Using Smartphones for Prototyping Semantic Sensor Analysis Systems

Hassan Issa
German Research Center for Artificial Intelligence & Kaiserslautern University of Technology
Kaiserslautern, Germany
Hassan.Issa@dfki.de

Ludger van Elst
German Research Center for Artificial Intelligence
Kaiserslautern, Germany
Ludger.van_Elst@dfki.de

Andreas Dengel
German Research Center for Artificial Intelligence & Kaiserslautern University of Technology
Kaiserslautern, Germany
Andreas.Dengel@dfki.de

ABSTRACT

The increasing usage of sensors in modern technical systems and in consumer products necessitates using efficient and scalable methods for storing and processing sensor data. Coupling big data technologies with semantic techniques not only helps achieving the desired storage and processing goals, but also facilitates data integration, data analysis and the utilization of data in unforeseen future applications through preserving the data generation context. In this work, an approach for prototyping semantic sensor analysis systems using Apache Spark is proposed. The approach uses smartphones to generate sensor data which is transformed into semantic data according to the Semantic Sensor Network ontology. Efficient storage and processing methods of semantic data are proposed and a use case where a smartphone is deployed in a transportation bus is presented along with a street anomaly detection application.

CCS Concepts

•**Computing methodologies** → **Semantic networks;** *Ontology engineering;* •**Information systems** → Distributed storage;

Keywords

Big Data, Sensor Data, RDF, OWL, Semantics, Sensor Networks, SSN, Spark, Ontology

1. INTRODUCTION

Modern technical systems and production processes often comprise a large number of sensors providing data about current operating conditions as well as the system's environment. Examples can be found in vehicles (private and industrial), factories ("Industry 4.0"), agriculture (weather and oil sensors) and in energy production (solar and wind power plants). In the past, sensor data has mainly been used for controlling the current operations of a system. Nowadays however, high-frequency data about operating conditions and use contexts can in addition be seen as a *strategic information asset* for the operator or manufacturer. This is made possible through the advancements in big data technologies which allow processing the vast amounts of collected sensor data and by the use of the Semantic Sensor Network (SSN) ontology which adds semantic compatibility for sensor data[2].

The use of semantics helps towards achieving autonomous processing and reasoning about sensor data despite the increasing complexity in sensor networks deployed in modern systems. It also allows easy data integration through incorporating different ontologies that describe new application contexts in order to extend the data processing and reasoning capabilities over the collected data. A great advantage in using semantic data rather than building applications on top of the raw data is that with semantics, the context of data generation is explicitly captured. This leads to easier analysis in any later applications even if they were not initially intended when the data was collected. Section 6.2 demonstrates a direct application for collected sensor data and discusses the ease of utilizing the same semantic data in further future applications.

To illustrate the potential of harnessing big data and semantic technologies for processing sensor data, we propose a simple prototyping approach that uses smartphones for transforming objects or devices into semantic sensor data sources and providing the means to semantically analyze the captured data. This approach is defined by the following stages:

1. **Generating sensor data**: Smartphones nowadays are equipped with a notable set of sensors including accelerometer, gyroscope, GPS, light sensor, etc. Given its connectivity, storage, and processing features, physically attaching a smartphone to an object of interest is enough to generate sensor data about this object.

2. **Semantic modeling of the object with deployed sensors**: Given a set of sensors being tracked, an ontology is generated based on SSN to model the use of the smartphone and its tracked sensors to monitor an object of interest.

3. **Transforming raw sensor data into semantic data**: Utilizing Apache Spark [12] to process possibly huge

SBD'16, July 01 2016, San Francisco, CA, USA
© 2016 ACM. ISBN 978-1-4503-4299-5/16/07...$15.00
DOI: http://dx.doi.org/10.1145/2928294.2928299

amounts of raw data, sensor data is transformed to semantic data using the generated SSN-based ontology. The use of big data technologies, such as Spark, aims at facilitating the process of scaling up the system when smartphones are replaced by permanent sensors and the need to process always-on sensor streams arises.

4. **Analysis and reasoning about the data**: After obtaining semantic sensor data, the possibilities for materializing semantics through reasoning about the data, also when combined with other contextual data, can be achieved. This also involves the use of Spark for robust and scalable performance.

It is worth mentioning that smartphones are not the best option when it comes to deployment of sensors in real life sensor networks scenarios. This is due to their high energy consumption and need for a permanent power connection, operating system overhead and unreliability, relatively large size, and inability to operate in several environments and conditions such as underwater or in high temperature surroundings. However, the effortless deployment of the proposed system can often yield sufficient insights and results for certain simple applications and serves as a prototype that could scale up to meet high demand application scenarios with only few changes required.

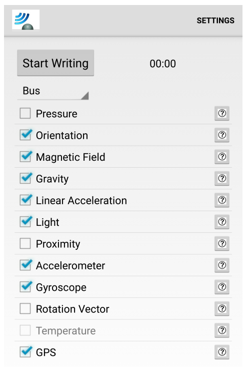
This paper proceeds as follows: Section 2 lists the related-work to this paper. Section 3 describes SensorTracker, which is an Android app developed for collecting smartphone sensor data. Section 4 presents the process of realization of semantic sensor data through generating an SSN-based ontology which is used to transform raw sensor data to semantic data. Section 5 describes how Apache Spark is used in processing and analysing sensor data. Section 6 presents a use case where the system is deployed to monitor a public transport bus. And finally, Section 7 discusses the conclusions of this research and the future work opportunities.

2. RELATED WORK

Using manual user description of sensor deployment, [13] presents an approach that helps in transforming raw sensor data into RDF conforming to SSN ontology. This method is usable for general sensor data however for the proposed smartphone based prototyping approach a fully automated transformation of sensor data is feasible as shown in Section 4. Apache Spark [12], which is the general cluster computing framework used in this work does not support semantic data natively. [3] presents an evaluation of RDF distribution algorithms using Spark, while [4] presents an encoding scheme for ontology classes and properties using Spark with a restriction to single-inheritance ontologies. In a similar use case to the one presented in Section 6, [6] presents an approach to detect potholes in streets using accelerometer data collected from taxis in the greater Boston area, USA.

3. SENSORTRACKER APP

The starting point for realization and analysis of semantic sensor data is data collection. Using smartphones for acquiring data has the advantages of abundance of sensors included, connectivity, storage space and processing power. SensorTracker is an Android app developed for the purpose of acquiring sensor data. After attaching the smartphone



- [ ] Pressure
- [x] Orientation
- [x] Magnetic Field
- [x] Gravity
- [x] Linear Acceleration
- [x] Light
- [ ] Proximity
- [x] Accelerometer
- [x] Gyroscope
- [ ] Rotation Vector
- [ ] Temperature
- [x] GPS

Figure 1: Snapshot of the SensorTracker Android app interface.

to the object of interest, sensors to be tracked are easily selected using SensorTracker interface as shown in Figure 1. Even though smartphones are not the ideal sources for sensor data streams as indicated in Section 1, the following properties were considered to overcome the limitations and utilize the advantages of using a smartphone in building SensorTracker application:

- **Robustness**: SensorTracker runs in the background and was tested for extended running times of several days recording a total of over 14 million records equivalent to 1.2 GB of raw sensor data as detailed in the experiment setup of Section 6.1.
- **Storage and Connectivity**: Sensor data is saved in the form of CSV files and can be optionally transferred or inserted to a remote database using internet connectivity when available.
- **Local Processing**: Given the processing power of modern day smartphones, SensorTracker does local aggregation of sensor data within user-specified time frames to reduce redundant data if required. Also, SensorTracker uses sensor fusion techniques to provide fused sensor data and not only data from the phone's physical sensors. For example, using accelerometer, gyroscope and magnetic field sensor data to produce fused orientation sensor data.

4. REALIZATION OF SEMANTIC SENSOR DATA

As discussed in Section 1, two steps are needed to transform raw sensor data into semantic data. The first step in this transformation is the creation of a conceptual model to describe the created sensor network. Combining the user's choice of tracked sensors in the smartphone application presented in Section 3, the user's manual description of the object to which the smartphone is deployed, and the Semantic Sensor Ontology (SSN), an SSN-based ontology is created to describe the sensors attached to the object of interest as specified in Section 4.1. A mapping function is then used to create all the needed RDF triples for each sensor data record collected. This mapping is further discussed in Section 4.2.

Since transforming raw sensor data into semantic data comprises processing huge amounts of data, a Spark based storage and processing approach introduced in Section 5 is used in order to enhance the performance while preserving the semantic structure of data presented in this section.

4.1 Generating an SSN-Based Ontology

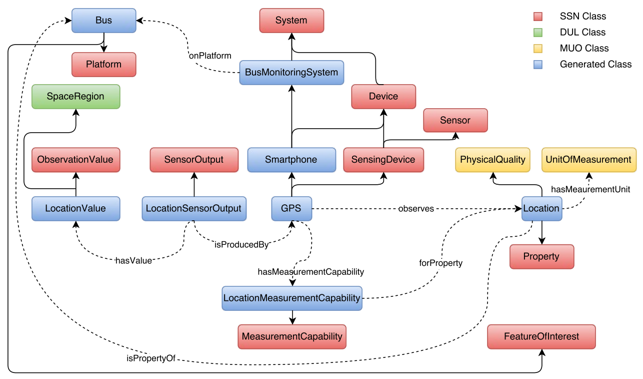


Figure 2: Example of a generated SSN-Based ontology describing a smartphone with GPS sensor connected to a bus.

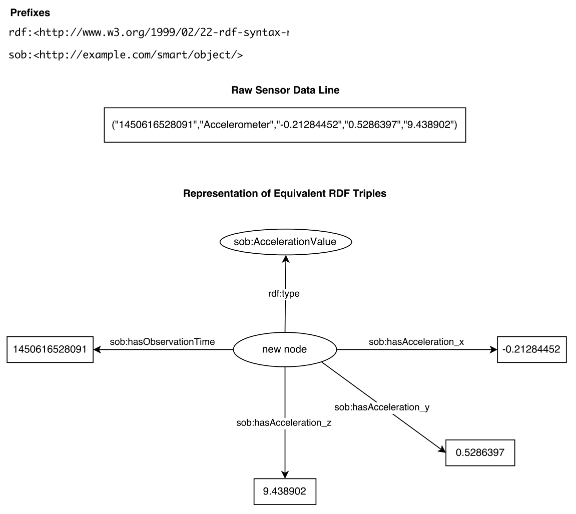
The W3C Semantic Sensor Network Incubator Group[1] (SSN-XG) developed the Semantic Sensor Network ontology as an ontology to describe sensors and sensor networks for the use in sensor web and sensor network applications. The SSN ontology is created around the Stimulus-Sensor-Observation design pattern [8] where the act of sensing is conceptually separated into three parts: a stimulus, a sensor and an observation. In addition to its *Skeleton* module that describes the sensing activity, several conceptual modules are built in the SSN ontology to cover key sensor concepts such as *Deployment*, *Device*, *System*, *OperatingRestriction* and *Data*. The Semantic Sensor Network ontology was quickly adopted in research and applications and became the standard ontology for semantic sensor networks.

Using the simple assumption of attaching the sensing device (smartphone) to any object of interest, and knowing the available sensors on the smartphone, the proposed system generates an ontology based on SSN to describe the sensors deployment setup. Figure 2 shows a part of an example ontology generated to describe a smartphone attached to a public transportation bus. For the readability of the figure, only the GPS sensor is displayed. The generated ontology considers the object of interest as sub-class of both the `ssn:Platform` and `ssn:FeatureOfInterest` classes. Since SSN does not include a model for describing sensor values and their units of measurement, the SSN-based ontology is complimented by the use of the Dolce Ultralite Ontology(DUL) [9] and the Measurement Units Ontology(MUO) [1].

4.2 Mapping Raw Sensor Data to RDF Triples

As previously mentioned in Section 3, the SensorTracker app records the sensor data for all the smartphone sensors chosen by the user in CSV format. After parsing the raw data files and discarding any erroneous records, a group of RDF triples are created depending on the type of the sensor in each data record and added to the base ontology. Figure 3 shows an example of an acceleration data record and its mapping to its equivalent RDF triples.

[1]https://www.w3.org/2005/Incubator/ssn/



Prefixes
rdf:<http://www.w3.org/1999/02/22-rdf-syntax-r
sob:<http://example.com/smart/object/>
Raw Sensor Data Line
("1450616528091","Accelerometer","-0.21284452","0.5286397","9.438902")
Representation of Equivalent RDF Triples
sob:AccelerationValue
rdf:type
new node
sob:hasObservationTime
1450616528091
sob:hasAcceleration_x
-0.21284452
sob:hasAcceleration_y
0.5286397
sob:hasAcceleration_z
9.438902

Figure 3: Example of mapping raw sensor data record to its equivalent RDF triples.

5. SPARK INTEGRATION

Current and emerging use cases involve huge amounts of raw sensor data which imposes the use of a scalable cluster computing framework that ensures data parallelism and fault-tolerance. Apache Spark [12] is a general engine for large-scale data processing that utilizes in-memory computing to achieve up to 100x better performance than the state of the art MapReduce [5] approach. Spark's efficient performance is a result of using a read-only data structure called Resilient Distributed Datasets (RDDs) [11]. RDDs can be stored in main memory which makes a significant difference especially with iterative algorithms where intermediate results do not need to be replicated and persisted on the distributed file system with each iteration. To maintain fault tolerance, Spark records all transformations and rebuilds any lost RDDs by reapplying the recorded transformations on the original datasets.

In any ontology knowledge base, a natural separation exists between *TBox* and *ABox* statements where the former denotes the set of classes and properties that describe the conceptualization of a system and the latter denotes a set of facts about individuals belonging to these classes. This separation is used to distinguish between the base ontology and the generated RDF statements representing the collected sensor data.

5.1 Processing of TBox Statements

Given the relatively small size of a base ontology, there is no need for a distributed processing of its statements. Instead, the generated OWL ontology is processed locally to produce an encoding of the TBox statements which is passed as a Spark broadcast variable to be available during runtime on all the working nodes of the Spark cluster. Two tables for the classes and the properties of the ontology are created:

- **Ontology Classes**: For each class, a mapping is created from its URI to a unique numerical ID and a set of the IDs of its sub-classes using `rdfs:subClassOf` property.

- **Ontology Properties**: For each property, a mapping is created from its URI to a unique numerical ID and the numerical IDs for its domain(`rdfs:domain`) and range(`rdfs:range`) classes as well as a set of the IDs of its sub-properties using `rdfs:subPropertyOf` property.

The use of numerical IDs to replace string based ontology URIs enhances the utilization of the available memory and improves the performance of identifiers' comparisons. Note that using the transitivity of `rdfs:subClassOf` and `rdfs:subPropertyOf` properties, the set of sub-classes and sub-properties are computed by recursively finding the sub-entities of the direct sub-classes (resp. sub-properties) of a class or a property. [4] uses a similar encoding however does not account for multiple inheritance of classes and properties which is common in most ontologies such as the SSN ontology used here.

5.2 Processing of ABox Statements

For the ABox statements, which comprise the vast majority of the statements in a knowledge base, Spark is used in all operations involving these statements starting with the creation of the statements using raw sensor data and in all the following steps including querying the knowledge base and analyzing its data. As indicated in Section 4.2, a set of RDF statements are assigned to each record of sensor data. These statements however, are not stored in a single RDD containing all the triples created. Instead, an RDD is created for each property containing a key-value pair representing the subject and the object of the conceptual RDF triple. Figure 4 shows an example outcome for this process where numerical IDs are replaced with textual IDs for clarity. The analogy to column-oriented DBMS enhances the performance where not all triples need to be loaded into main memory as it is the case with many sensor data analysis scenarios such as the use case presented in Section 6.2.

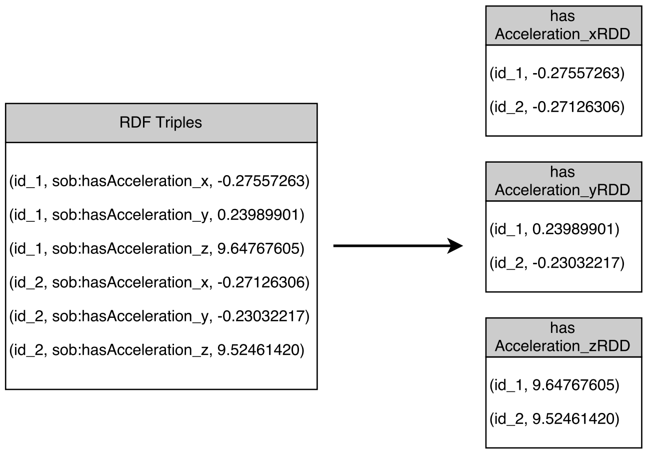

RDF Triples
(id_1, sob:hasAcceleration_x, -0.27557263)
(id_1, sob:hasAcceleration_y, 0.23989901)
(id_1, sob:hasAcceleration_z, 9.64767605)
(id_2, sob:hasAcceleration_x, -0.27126306)
(id_2, sob:hasAcceleration_y, -0.23032217)
(id_2, sob:hasAcceleration_z, 9.52461420)
has Acceleration_xRDD
(id_1, -0.27557263)
(id_2, -0.27126306)
has Acceleration_yRDD
(id_1, 0.23989901)
(id_2, -0.23032217)
has Acceleration_zRDD
(id_1, 9.64767605)
(id_2, 9.52461420)

Figure 4: An example of storing ABox RDF statements in properties' RDDs.

Note that restoring the triples format is a straight forward process where for each element in a property RDD, a triple can be formed from the element's key and value as the subject and the object of the triple respectively, and the property represented by the RDD as the property of the triple.

5.3 Analyzing Semantic Sensor Data

In order to query the semantic sensor data using Spark, the query is transformed into an equivalent set of Spark operations. [10] proves that any conjunctive query could be transformed into a set of the following Spark operations:

- **Map**: applies a specified function to all the elements of an RDD.
- **Filter**: returns a new RDD consisting of the subset of the data in an existing RDD that satisfies a certain predicate.
- **Join**: joins two RDDs based on the equality of their respective elements' keys.

This essentially permits the execution of SPARQL queries on the dataset through transforming the SPARQL query into its equivalent set of Spark operations. A simple example of such transformation is presented in Section 6.1. Additionally, analysts could further utilize Spark transformations and actions like *sortByKey*, *ReduceByKey*, *count* and *reduce* to get more insights from the dataset.

6. EXAMPLE USE CASE

The main motivation for using the system proposed is being able to effortlessly obtain a working prototype of sensors attached to an object of interest, modeling the obtained sensor deployment according to the SSN ontology, and enabling the design of big data algorithms and results analysis to run on the collected sensor data in addition to other possible integrated data from different sources. This usage scenario conveniently scales up when applied to different sensor data sources and much larger amounts of collected data. In what follows, Section 6.1 introduces the experiment setup for the presented use case while Section 6.2 discusses the application details.

6.1 Setup and Collected Data

In order to test the system, we used a simple setup where an Android smartphone (Samsung Galaxy Note 4) was attached to a bus of the Rhine-Neckar public transport network (VRN[2]) that operates on multiple lines in the city of Kaiserslautern in Germany. The smartphone was stored inside a closed cabinet in the bus and connected to a stable power supply through a USB cable. To achieve consistent sensor recordings that reflect the movements of the bus rather than the smartphone itself, the phone was fixed to a keep a vertical position along the experiment period of 8 days. Over this time, the app recorded a total of $14,248,629$ records in which the bus crossed a distance of over 1600 km.

Figure 5 shows the trajectory followed by the bus during the 8-day experiment in addition to the bus stops located in Kaiserslautern.

To illustrate the ability to query the data using the proposed system as introduced in Section 5, the following SPARQL query to retrieve all bus trajectory points is presented with its equivalence in Spark using Spark's Python API:

```
SELECT ?lat ?lon
    WHERE{
?a rdf:type sob:LocationValue.
```

[2]http://www.vrn.de/

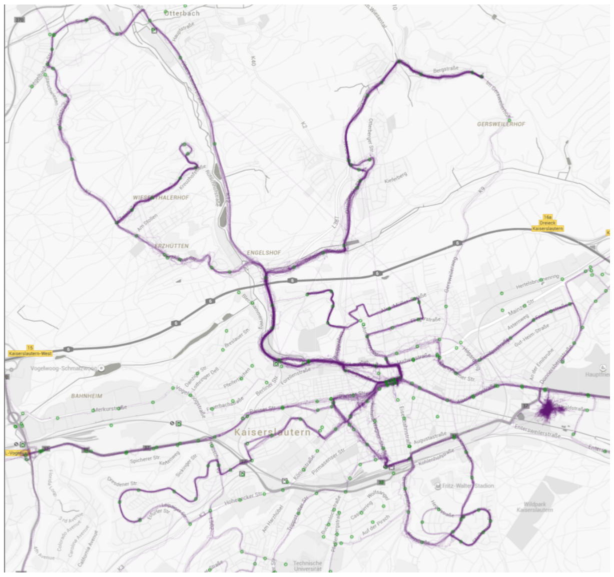


Figure 5: Bus trajectory followed during the experiment(purple) with locations of bus stops(green).

```
?a sob:hasLatitude ?lat.
?a sob:hasLongitude ?lon.
}
```

which is equivalent to the following operations pipeline in Spark:

```
(TypeRDD
 .filter(lambda (nodeID,typeID):
        equalsType(typeID,"sob:LocationValue") )
 .join(hasLatitudeRDD)
 .join(hasLongitudeRDD)
 .map(lambda (nodeID,(typeID,lat,lon)): (lat,lon))
).collect()
```

Note that the method `equalsType()` uses the TBox encoding availability in memory to verify that the node type is either equal to the class `sob:LocationValue` or to one of its sub-classes.

6.2 Street Quality Assessment: Anomaly Detection

As an application for the analysis of the collected public transport bus data using the proposed system, the acceleration and the location data are utilized to assess the quality of the streets of the city through detecting street anomalies. Street anomalies include potholes, manholes, bumps and any other damages to the street that disturb the driving experience. By providing a simple experimental definition of an anomaly as a location where a spike of at least $1.8m/s^2$ occurs within a 2 seconds time frame along the z-acceleration axis, we were able to locate locations of street anomalies on the streets travelled by the bus. Figure 6 shows the bus acceleration data along all axes with a clear spike in z-acceleration value indicating the presence of a street anomaly.

In this paper, the focus is on demonstrating the ease of building applications and semantic reasoning on top of the obtained sensor data knowledge base. The evaluation of

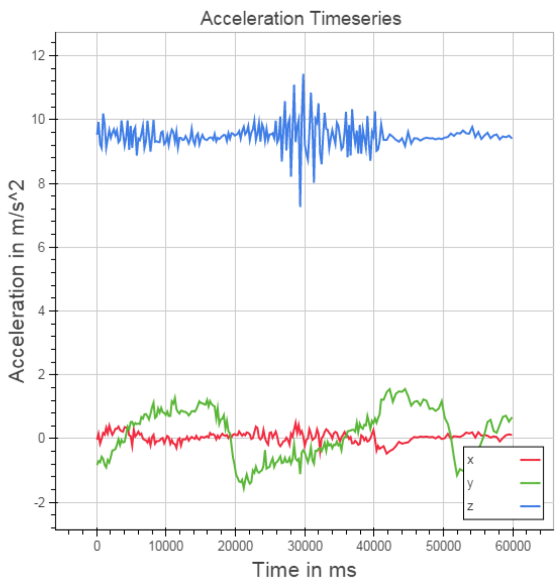


Figure 6: One minute of acceleration data with a spike in acceleration across z-axis denoting presence of a street anomaly.

the precision of the introduced street anomaly detector is hence omitted. Still, as a refinement of the results, clustering of the retrieved street anomaly locations is applied using DBSCAN [7] with haversine distance as the spatial distance metric. The results of this clustering is shown in Figure 7.

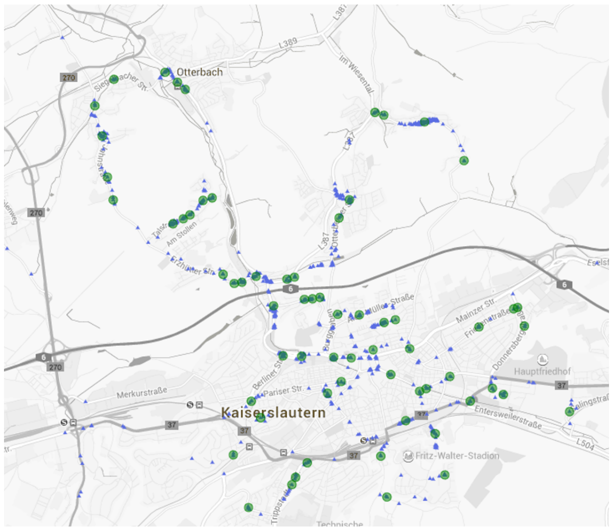


Figure 7: Locations of clusters of street anomalies (green) with single anomalies displayed in blue color.

In a related application, [6] utilizes acceleration data from taxi cars to distinguish the different types of street anomalies. Using smartphones, the detection of anomalies is computed locally on the phone where only locations of detected anomalies are stored and later clustered in order to obtain a final result. As this might be sufficient for this exact use case, the original sensor data collected and processed locally in smartphones is lost and could not be utilized in any further applications different from the original application context of detecting potholes. Following our proposed approach, all the raw sensor data was to be stored and processed using big data techniques. And by storing the data in semantic form, many other applications could be built on top of the data by integrating different information sources. Potential examples are analyzing the demands for taxis at times of concerts or sports events, analyzing the activity per taxi station, and analyzing taxi delays if integrated with the company reservations data.

7. CONCLUSIONS AND FUTURE WORK

In this work, an approach for prototyping semantic sensor analysis systems is proposed. This approach starts by physically attaching a smartphone to an object of interest and recording the sensor value variations through an Android app. The use of smartphones for sensor tracking is not vital for the following steps as data from any source such as Internet of Things (IoT) devices or dedicated sensors could be used. However, the usage of a smartphone is sufficient for simple applications and for prototyping more complex ones. An ontology based on the Semantic Sensor Network ontology is then generated to model the setup and add an abstraction layer for analysing the sensor data. To provide the scalability required in data intensive scenarios, big data technologies are used for transforming raw sensor data into semantic data and for the latter processing of the obtained semantic data. Using the conceptual division between TBox and ABox statements in the resulting ontology, an effective approach is presented to make TBox statements available in memory at runtime in all working nodes of an Apache Spark cluster while storing the ABox statements in semantic property RDDs in an analogy to column-oriented storage in database systems. An example deployment of a smartphone in a public transportation bus is presented and the data collected is processed to provide potential locations of street anomalies.

For future work, the techniques for storing and processing the data will be evaluated and further optimizations for data distribution are to be considered. Despite the ability to represent any SPARQL query using Spark operations, supporting SPARQL queries by the system adds a second layer of abstraction with which many data analysts are familiar. And for the public transportation use case, extending the ontology to model the bus network can facilitate realizing several applications beyond street quality assessment such as analysing bus delays and finding their causes.

8. ACKNOWLEDGMENTS

This work has been supported by Stiftung Rheinland-Pfalz für Innovation, (Grant 386261-1181, Project AnaKonDa Analyse und Kontextualisierung von Sensor-Daten mit semantischem Hintergrundwissen). The initial version of the SensorTracker app has been developed within the MyCustomer project, supported by Ministerium für Bildung, Wissenschaft, Weiterbildung und Kultur des Landes Rheinland-Pfalz.

9. REFERENCES

[1] Measurement units ontology (muo). http://idi.fundacionctic.org/muo/. Accessed: 2016-02-29.

[2] M. Compton, P. Barnaghi, L. Bermudez, R. García-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, A. Herzog, V. Huang, K. Janowicz, W. D. Kelsey, D. L. Phuoc, L. Lefort, M. Leggieri, H. Neuhaus, A. Nikolov, K. Page, A. Passant, A. Sheth, and K. Taylor. The {SSN} ontology of the {W3C} semantic sensor network incubator group. *Web Semantics: Science, Services and Agents on the World Wide Web*, 17(0):25 – 32, 2012.

[3] O. Curé, H. Naacke, M.-A. Baazizi, and B. Amann. On the evaluation of rdf distribution algorithms implemented over apache spark. *arXiv preprint*, 2015.

[4] O. Curé, H. Naacke, T. Randriamalala, and B. Amann. Litemat: a scalable, cost-efficient inference encoding scheme for large rdf graphs. In *Big Data (Big Data), 2015 IEEE International Conference on*, pages 1823–1830. IEEE, 2015.

[5] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[6] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan. The pothole patrol: using a mobile sensor network for road surface monitoring. In *Proceedings of the 6th international conference on Mobile systems, applications, and services*, pages 29–39. ACM, 2008.

[7] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[8] K. Janowicz and M. Compton. The stimulus-sensor-observation ontology design pattern and its integration into the semantic sensor network ontology. In *Proceedings of the 3rd International Conference on Semantic Sensor Networks-Volume 668*, pages 64–78. CEUR-WS. org, 2010.

[9] C. Masolo, S. Borgo, A. Gangemi, N. Guarino, A. Oltramari, and L. Schneider. Dolce: a descriptive ontology for linguistic and cognitive engineering. *WonderWeb Project, Deliverable D*, 17, 2003.

[10] R. S. Xin, J. Rosen, M. Zaharia, M. J. Franklin, S. Shenker, and I. Stoica. Shark: Sql and rich analytics at scale. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of data*, pages 13–24. ACM, 2013.

[11] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 2–2. USENIX Association, 2012.

[12] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: Cluster computing with working sets. *HotCloud*, 10:10–10, 2010.

[13] X. Zhang, Y. Zhao, and W. Liu. A method for mapping sensor data to ssn ontology. *International Journal of u-and e-Service, Science and Technology*, 8(9):303–316, 2015.