lost. Hence, there is a tradeoff between the summary quality and size; that is, the quality gets better for higher k as more triples are selected. Therefore, the chances of a summary containing an ideal triple by the judges are higher. Also, the quality is affected by the agreement Agr and diversity consensus $DivCon$ metrics. Specifically, the agreement showed that there is a slight discrepancy among judges of what is an ideal summary, whereas the diversity consensus showed that some judges are interested in conceptually similar information or even repetitive attributes. This dictates that the quality will never achieve the maximum possible results as the summary will not cover all user choices since the size is limited and it emphasizes diverse information only. The summarization could at best cover the highest percentage of the users' agreement for an entity.

The quality of the final summary is not only dependent on the triple ranking as depicted in Table 4 but also on the conceptual clusters that have been formed. This is not depicted in the QUD metric as it may lead to a good result for an entity, but at the same time, the entity's conceptual clustering might perform poorly. Therefore, $RS_F-score$ and $RCC_F-score$ shown in Table 5 give a more thorough view of the quality of all steps of an approach (triple's vector representation, conceptual clustering, and ranking). FACES is only analyzed for its dataset in regards to $RS_F-score$ since its triple-ranking methodology is semantically coupled; therefore, it could not be applied for other data than DBpedia. The rest of the datasets concern either data-type (numerical) properties (Numerical Healthcare and Numerical Smart Cities) or object-type properties (FACES dataset integrated with medicalRecord triples for the Object-type Healthcare dataset, and with culturalEvents for the Object-type Smart Cities one). All Healthcare (merged) and All Smart Cities (merged)

Table 4. Average QUD for Each Entity

Entity	QUD					
	FACES		PubSum		PoSSUM	
	k = 5	k = 10	k = 5	k = 10	k = 5	k = 10
Bill Gates	0.125	0.273	0.205	0.285	0.110	0.268
Richard Mentor Johnson	0.105	0.175	0.015	0.189	0.130	0.311
Albert Einstein	0.080	0.254	0.354	0.500	0.229	0.443
England	0.215	0.503	0.150	0.413	0.250	0.435
Joe Biden	0.249	0.491	0.029	0.221	0.111	0.265
Walt Disney	0.051	0.177	0.114	0.231	0.085	0.194
Rice University	0.473	0.562	0.326	0.478	0.225	0.443
Barack Obama	0.220	0.298	0.190	0.304	0.025	0.271
Benjamin Franklin	0.273	0.403	0.136	0.317	0.121	0.307
Alan Turing	0.126	0.323	0.125	0.277	0.206	0.226
Edward Teller	0.255	0.475	0.225	0.322	0.375	0.432
Mike Turner	0.206	0.357	0.200	0.369	0.366	0.346
Charles Darwin	0.097	0.248	0.137	0.318	0.219	0.292
Amsterdam	0.160	0.259	0.095	0.237	0.075	0.150
Vladimir Putin	0.046	0.210	0.057	0.210	0.063	0.218
Poland	0.030	0.247	0.035	0.143	0.400	0.545
John Napier	0.295	0.572	0.200	0.478	0.350	0.580
Czech Republic	0.215	0.358	0.030	0.278	0.425	0.483
Bruce Lee	0.098	0.240	0.221	0.324	0.071	0.456
Marie Curie	0.170	0.455	0.310	0.390	0.325	0.488
Henry IV of France	0.045	0.290	0.130	0.273	0.235	0.310
Seychelles	0.177	0.386	0.171	0.343	0.343	0.500
J. C. Penney	0.394	0.734	0.486	0.600	0.371	0.657
Ann Arbor, Michigan	0.205	0.407	0.155	0.307	0.100	0.279
Nile	0.139	0.291	0.120	0.311	0.133	0.383
Yukon River	0.151	0.418	0.160	0.478	0.413	0.413
Norway	0.200	0.366	0.069	0.220	0.314	0.563
Texas	0.137	0.333	0.000	0.191	0.093	0.247
Friedrich Engel (mathematician)	0.130	0.270	0.230	0.347	0.300	0.360
Usain Bolt	0.360	0.390	0.149	0.259	0.309	0.357
All (average)	0.181	0.359	0.161	0.320	0.226	0.371

datasets refer to the results when both data-type (numerical) and object-type properties are considered and a merged top-k ranking is taking place for the final notification to the subscriber.

RCC_F – score depicts the overlap between the conceptual ground truth and the approach’s conceptual clusters. It is shown that in almost all datasets PoSSUM is the best approach, indicating that it forms the closest to the original conceptual clusters. The worst results are for the FACES dataset since it contains the highest conceptual diversity; nevertheless, it still achieves *RCC_F – score* = 0.69, 27% better than that of FACES. PubSum’s performance was close to PoSSUM’s, showing that when high conceptual diversity is involved, embedding models can prove superior to thesauri/ontologies used in FACES. Similar behavior of the results is depicted in all object-type datasets, since they contain the FACES one, with the embedding-based approaches

Table 5. Average RS_F-score and RCC_F-score

Dataset	RS_F-score						RCC_F-score		
	FACES		PubSum		PoSSUM		FACES	PubSum	PoSSUM
	k = 5	k = 10	k = 5	k = 10	k = 5	k = 10			
FACES	0.796	0.739	0.823	0.838	0.833	0.863	0.429	0.657	0.690
Healthcare									
Numerical Healthcare	N/A	N/A	0.740	0.533	0.900	0.533	0.500	0.444	0.833
Object-type Healthcare	N/A	N/A	0.932	0.939	0.923	0.942	0.434	0.805	0.803
All Healthcare (merged)	N/A	N/A	0.899	0.907	0.917	0.919	0.467	0.625	0.818
Smart Cities									
Numerical Smart Cities	N/A	N/A	0.949	0.949	0.949	0.956	0.333	0.439	0.715
Object-type Smart Cities	N/A	N/A	0.819	0.831	0.839	0.862	0.362	0.671	0.725
All Smart Cities (merged)	N/A	N/A	0.912	0.906	0.910	0.925	0.347	0.555	0.720

performing up to two times better than FACES. The biggest difference is depicted in the numerical datasets, with PoSSUM performing almost two times better than the other approaches for both use cases. Nevertheless, the numerical datasets have less conceptual diversity; therefore, even a single erroneous conceptual cluster could significantly affect the final RCC_F – score. The advantage of PoSSUM over the other approaches is also depicted in the merged datasets (All) with the Healthcare use case having the best results (RCC_F – score = 0.818) followed by Smart Cities (RCC_F – score = 0.720). The poor results of FACES regarding RCC_F – score but its good results for QUD prove that only evaluating a summary’s quality based on humans should not be the only criterion for the performance of all steps of an algorithm, especially when semantic and conceptual diversity are concerned.

RS_F – score depicts the loss of non-conceptually similar information in a summary due to top-k ranking. This metric is partly affected by the RCC_F – score as the better the latter is, the more diversity the final summary will have, and therefore the lower the loss of non-conceptually similar information. This is shown for PoSSUM, which has the best results for k = 10 and in most cases for k = 5. Nevertheless, RS_F – score is also partly affected by the triple ranking; therefore, PubSum may be slightly better than PoSSUM in the case of Numerical Smart Cities for top 5, affecting in this way the overall result (All Smart Cities (merged)), even though it has worse RCC_F – score than PoSSUM. Specifically, in this case, PubSum picked the most diverse triples even from erroneous conceptual clusters during triple ranking. The worst approach is FACES, supported also by the fact it had the worst RCC_F – score. In general, the lower the k, the lower the RS_F – score as the information filtering is stricter. The cases where this is not applied are when the number of conceptual clusters is less than k; therefore, more conceptual similarity will exist in the final summary (Numerical Healthcare).

7.4.3 System Performance. The system performance is shown in Tables 6 and 7. Table 6 presents the processing time of the conceptual clustering and the triple ranking. The window processing time is the overall time it takes from the first element that reaches the window to the window’s final summary creation. The window time contains the other times, and hence it is the longest. The times are shown for a small (windowSize = 50) to a large window (windowSize = 500) to observe how the size affects the performance. We see that the fastest approach is PoSSUM for all cases apart from the case of windowSize = 50 in Smart Cities, where PubSum is slightly overall faster. Nevertheless, PoSSUM has the fastest time in clustering and scoring on all occasions. This means that even though it contains two additional steps (aggregation and reasoning), the semi-incremental power of DBVARC clustering and the simplification of Triple2Rank have contributed

Table 6. Processing Time Performance (ms) for an Entity

Metric	FACES		PubSum		PoSSUM	
	Window Size 50	Window Size 500	Window Size 50	Window Size 500	Window Size 50	Window Size 500
Healthcare						
Window processing time	63,631	413,784	24,975	257,384	24,430	249,508
Clustering processing time	118.067	727.267	6.033	32.267	1.933	18.667
Scoring processing time	483.933	1,726.133	6.333	355.733	5.433	15.400
Smart Cities						
Window processing time	67,809	373,500	24,267	254,146	24,989	250,827
Clustering processing time	156.633	1,378.867	10.533	47.000	5.533	25.733
Scoring processing time	301.133	1,101.733	8.433	290.667	4.567	18.267

Table 7. Average Latency (ms) and Throughput (Events Per Second) for 10 Entities

Metric	No Top-k	FACES		PubSum		PoSSUM	
		Window Size 50	Window Size 500	Window Size 50	Window Size 500	Window Size 50	Window Size 500
Healthcare							
End-to-end latency	21,265	1,118,700	1,234,488	279,675	308,622	279,101	310,203
Throughput in	235.109	183.451	174.943	233.733	231.962	234.089	234.267
Throughput out		2.083	0.105	35.000	33.333	35.250	33.333
Smart Cities							
End-to-end latency	19,986	753,794	904,376	178,182	259,393	171,317	204,688
Throughput in	913.600	620.791	602.246	602.251	592.259	871.132	866.294
Throughput out		1.083	0.051	18.750	16.667	28.083	25.000

to better overall processing time. Specifically, PoSSUM takes half or even less time than PubSum in clustering, and it is slightly better for windowSize = 50, but significantly better for bigger windows than PubSum in scoring. Both approaches are significantly faster than FACES. In general, bigger windows perform more slowly since the window is waiting longer for its maximum size to be reached, and the triples are more in number when clustering and scoring occur. We also observe that usually clustering takes more time than scoring for PoSSUM, but the case is reversed for PubSum and FACES.

Similar behavior is depicted in Table 7. We see that the best latency and throughput are observed for No top-k since no additional analysis occurs, but when summarization happens, the results are mostly better for PoSSUM. PubSum is generally worse than PoSSUM, but in the case of Healthcare, it is not significantly different. The worst approach is FACES, with very poor results compared to the embedding-based approaches. In general, the bigger the window, the higher the latency and the lower the throughput. The throughput is analogous to the latency; therefore, the longer the events are processed, the fewer events the system can handle. This also explains the reason the throughput in is higher than the throughput out, since the system may integrate more and more events in parallel to the summarizations, but the events that are put out in the form of a summary are dependent on the time these summarizations take. The latency and throughput are also dependent on the parallelism, which is shown in the different results between the Healthcare and Smart Cities use cases. Smart Cities has lower latency and higher throughput than Healthcare since more publishers take place. We should also note that the end-to-end latency is higher than the window processing time since it takes into account the timestamp of the earliest published

event that is included in the summary. This means that the longer the events are generated, the higher the average end-to-end latency will be as the rate at which the events are generated is higher than the one at which they get consumed (throughput in is higher than throughput out).

PoSSUM behaves better than PubSum when the heterogeneity or conceptual diversity of the data is higher; hence, PubSum's performance is not significantly different than PoSSUM's in the case of Healthcare. Specifically, the Numerical Smart Cities dataset contains a higher volume of different attributes and heterogeneity than the Numerical Healthcare one. This results in a higher possibility of misclustering in the case of PubSum that, in return, leads to more processing time of clustering and scoring since more clusters that are also conceptually strict (i.e., higher FN) are created. Furthermore, PubSum is an extractive summarization approach that does not involve abstractive methodologies like incremental aggregation and reasoning of numerical data like PoSSUM. This means that PubSum will retain clusters that contain all of the measurements generated by the numerical sensors, even if they refer to the same measurement, whereas PoSSUM will incrementally aggregate the values referring to the same measurement and retain only one triple that contains its final reasoning result. This results in PubSum's scoring taking more time since it needs to examine all of the separate elements to observe which is the most representative from each cluster. This is more evident in the case of more heterogeneous datasets, like Smart Cities, especially for larger windows.

No top-k might have the best latency and throughput, but since no summarization takes place, all messages/events are sent to the subscriber. The summarization approaches contribute to the reduction of messages by eliminating conceptually similar data. The bigger the window and the lower the k, the higher the reduction of messages. For example, for `windowSize` = 50, there is 90% message reduction for `k` = 5 and 80% for `k` = 10. For `windowSize` = 500, the reduction is 99% and 98%, respectively. Therefore, we see the impact the filtering has in reducing significantly the final number of forwarded messages and instead sending only an important and representative subset. We should note though that the summarization approaches come with a longer memory footprint. For example, FACES uses the whole DBpedia for the triple ranking with 11.8GB of memory only for the relevant triples, whereas PubSum uses GoogleNews Word2Vec (<https://code.google.com/archive/p/word2vec/>) that holds 3.35GB. PoSSUM holds the lowest memory since it uses ConceptNet (<https://github.com/commonsense/conceptnet-numberbatch>), which has 1.09GB of memory for the English version.

8 CONCLUSION AND FUTURE WORK

In this article, we observe that IoT has contributed to the demand of systems that should deal with data integration, heterogeneity, redundancy, data enrichment, semantic abstractions, usability, and resource constraints. For this purpose, we proposed PoSSUM, a novel entity-centric PSS that provides entity summaries of data for user-friendly diversity-aware queries with the use of data integration, a novel DBVARC diverse entity summarization that is parameter-free and partly incremental, reasoning rules, and a novel Triple2Rank top-k filtering that is based on importance, informativeness, and diversity. We evaluated two use cases, Healthcare and Smart Cities, with a novel evaluation methodology that created a set of ground truths and metrics that capture the quality of entity summaries. According to the evaluation results, we showed that although there is a minor discrepancy of which information is important among humans, the less information is provided, the higher the diversity humans need that could reach up to 80%. Regarding the approaches, the typical No top-k PSS has the best latency and throughput, but it creates an abundance of notifications that overwhelm the system and the subscriber. On the other hand, the summarization approaches achieve from 80% to 99% message reduction, with PoSSUM achieving the best overall results in terms of correctness and system performance. Specifically, it achieves the best

ranking quality for 17 (top 5) to 13 (top 10) out of 30 entities by a significant margin over the second-best approach, and it has the highest conceptual clustering F-score ranging from 0.69 to 0.83 and a redundancy-aware F-score up to 0.95 with cases, where it is almost two times better than the other approaches. Also, compared to the second-best approach, PoSSUM takes 50% or less clustering processing time and it performs scoring significantly faster for larger windows. It also has comparable latency and throughput values, and it occupies a third of the memory. The results prove the superiority of embedding models compared to rigid and domain-dependent thesauri/ontologies when high data conceptual diversity is concerned, and that the incremental phase of DBVARC and the simplified but effective Triple2Rank can significantly boost the quality of the summaries and the processing performance of the system. PoSSUM is parameter-free; therefore, no prior parameter tuning is needed, in contrast to the other approaches. However, it may be too strict for sparse elements as it focuses on strict densely connected regions. This means that it is affected by the curse of dimensionality; that is, if the dimensionality increases but the volume of the elements stays the same, then there may be no points that overlap within a region to form a conceptual cluster.

Future work will focus on non-manual reasoning methodologies that are not only effective but also efficient for a complex real-time environment like IoT. Also, an incremental triple scoring approach will be examined for even better efficiency as well as an automatic typing information extraction of object-type properties. An interesting application, for the future, is to apply PoSSUM on multimedia data for the creation of image and video summaries.

ACKNOWLEDGMENTS

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] Charu C. Aggarwal. 2007. *Data Streams: Models and Algorithms*, Vol. 31. Springer Science & Business Media.
- [2] Charu C. Aggarwal and S. Yu Philip. 2007. A survey of synopsis construction in data streams. In *Data Streams*. Springer, 169–207.
- [3] Charu C. Aggarwal, S. Yu Philip, Jiawei Han, and Jianyong Wang. 2003. A framework for clustering evolving data streams. In *Proceedings of the 29th International Conference on Very Large Data Bases (VLDB'03)*. Morgan Kaufmann, 81–92.
- [4] Noura Alhakbani, Mohammad Mehedi Hassan, Mourad Ykhlef, and Giancarlo Fortino. 2019. An efficient event matching system for semantic smart data in the internet of things (IoT) environment. *Future Generation Computer Systems* 95 (2019), 163–174.
- [5] Payam Barnaghi, Wei Wang, Cory Henson, and Kerry Taylor. 2012. Semantics for the Internet of Things: Early progress and back to the future. *International Journal on Semantic Web and Information Systems (IJSWIS)* 8, 1 (2012), 1–21.
- [6] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research* 3 (Jan. 2003), 993–1022.
- [7] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Neural Information Processing Systems (NIPS'13)*. 1–9.
- [8] César Cañas, Eduardo Pacheco, Bettina Kemme, Jörg Kienzle, and Hans-Arno Jacobsen. 2015. Graps: A graph publish/subscribe middleware. In *Proceedings of the 16th Annual Middleware Conference*. 1–12.
- [9] Feng Cao, Martin Ester, Weining Qian, and Aoying Zhou. 2006. Density-based clustering over an evolving data stream with noise. In *Proceedings of the 2006 SIAM International Conference on Data Mining*. SIAM, 328–339.
- [10] Matthias Carnein, Dennis Assenmacher, and Heike Trautmann. 2017. An empirical comparison of stream clustering algorithms. In *Proceedings of the Computing Frontiers Conference*. 361–366.
- [11] Šejla Čebirić, François Goasdoué, Haridimos Kondylakis, Dimitris Kotzinos, Ioana Manolescu, Georgia Troullinou, and Mussab Zneika. 2019. Summarizing semantic graphs: A survey. *The International Journal on Very Large Data Bases (VLDB Journal)* 28, 3 (2019), 295–327.
- [12] Rustem Dautov, Salvatore Distefano, and Rajkumar Buyya. 2019. Hierarchical data fusion for smart healthcare. *Journal of Big Data* 6, 1 (2019), 19.

- [13] Christoph Doblander, Tanuj Ghinaiya, Kaiwen Zhang, and Hans-Arno Jacobsen. 2016. Shared dictionary compression in publish/subscribe systems. In *Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems*. 117–124.
- [14] Margaret H. Dunham. 2006. *Data Mining: Introductory and Advanced Topics*. Pearson Education India.
- [15] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, Vol. 96. 226–231.
- [16] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. 2003. The many faces of publish/subscribe. *ACM Computing Surveys (CSUR)* 35, 2 (2003), 114–131.
- [17] Douglas H. Fisher. 1987. Knowledge acquisition via incremental conceptual clustering. *Machine Learning* 2, 2 (1987), 139–172.
- [18] Kalpa Gunaratna, Krishnaprasad Thirunarayan, and Amit Sheth. 2015. FACES: Diversity-aware entity summarization using incremental hierarchical conceptual clustering. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*.
- [19] Kalpa Gunaratna, Krishnaprasad Thirunarayan, Amit Sheth, and Gong Cheng. 2016. Gleaning types for literals in RDF triples with application to entity summarization. In *International Semantic Web Conference*. Springer, 85–100.
- [20] Michael Hahsler and Matthew Bolaños. 2016. Clustering data streams based on shared density between micro-clusters. *IEEE Transactions on Knowledge and Data Engineering* 28, 6 (2016), 1449–1461.
- [21] Stephen Jose Hanson and Malcolm Bauer. 1989. Conceptual clustering, categorization, and polymorphy. *Machine Learning* 3, 4 (1989), 343–372.
- [22] Souleiman Hasan and Edward Curry. 2014. Approximate semantic matching of events for the internet of things. *ACM Transactions on Internet Technology (TOIT)* 14, 1 (2014), 1–23.
- [23] Arijit Khan, Sourav S. Bhowmick, and Francesco Bonchi. 2017. Summarizing static and dynamic big graphs. *Proceedings of the Very Large Data Bases Endowment* 10, 12 (2017), 1981–1984.
- [24] Şefki Kolozali, Maria Bermudez-Edo, Nazli FarajiDavar, Payam Barnaghi, Feng Gao, Muhammad Intizar Ali, Alessandra Mileo, Marten Fischer, Thorben Iggena, Daniel Kuemper, and Ralf Tonjes. 2018. Observing the pulse of a city: A smart city framework for real-time discovery, federation, and aggregation of data streams. *IEEE Internet of Things Journal* 6, 2 (2018), 2651–2668.
- [25] Haridimos Kondylakis, Dimitris Kotzinos, and Ioana Manolescu. 2019. RDF graph summarization: principles, techniques and applications. In *Proceedings of the 22nd International Conference on Extending Database Technology*. 433–436.
- [26] Philipp Kranen, Ira Assent, Corinna Baldauf, and Thomas Seidl. 2011. The ClusTree: Indexing micro-clusters for any-time stream mining. *Knowledge and Information Systems* 29, 2 (2011), 249–272.
- [27] Haifeng Liu, Zhaohui Wu, Milenko Petrovic, and Hans-Arno Jacobsen. 2011. Optimized cluster-based filtering algorithm for graph metadata. *Information Sciences* 181, 24 (2011), 5468–5484.
- [28] Qingxia Liu, Gong Cheng, Kalpa Gunaratna, and Yuzhong Qu. 2021. Entity summarization: State of the art and future challenges. *Journal of Web Semantics* (2021), 100647.
- [29] Qingxia Liu, Gong Cheng, and Yuzhong Qu. 2020. Deeplens: Deep learning for entity summarization. In *Proceedings of the Workshop on Deep Learning for Knowledge Graphs*.
- [30] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*. 3111–3119.
- [31] George A. Miller. 1956. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review* 63, 2 (1956), 81.
- [32] Mansoor Nasir, Khan Muhammad, Jaime Lloret, Arun Kumar Sangaiah, and Muhammad Sajjad. 2019. Fog computing enabled cost-effective distributed summarization of surveillance videos for smart cities. *Journal of Parallel and Distributed Computing* 126 (2019), 161–170.
- [33] Shobharani Pacha, Suresh Ramalingam Murugan, and R. Sethukarasi. 2020. Semantic annotation of summarized sensor data stream for effective query processing. *Journal of Supercomputing* 76, 6 (2020), 4017–4039.
- [34] Santosh Pattar, Rajkumar Buyya, Kuppanna Rajuk Venugopal, S. S. Iyengar, and L. M. Patnaik. 2018. Searching for the IoT resources: Fundamentals, requirements, comprehensive review, and future directions. *IEEE Communications Surveys & Tutorials* 20, 3 (2018), 2101–2132.
- [35] Niki Pavlopoulou and Edward Curry. 2019. Using embeddings for dynamic diverse summarisation in heterogeneous graph streams. In *Proceedings of the 1st International Conference on Graph Computing (GC’19)*. IEEE, 5–12.
- [36] Niki Pavlopoulou and Edward Curry. 2020. Dynamic diverse summarisation in heterogeneous graph streams: A comparison between thesaurus/ontology-based and embeddings-based approaches. *International Journal of Graph Computing*, 1, 1 (2020), 23–39.

- [37] Niki Pavlopoulou and Edward Curry. 2022. IoTSAx: A dynamic abstractive entity summarisation approach with approximation and embedding-based reasoning rules in publish/subscribe systems. *IEEE Internet of Things Journal* 9, 3 (2022), 1830–1847.
- [38] Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. 2013. Context aware computing for the internet of things: A survey. *IEEE Communications Surveys & Tutorials* 16, 1 (2013), 414–454.
- [39] Aírel Pérez-Suárez, José F. Martínez-Trinidad, and Jesús A. Carrasco-Ochoa. 2019. A review of conceptual clustering algorithms. *Artificial Intelligence Review* 52, 2 (2019), 1267–1296.
- [40] Milenko Petrovic, Haifeng Liu, and Hans-Arno Jacobsen. 2005. G-ToPSS: Fast filtering of graph-based metadata. In *Proceedings of the 14th international conference on World Wide Web*. 539–547.
- [41] Seyedamin Pouriyeh, Mehdi Allahyari, Krys Kochut, Gong Cheng, and Hamid Reza Arabnia. 2018. Combining word embedding and knowledge-based topic modeling for entity summarization. In *Proceedings of the 12th International Conference on Semantic Computing (ICSC’18)*. IEEE, 252–255.
- [42] Daniel Philip Puschmann. 2018. *Extracting Information from Heterogeneous Internet of Things Data Streams*. Ph.D. Dissertation. University of Surrey.
- [43] Yongrui Qin, Quan Z. Sheng, Nickolas J. G. Falkner, Schahram Dustdar, Hua Wang, and Athanasios V. Vasilakos. 2016. When things matter: A survey on data-centric internet of things. *Journal of Network and Computer Applications* 64 (2016), 137–153.
- [44] Petar Ristoski and Heiko Paulheim. 2016. Rdf2vec: RDF graph embeddings for data mining. In *International Semantic Web Conference*. Springer, 498–514.
- [45] Antonio F. Skarmeta, José Santa, Juan A. Martínez, Josiane X. Parreira, Payam Barnaghi, Shirin Enshaeifar, Michail J. Beliatas, Mirko A. Presser, Thorben Iggena, Marten Fischer, Ralf Tönjes, Martin Strohbach, Alessandro Sforzin, and Hien Truong. 2018. IoTcrawler: Browsing the internet of things. In *2018 Global Internet of Things Summit (GloTS’18)*. IEEE, 1–6.
- [46] Marcin Sydow, Mariusz Pikula, and Ralf Schenkel. 2010. DIVERSUM: Towards diversified summarisation of entities in knowledge graphs. In *Proceedings of the 26th International Conference on Data Engineering Workshops (ICDEW’10)*. IEEE, 221–226.
- [47] Ioan Szilagyí and Patrice Wira. 2016. Ontologies and semantic web for the Internet of Things-A survey. In *Proceedings of the 42nd Annual Conference of the IEEE Industrial Electronics Society (IECON’16)*. IEEE, 6949–6954.
- [48] James W. Tanaka and Marjorie Taylor. 1991. Object categories and expertise: Is the basic level in the eye of the beholder? *Cognitive Psychology* 23, 3 (1991), 457–482.
- [49] Andreas Thalhammer. 2017. *Linked Data Entity Summarization*. Ph.D. Dissertation. Karlsruhe Institute of Technology.
- [50] Peter Triantafyllou and Andreas Economides. 2004. Subscription summarization: A new paradigm for efficient publish/subscribe systems. In *Proceedings of the 24th International Conference on Distributed Computing Systems, 2004*. IEEE, 562–571.
- [51] Dongjun Wei, Yaxin Liu, Fuqing Zhu, Liangjun Zang, Wei Zhou, Jizhong Han, and Songlin Hu. 2019. ESA: entity summarization with attention. In *Proceedings of the 2nd International Workshop on Entity Retrieval (EYRE’19)*. ACM.
- [52] Yi Zhang, Jamie Callan, and Thomas Minka. 2002. Novelty and redundancy detection in adaptive filtering. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 81–88.

Received June 2021; revised October 2021; accepted December 2021