Enabling Context-Aware Search using Extracted Insights from IoT Data Streams

V. Janeiko[1], R. Rezvani[1], N. Pourshahrokhi[1], S. Enshaeifar[1],
M. Krogbæk[2], S. H. Christophersen[2], T. Elsaleh[1], and P. Barnaghi[1]

[1]CVSSP, Department of Electrical and Electronic Engineering, University of Surrey, UK
{*v.janeiko, r.rezvani, n.pourshahrokhi, s.enshaeifar, t.elsaleh, p.barnaghi*}*@surrey.ac.uk*
[2]ITK Lab, Aarhus Municipality, Aarhus, Denmark
{*mkrog, sech*}*@aarhus.dk*

***Abstract*—The rapid growth in collecting and sharing sensory observation form the urban environments provides opportunities to plan and manage the services in the cities better and allows citizens to also observe and understand the changes in their surrounding in a better way. The new urban sensory data also creates opportunities for further application and service development by creative industries and start-ups. However, as the size and diversity of this data increase, finding and accessing the right set of data in a timely manner is becoming more challenging. This paper describes a search engine designed for indexing, searching and accessing urban sensory data. We present the key feature and architecture of the system and demonstrate some of the functionalities that are provided by searching for raw sensory observations and also pattern search functions that are enabled by a pattern analysis algorithm, supported by monitoring of data streams for changes in quality of information and remediation.**

***Index Terms*—Time-series data, pattern analysis, Internet of Things, information search and retrieval, stream data monitoring, missing data.**

I. Introduction

The Internet of Thing (IoT) technologies provide opportunities to collect, publish and share sensory observations and measurement data from the real-world environment. Over the past decade, there has been a rapid increase in deploying IoT infrastructure and sensors in urban environments to report information in various areas such as traffic, air quality, weather, public transport and waste collection. This information is often published on data sharing portals or offered via APIs in open or authenticated forms. Similarly, IoT technologies are also increasingly being used in industrial environments for control and management, productivity, safety and other purposes [1]. With a rapid increase in the observation and measurement data emerging from this IoT infrastructure in open networks such as open urban data portals or closed and managed intranet of sensors, finding and accessing the data is becoming a key challenge. Most of the current IoT systems, rely on meta-data descriptions for searching and discovering data sources and have limited means to search for the patterns and change within the IoT data streams [2]. In the open internet world, conventional search engines are designed to process hyperlinks and multimedia data, but they are not very effective in searching for sensory observations and measurement data [3]. While there has been a lot of work on developing interoperable models for the IoT systems [4], [5], [6] there have been relatively few works on developing indexing, search and discovery solutions for IoT resources and their streaming data.

978-1-7281-6728-2/20/$31.00© 2020 IEEE

One of the motivations of our work is to build a platform that can include IoT sources from different systems and sources and can provide indexing, search and discovery across multiple platforms and systems. Another aim is to provide content analysis for the data sources and enable the search and discovery of information based on historical data and patterns.

Providing an effective search and discovery for an IoT system requires indexing and finding relevant resources based on temporal, spatial and thematic data [7]. It also involves an analysis of the streaming data to find patterns and changes in the raw data. A key challenge associated with extracting insights is in the processing and interpreting of IoT stream data with respect to dynamicity and quality variations. Time-series data collected from sensory observations often have missing values or the observations are sometimes incomplete. In this paper, we discuss a search and discovery method for streaming IoT data and demonstrate how the proposed method has been in an end-to-end system to enable search and discovery of raw data and automatically discovered and labelled patterns in an urban open data ecosystem. The proposed system is described in the context of an open data platform. However, the proposed solutions can be adopted and applied to other use-cases and intranet systems in Industry 4.0, healthcare IoT data and other application areas.

II. Related Work

If one uses a conventional search engine such as Google to search for example ”traffic in Aarhus”, the search engine returns more than 2.5 million entries. In a sense, the conventional search engines often act as an information locator rather than providing the exact information requested. In the case of IoT and sensory measurements and ad-hoc observations, it becomes even more apparent that the conventional hypertext/-multimedia search engines are not equipped for this new form

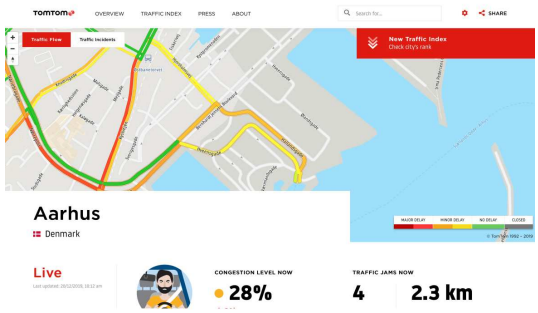


Fig. 1: Traffic data from the City of Aarhus

of ubiquitous and dynamic information search and retrieval. In the case of public data, however, various sources provide some information search and visualisation. For example, traffic data for the City of Aarhus can be seen on various platforms such as TomTom[1], as shown in Figure 1. However, these online platforms are often proprietary and do not offer pattern or co-relation analysis and do not include data from other sources.

Other search engines are also specifically designed for IoT data. For example, Shodan[2] provides search functionality for internet-connected devices using internet protocols. Wolfram Data Drop[3] offers open services to add semantics to the data and process collected time-series data. Data Drop offers powerful tools to build and program a system. However, it is based on a proprietary engine and requires programming and development for adding and processing the data sources. Thingful.net[4] provides search functions for manually included data sources. However, these existing IoT search systems do not provide automated and scalable indexing of the time-series data. They also do not offer scalable solutions to create IoT search and discovery functions across different platforms.

Data representation is an important process for extracting insights from data streams. They can be used in the indexing process of a search engine by reducing the dimensions of the data. These can also be used in pattern analysis and higher-level analysis as we detect patterns of data and by knowing the domain it can be used to construct contextual topics and events. There exist various methods such as Discrete Fourier Transform (DFT) [8], Discrete Wavelet Transform (DWT) [9], Piece-wise Aggregate Approximation (PAA) [10], Singular Value Decomposition (SVD) [11], Adaptive Piece-wise Constant Approximation (APCA) [12], and Symbolic Aggregate approXimation (SAX) [10]. Although, these methods have some disadvantages such as; in the case of SAX, a Gaussian process is presumed, whereas DFT and DWT are computationally resource intensive. To counter these drawbacks, we have employed a time-series data segmentation and representation method based on Lagrange multipliers which creates a flexible and scalable method that is also suitable for dynamic data [13].

Regarding handling missing data, various statistical and probabilistic methods are used to impute missing values in continuous data such as KNN and missForest. In K-nearest neighbour (KNN) [14], the missing data values replaced by borrowing values from the nearest completely observed neighbour, which has more similarity to the current state (the donor). The method relies on lots of assumptions, including the choice of the distance metric, the definition of the donor pool, and how to sample from the donor pool. This method benefits from being practically useful for dealing with different types of data such as continuous, discrete, ordinal and categorical. On the other hand, the assumption of approximating a missing value based on neighbours may lead to lack of confidence in results.

MissForest [15] is an iterative imputation method based on the random forest which averages over many classification and regression trees to establish a natural multiple imputation. It is equipped with built-in error estimates of random forest. Therefore, without the need for a test set, imputation error can be calculated [16].

The existing approaches focus on statistical features of the data such as mean, median and standard deviation or use probabilistic measures such as maximum likelihood values. However, these models are not very effective to utilise the inherent correlations across multiple features (i.e. dimensions) of data streams to impute the missing values. Markov Chain Monte Carlo (MCMC) [17] models provide a more effective approach to process the cross-dimensional relations and use random walk and sampling methods to impute the missing values in large-scale data streams. Using MCMC, a model can create multiple samples that can be used for training more generalisable learning models.

III. System Architecture

The system is designed in a way that enables urban-scale data integration and processing and can be easily scaled to support larger datasets. The architecture diagram is shown in Figure 2. All communication between system entities is done via HTTP/S. The data is exported from a CKAN[5] Open Data portal and then semantically annotated and represented as NGSI-LD entities based on the IoT-Stream [18] and the SOSA (Sensor, Observation, Sample, and Actuator) [4] ontologies. The semantically annotated NGSI-LD entities are then published to an NGSI-LD broker, which is shown in Figure 2. The pattern extractor and indexing components can access the data and its associated metadata using this broker component. The pattern extractor analyses the data and generates higher-level events that are published back to the broker.

When a user makes a query, it is directed to the indexing component. If a query is a simple metadata query (i.e. searching for a sensor or a data source), it is answered directly using

[1]https://www.tomtom.com/en_gb/traffic-index/aarhus-traffic
[2]https://www.shodan.io
[3]https://datadrop.wolframcloud.com
[4]http://www.thingful.net
[5]https://ckan.org

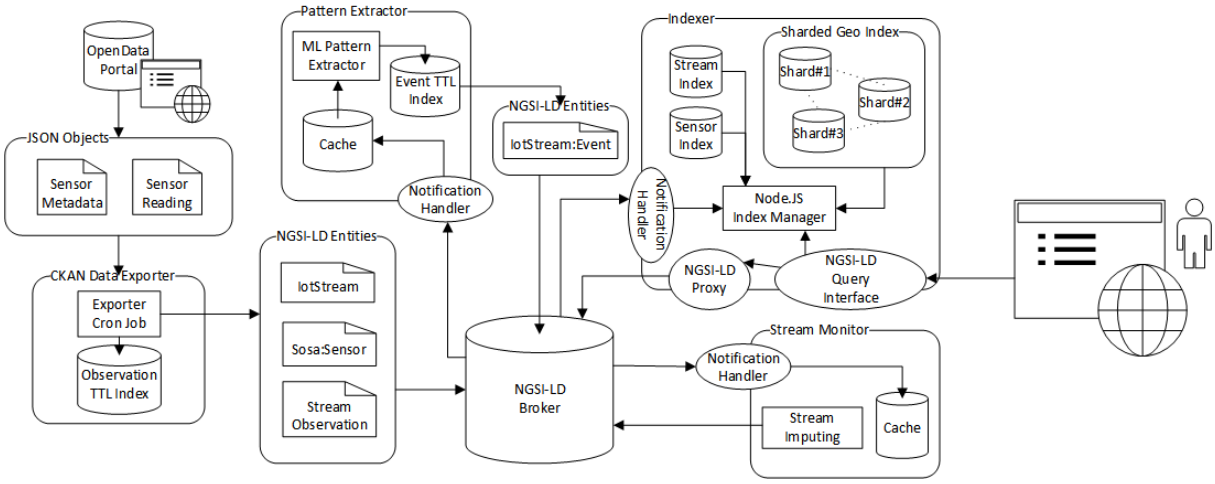


Fig. 2: The Architecture of the System

the indexed metadata. Otherwise, the indexing component acts as a proxy and forwards the request to the NGSI-LD broker.

The scalability is achieved by the fact that all components can have multiple distributed instances, with each one only responsible for a portion of the data. NGSI-LD broker can be deployed in a federated fashion, where the top-level broker only acts as a gateway for lower level (domain) brokers. Multiple instances of the data exporter and pattern extractor components can be deployed and configured to only process a portion of the data. Metadata in the index is partitioned into clusters based on the physical location of the sensor. These processes are facilitated by a linked-data model used throughout the system.

A. Data Exporter

To support semantic interoperability and to integrate data from various sources across multiple systems and different platforms, we use W3C/OGC SOSA[6] and IoTStream[7] ontologies. To accommodate other data models, we use the data exporter component to publish the data according to the common model used in our system.

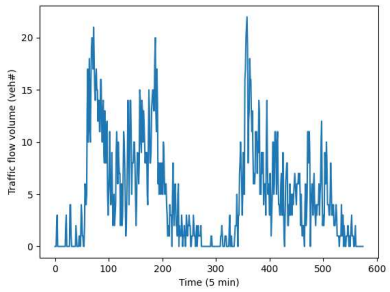
The data exporter component periodically sends a request to CKAN DataStore API to obtain the latest data. If a new sensor is detected, new *sosa:sensor* and IoTStream entities are created in the NGSI-LD broker. Sensory observations and measurements are stored as StreamObservation entities in the broker. In addition to publishing the data into the broker, the observation ids are also stored in a Time-To-Live (TTL) index. This index is then used to delete the observations from the broker that are older than a pre-defined threshold. Deleting obsolete raw data from the system allows us to avoid storage problems and keeps the systems' performance at an adequate level.

There are no uniform data model requirements in the CKAN portal. However, a separate data exporter needs to be implemented for each dataset if they are represented based on a different model. Metadata properties of these various datasets can be then mapped to *sosa:sensor* and IoTStream properties in the broker. There are only two required properties: *sosa:observes* and *sosa:location*. The first property is used to determine the type of the sensor. The latter is used in responding to geospatial queries. The location data is represented as a *GeoJSON*[8] object. In this demonstrator, the location of each sensor is represented as a *LineString* object, where the start and endpoint are the points of the road segment that a sensor is deployed. The observation and measurement data is mapped to *StreamObservation* entity, using *resultTime* and *hasSimpleResult* within the IoTStream ontology. These properties are used to represent the observation time and observed value. These three properties are used by the pattern extractor component to construct the observation windows that are used in the data analysis and pattern representations.

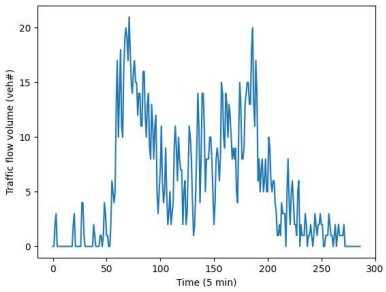
B. Stream Monitor

The stream monitor observes data streams with high-quality metrics, such as consistency in frequency, then based on a Kernel density function it fits a model to data to estimate the probability density of data. Markov chain Monte Carlo (MCMC) methods are designed to generate samples from probability distribution by construction a chain of states which are reasonably far from each other and logically has the target distribution as its stationary distribution. It works by tracking the states of the chain after the initial phase (burn-in time). These chains are stochastic walkers moving around randomly looking for states with high contribution to the integral to move into the next state. The more the algorithm has time to explore the distribution, the more accurate and closer it gets to target distribution [19]. With the help of the probability density, a Monte Carlo Markov Chain (MCMC) after converging to the target distribution, is capable of generating samples for time slots that fit data. The generated samples are used to impute

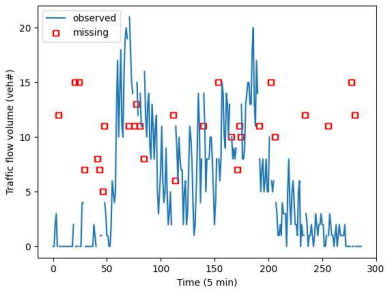
[6]https://www.w3.org/TR/vocab-ssn/
[7]http://purl.org/iot/ontology/iot-stream
[8]https://geojson.org



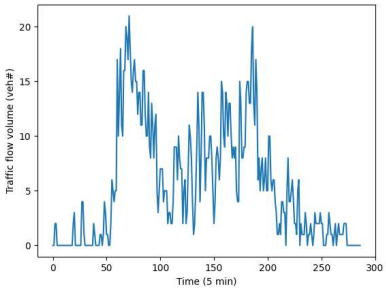
(a) City traffic flow in 2 days captured every 5 minutes



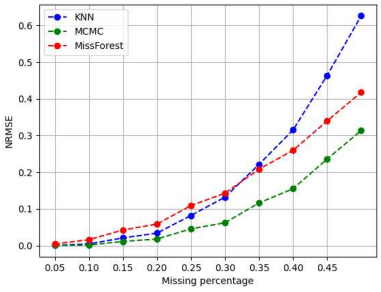
(b) Complete data captured for one day



(c) Missing values detected in the data stream



(d) Imputed result using the algorithm



(e) Comparison between the performance of MCMC and traditional methods for data imputation

Fig. 3: Monitoring and Imputation for Traffic Data Streams

missing values by marginalisation over samples for missing values. The algorithm is run on traffic data during observation of every 5 minutes traffic sensor from the city of Aarhus. Figure 3a shows traffic flow based on vehicle number captured every 5 minutes during dates 4/12/2019 to 5/12/2019.

Figure 3b shows completed data captured for one day and Figure 3c shows the missing values detected in the data stream and finally Figure 3d shows imputed result using the algorithm.

Figure 3e shows a comparison between the performance of MCMC and traditional methods for data imputation. The result is compared with well-known methods in the literature, and we repeated the experiments with different missing percentage rate to assess the robustness of results. All the experiments are run five times to make sure that results are reliable and reproducible. NRMSE (Normalised root mean square) is reported as a performance metric along with standard deviation across five experiments. NRMSE in Equation 1 is defined as:

$$\sqrt{mean((X_{true} - X_{imp})^2)/var(X_{true})} \tag{1}$$

where X_{true} the complete data matrix, X_{imp} the imputed data matrix and $mean/var$ being used as a short notation for the empirical mean and variance computed over the continuous missing values only.

C. Pattern Extractor

The pattern extractor component uses the StreamObservations and to detect higher-level events, which are then published to the broker. A machine learning method is used to analyse a group of *sosa:observes* properties. This analysis model is registered as a new subscription in the broker. This subscription includes spatial and temporal and the data type specifications for data streams. Once a stream matches the specifications of a registered process, the pattern extractor then caches the observations that are necessary to perform the analysis (in the demo this was set to 1 hour).

When the observations are received for each stream in the group, the pattern extractor constructs a 1-hour window to analyse the data. The result of the analysis is a label for a pattern of data. This label, together with the start and end times of the pattern, is then represented as an *IoTStream:Event* entity and published into the broker. The broker then makes these patterns search-able.

1) The Pattern Extraction Algorithm: One of the key issues in extracting insights from IoT data is the large volumes of data collected in (near) real-time. The velocity and volume of the data make the analysis complex and demanding. It is also hard for users to explore a large amount of raw data. Pattern extraction techniques are beneficial in obtaining higher-level information from data. The patterns help users to search for

higher-level concepts and insights rather than browsing and interpreting raw data. To extract patterns from the data in this framework, we first apply Piecewise Aggregate Approximation (PAA) [12] to reduce the quantity of data and then apply the Lagrangian Multiplier to normalise the data [13]. Sensor data are multivariate time-series data. Using the Lagrangian Multiplier, we scale each dimension between [-1, 1]. We then perform a clustering method to group the data into clusters. The label and size of the clusters are pre-defined according to the data types. For example, for the traffic density, we assume low, medium, high and critical which corresponding to 4 clusters. Also, for traffic flow, there are 4 clusters; congestion, low, medium and high traffic. After the clustering process, we perform a statistical analysis to measure the closeness of the data to each of the labels. The advantage of this method is that rather than relying on single or multiple observations, we use pattern identification and clustering methods to analyse the trend of the data and changes in these trends.

D. Query Interface

NGSI-LD is used in the system to support interoperability and also allow querying the data using a unified format. NGSI-LD is a well-specified format [20] and offers powerful linked-data querying capabilities. The NGSI-LD broker[9] offers basic querying capabilities. However, it is designed with a very generic data model in mind. To enhance the query processing performance, we use an indexing component.

When a user submits a query, it is translated to an *IoT-Streams* or sosa:Sensors property query. The queries are then directed to the indexing component. If the query only contains entities and properties that are indexed (the ones mentioned in previous sections), it is responded to directly. Otherwise, the query is forwarded to the broker for an overall linked-data query. This architecture significantly enhances the performance of the query. For example, similar locations, data types and temporal data can be indexed and grouped in virtual segments using the indexing component, and this will significantly reduce the search scope in query processing for large-scale data collections.

1) Indexing Process: The indexing component includes an NGSI-LD query interface and uses the data from the broker to respond to common queries. This component creates a subscription for metadata changes in the broker to keep the indexes up-to-date. Each time the metadata changes, the indexing component receives a notification and updates the index appropriately.

The index consists of three parts: two indices that map entity ids of *IoTStream* and *sosa:Sensor* to geo-partition key, and a geo-partitioned index containing the data. A partition key is computed according to the location of the data. The partition key is computed by intersecting sensor location GeoJSON objects with predefined region GeoJSON polygons. Entities in the index are stored as a graph. Instead of storing all entities separately, linked entities are stored as a single document.

[9]https://github.com/FIWARE/context.Orion-LD

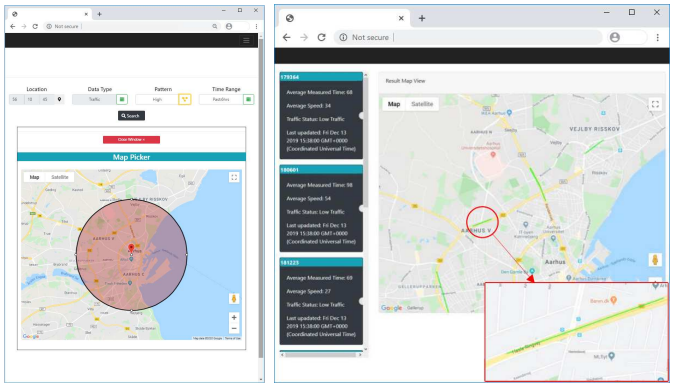


Fig. 4: An overview of the user interface and query results in searching for patterns

The top-level entity is *IoTStream* and all related entities are stored as nested documents in the properties of the parent document. This structure allows to define compound indices, thus accelerates nested queries.

2) Subscriptions: The NGSI-LD broker also provides a capability to use repetitive queries to create subscriptions. The detected patterns for the data streams are published to the broker as event entities by the pattern extractor component. These patterns can be queried, and more importantly, the users can subscribe to the events. For example, a user can subscribe for traffic congestion events on a route that she is taking, and be notified about emerging events in (near) real-time.

IV. A Prototype System

The system can be used in various domains. In this paper, we demonstrate a use-case in urban information search and retrieval. The implemented use-case demonstrates how higher-level abstractions (e.g. traffic patterns) can be used by citizens and urban planners to make informed decisions. Such a system makes the data more accessible to users and allows search for patterns and trends rather than exploring large volumes of data. For examples, departments in the city responsible for traffic management can use the tool to identify congested roads and correlations between different traffic patterns in different parts of the city.

The user interface is shown in Figure 4. On the left, users can select a desired search region on the map by dragging a red circle to an area of interest. Users can also adjust the radius of the search region by dragging one of the markers on the edge of the circle. Then users can select the data type or patterns for a search query. Users can also select a time range to search.

Users are presented with a visual representation of results on the map and a list of discovered patterns (the figure on the right). On the map, each road segment with the requested pattern is shown with a different colour according to the detected pattern. The pattern results on the left provide more information about the detected pattern and their corresponding road segment; such as the segment identifier, average speed of the cars and time it took to drive through that segment within

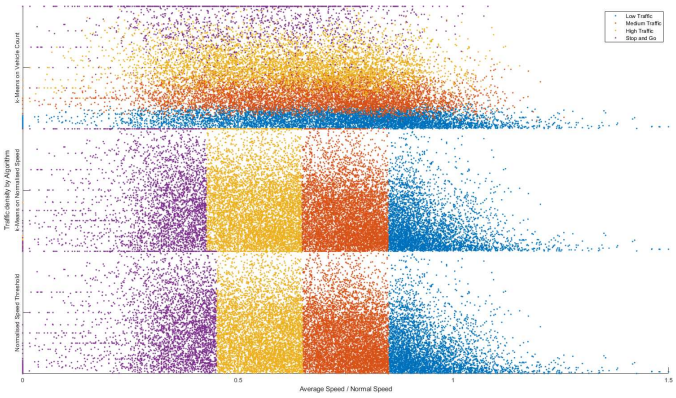


Fig. 5: Results of proposed traffic (density and flow) classification algorithms compared with traditional thresholds based approach

the searched time window. Each information card is clickable to zoom-in and highlights the corresponding road segment on the map.

For clustering, we used k-means clustering method. The number of cars on the road has been used for extracting traffic density patterns. To evaluate the performance of the clustering method, the Silhouette coefficient has been used. The Silhouette coefficient calculates how separate each cluster is from the other clusters, and its value is around 0.7 for our traffic density analysis. For traffic flow patterns, we used speed values divided by the average speed measured by each sensor as an input for the pattern extractor. The clustering result is shown in Figure 5.

V. Conclusions

This paper presented a search engine for streaming IoT data. We discuss the key components of the proposed system and show how, with the use of a common NGSI-LD model description and representation model we have provided interoperability across different systems and platforms. The system allows searching for data sources (i.e. sensory devices) based on spatial, temporal and thematic features. It also provides a search function based on higher-level concepts associated with patterns (e.g. traffic congestion patterns in a particular street). Data segmentation and normalisation is used to create the patterns, and a clustering method is used to group the patterns. The patterns are then labelled by analysing the statistical properties of the patterns in each group and using advance knowledge about the data streams (e.g. the traffic data is categorised as high, congested, moving and low traffic). A Stream monitoring component has been developed to compensate for missing data using imputation, which is an expected case for real-world scenarios.

The work presents the building blocks of a scalable and distributed data stream search and pattern discovery for IoT data. It is demonstrated in the context of an urban data stream search. We also briefly discuss the indexing, query and pub/sub mechanisms that are developed in the system. The future work will focus on providing crawling and distributed orchestration mechanisms to test and evaluate the scalability of the proposed solutions for large-scale IoT environments.

Acknowledgement

This work has been sponsored by the EU Horizon 2020 Research and Innovation programme, through the IoTCrawler project (contract no. 779852).

References

[1] D. Puiu, P. M. Barnaghi *et al.*, "Citypulse: Large scale data analytics framework for smart cities," *IEEE Access*, vol. 4, pp. 1086–1108, 2016.

[2] S. Trilles, Ó. B. Fernández *et al.*, "A domain-independent methodology to analyze iot data streams in real-time. a proof of concept implementation for anomaly detection from environmental data," *International Journal of Digital Earth*, vol. 10, pp. 103 – 120, 2017.

[3] P. Barnaghi and A. Sheth, "On searching the internet of things: Requirements and challenges," *IEEE Intelligent Systems*, vol. 31, no. 6, pp. 71–75, Nov. 2016.

[4] K. Janowicz, A. Haller *et al.*, "Sosa: A lightweight ontology for sensors, observations, samples, and actuators," *Journal of Web Semantics*, 2018.

[5] M. Bermudez-Edo, T. Elsaleh *et al.*, "Iot-lite: a lightweight semantic model for the internet of things and its use with dynamic semantics," *Personal and Ubiquitous Computing*, vol. 21, no. 3, pp. 475–487, 2017.

[6] R. Agarwal, D. G. Fernandez *et al.*, "Unified iot ontology to enable interoperability and federation of testbeds," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, Dec 2016, pp. 70–75.

[7] A. Sheth, C. Henson, and S. S. Sahoo, "Semantic sensor web," *IEEE Internet Computing*, vol. 12, no. 4, pp. 78–83, July 2008.

[8] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast subsequence matching in time-series databases," in *SIGMOD '94*, 1994.

[9] K. pong Chan and A. W.-C. Fu, "Efficient time series matching by wavelets," *Proceedings 15th International Conference on Data Engineering (Cat. No.99CB36337)*, pp. 126–133, 1999.

[10] J. Lin, E. J. Keogh *et al.*, "A symbolic representation of time series, with implications for streaming algorithms," in *DMKD '03*, 2003.

[11] D. Wu, D. Agrawal *et al.*, "Efficient retrieval for browsing large image databases," in *CIKM '96*, 1996.

[12] E. Keogh, K. Chakrabarti *et al.*, "Dimensionality reduction for fast similarity search in large time-series databases," *Knowledge and information Systems*, pp. 263–286, 2001.

[13] R. Rezvani, S. Enshaeifar, and P. Barnaghi, "Lagrangian-based pattern extraction for edge computing in the internet of things," in *2019 6th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2019 5th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*. IEEE, 2019, pp. 177–182.

[14] G. E. Batista and M. C. Monard, "A study of k-nearest neighbour as an imputation method," *Front. Artif. Intell. Appl.*, vol. 87, 2002.

[15] M. S. Osman, A. M. Abu-Mahfouz, and P. R. Page, "A survey on data imputation techniques: Water distribution system as a use case," *IEEE Access*, vol. 6, 2018.

[16] M. M. Rahman and D. N. Davis, "Machine learning-based missing value imputation method for clinical datasets," *Lecture Notes in Electrical Engineering IAENG Transactions on Engineering Technologies*, 2013.

[17] C. P. Robert and G. Casella, *Monte Carlo statistical methods*. Springer, 2011.

[18] T. Elsaleh, S. Enshaeifar *et al.*, "Iot-stream: A lightweight ontology for internet of things data streams and its use with data analytics and event detection services," *Sensors*, vol. 20, no. 4, 2020.

[19] J. Rocca. Bayesian inference problem, mcmc and variational inference. https://towardsdatascience.com/bayesian-inference-problem-mcmc-and-variational-inference-25a8aa9bce29, Accessed at 11/11/2019.

[20] "Context information management (cim); ngsi-ld api technical report," 2019, accessed December 2019. [Online]. Available: https://www.etsi.org/standards