PLOMaR: An Ontology Framework for Context Modeling and Reasoning on Crowd-Sensing Platform

Yogesh Jagadeesan, Peizhao Hu and Carlos R. Rivero
Department of Computer Science
Rochester Institute of Technology, USA
Email: {yj6026, hxpvcs, crrvcs}@rit.edu

***Abstract*—Crowd-sensing is a popular way to sense and collect data using smartphones that reveals user behaviors and their correlations with device performance. PhoneLab is one of the largest crowd-sensing platform based on the Android system. Through experimental instrumentations and system modifications, researchers can tap into a sea of insightful information that can be further processed to reveal valuable context information about the device, user and the environment. However, the PhoneLab data is in JSON format. The process of inferring reasons from data in this format is not straightforward. In this paper, we introduce *PLOMaR* — an ontology framework that uses SPARQL rules to help researchers access information and derive new information without complex data processing. The goals are to (i) make the measurement data more accessible, (ii) increase interoperability and reusability of data gathered from different sources, (iii) develop extensible data representation to support future development of the PhoneLab platform. We describe the models, the JSON to RDF mapping processes, and the SPARQL rules used for deriving new information. We evaluate our framework with three application examples based on the sample dataset provided.**

I. Introduction

The proliferation of mobile devices, smartphones in particular, has changed the way services are delivered. Mobile devices generate a lot more digital content than desktop computers [1]. To provide appropriate services tailored to users' current situations, applications often rely on information gathered from users and environments using sensors onboard the mobile devices [2], [3]. This information, also known as context, helps automate system adaptations to users' changing situations [4], [5].

To better understand correlations between mobile data, crowd-sensing platforms were developed to facilitate the task of sensing and collecting relevant context information from a large group of participants. PhoneLab [6] for example, is one such crowd-sensing platform developed at the University of Buffalo. It is a large scale, open-access, smartphone testbed consisting of hundreds of Android devices. Participants receive subsidized call rates for participating in the crowd-sensing effort. Through experimental instrumentations and direct modification of the operating system's source code, researchers can use the provided JSON scheme and API functions to extract information of interest in order to study user behaviors and their correlations with device performance. Some examples include crowd-sensing network utilization in order to compute better resource allocation [7] and assisting access points to make channel assignment decisions with mobile clients [8]. Researchers can instrument participants' devices with functions that record parameters either periodically, or triggered by events. This measurement data is sent back to a server when devices are connected to trusted networks. Currently, there are five existing instrumentations on Android devices including, but not limited to, location, network, and battery power[1]. Data collected is stored in JSON format. This measurement data needs to be processed to make sense of implicit correlations between data. In this paper, we aim to make this measurement data more accessible, thus improving interoperability of data from different sources. We leverage modeling and reasoning techniques from the semantic web and context modeling communities.

Context information can be represented using context modeling techniques such as object-role model, ontology model and spatial model [9]. Such representations facilitate context management, support for reasoning and also derivation of new context information. Among those, ontology modeling has been widely used in prototype systems that are context-aware due to its reasoning capability being well supported [3], [9]. Ontology was used in [10] to capture context information and was combined with rules to derive new context information. It can represent a shared conceptualization of a particular domain. It is also a collection of axioms that offer constraints, meaning, rules and heuristics that can derive new information [3]. Our goal is to develop a mobile framework for automatic identification of new context based on implicit rules and statistical reasoning.

In this paper, we introduce *PLOMaR,* an ontology framework that captures measurement data and implicit relationships among them by applying ontology modeling and developing rules. Through the development of such framework, we present three main contributions in this paper: (i) create ontology-based context models for large scale crowd-sensing smartphone testbed, (ii) define rules that temporally correlate data to derive new information, (iii) support SPARQL queries to reveal user behavior.

[1]Example Instrumentation, https://phone-lab.org/experiment/existing/

This paper is organized as follows. Section II describes how we have modeled PhoneLab data and new concepts for deriving new context information. Section III discusses example SPARQL queries that demonstrate the usefulness of this work. This is followed by descriptions of some of the existing, related works in Section IV. The paper concludes in Section V and discusses future work.

II. The PhoneLab case study

In this section, we show an example of raw, mobile data recorded by PhoneLab, present the ontology that we devised to represent this raw data, and also describe how to model newly inferred context data to be stored back into our ontology.

A. Mobile data recorded on PhoneLab

Through experimental instrumentations or platform modifications on Android sources, data reflecting the performance of mobile devices and user behavior can be extracted in the form of JSON objects. A total of over 40 GB of sample data was available from their website[2] featuring data gathered from 11 PhoneLab devices. This formed the basis for our ontology. The sample dataset consists of one file per device per tag per day in this format: {*device_identifier*}/{*tag*}/{*year*}/{*month*}/{*day*}.*out.gz*.

Following this convention, we develop a mapping algorithm to automatically create individuals that conform to the ontology. Currently, there are eight groups of information covering five aspects of mobile devices, including location, network, package management, power and security. Each measurement data is reported as one of the 25 actions of an information group. The following snippet shows an example of the measurement data triggered when there is a change in location.

```
Location-Misc-PhoneLab {
  "Action":"android.location.LOCATION_CHANGED",
  "Counter":2159,
  "LogFormat":"1.0",
  "Location": {
     "ElapsedRealtimeNanos":54026639248,
     "HasSpeed":false,
     "IsFromMockProvider":false,
     "HasBearing":false,
     "Accuracy":1854,
     "Altitude":0,
     "Speed":0,
     "Time":1427045651782,
     "HasAltitude":false,
     "Latitude":42.9931695,
     "Longitude":-78.7472696,
     "Bearing":0,
     "Provider":"fused",
     "HasAccuracy":true
   }
}
```

Within each of these information groups, there are three common fields: `Action` corresponds to an Android intent, such as `LOCATION_CHANGED` (one of the 25 possible actions) as illustrated; `Counter` represents an event sequence ID; and `LogFormat` is simply a version number of the logging format. The rest of the location related information is self-explanatory. In addition, each measurement data is associated with the timestamp it was recorded at. In our example, this corresponds to "2015-03-01 00:03:47". We use this timestamp to correlate measurement data from two or more instrumentations. Note that some events might be recorded periodically and this period can vary from a few milliseconds to a few seconds. For example, information about battery charging and discharging and also information about updates in location are recorded this way. Some other events are recorded only when a change of state is detected. For example, information about a change in state of the phone's screen (from locked to unlocked) is recorded this way.

[2]https://phone-lab.org/static/experiment/sample_dataset.tgz

B. Ontology

Extracting PhoneLab measurement data from JSON objects requires a custom parser. Although retrieving a small set of data from the JSON objects can be straight-forward, making correlations between data directly from those JSON objects can be challenging. To facilitate this process, we use ontology models to represent the measurement data and their relationships. We develop a parser to appropriately map JSON formatted data as individuals of the corresponding ontology classes. The models are designed with a goal to maximize information interoperability and reusability. Figure 1 depicts an excerpt of our ontology using a UML-like notation. Due to lack of space, we only present classes that are used to support our discussions. (Prefix `plo:` corresponds to URI "https://phone-lab.org/ontology/".)

We model each measurement data entry as an action by creating individuals of `plo:PhoneAction` class. As shown in the `Location-Misc-PhoneLab` example, each measurement data block contains action, counter and log format, which can be represented as data properties. These properties are mapped to `plo:action`, `plo:counter` and `plo:logFormat` in the ontology. Each `plo:PhoneAction` happens at a specific timestamp, `plo:timestamp`, which is a property of the `plo:Event` class. When two actions are found to happen within the same timestamp, sharing the same individual of the `plo:Event` type, we relate these two actions together. In other words, we correlate measurement data recorded roughly at the same time, within a second.

The following snippet shows example RDF data, using prefix `pld:` as URI "https://phone-lab.org/data/", for correlating phone charging and location changed action.

```
pld:Action2159 rdf:type        plo:LocationChanged ;
               plo:action
                 ''android.location.LOCATION_CHANGED'' ;
               plo:has         pld:Coord783 .
pld:Coord783   rdf:type        plo:Coordinate ;
               plo:long        -78.7472696 ;
               plo:lat         42.9931695 .
pld:Event284   rdf:type        plo:Event ;
               plo:timestamp   ''2015-03-01 00:03:47'' .
pld:Action2159 plo:happened    pld:Event284 .
pld:Action2163 rdf:type        plo:BatteryChanged ;
               plo:happened    pld:Event284 .
```

In this case, we have an individual `pld:Action2159` of type `plo:LocationChanged`, `pld:Action2163` of type `plo:BatteryChanged`, and `pld:Event284` which

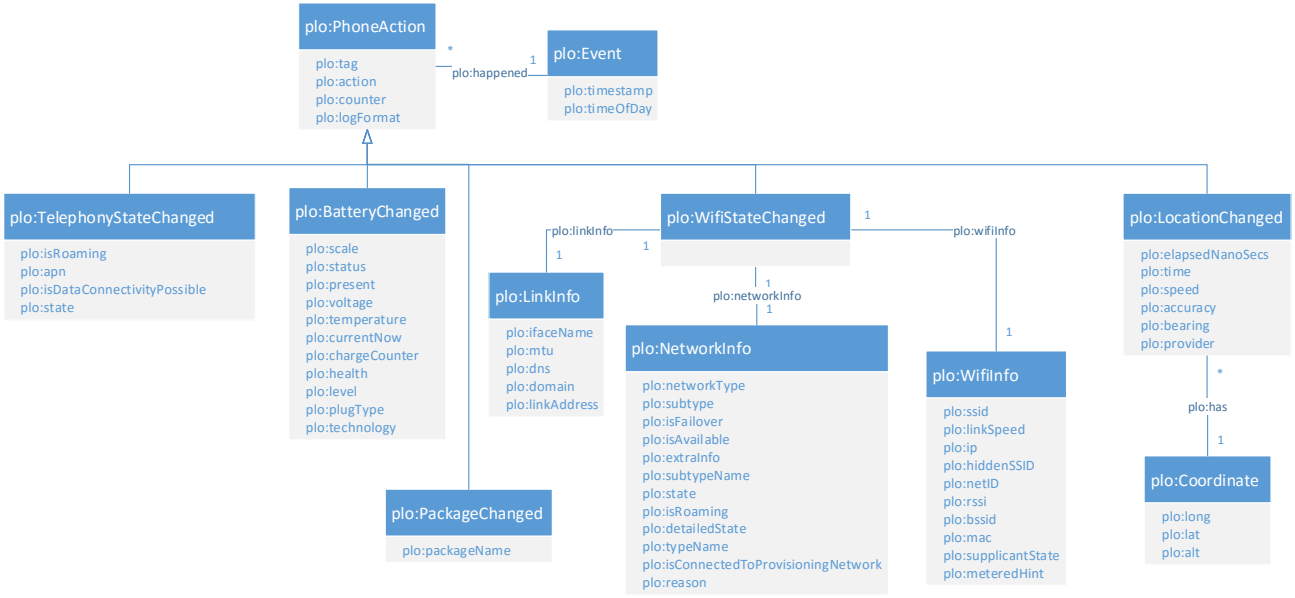


Fig. 1. Excerpt of our PhoneLab ontology

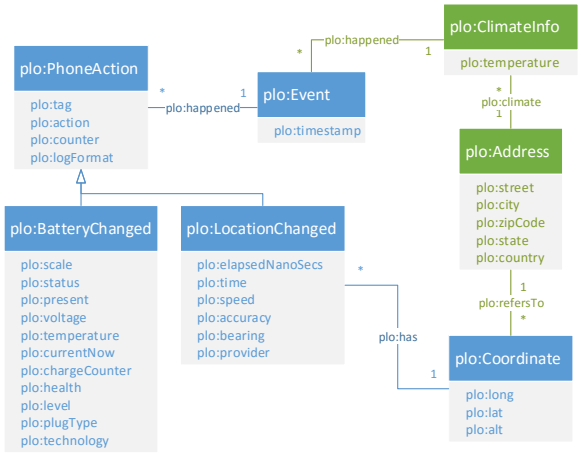


Fig. 2. Excerpt of the inferred data

is the timestamp connecting both actions. For clarity, we omitted some properties in the example.

C. Modeling inferred data

With the base ontology, we are interested to investigate whether we can derive new context information. Figure 2 illustrates the extended model of the derived context information "address has climate information". In this case, we correlate `plo:LocationChanged` with `plo:BatteryChanged` to infer the temperature of a location by using the temperature measurement of the battery. As shown in the figure, the two classes `plo:ClimateInfo` and `plo:Address` are derived from existing information.

We use SPARQL's `CONSTRUCT` clause to extend our base ontology. The following code snippet shows how we create the two new classes shown in Figure 2. Again, we omit some properties for clarity. We use Google's reverse geocoding service to resolve the address information from a coordinate.

```
CONSTRUCT {
  _:ci rdf:type plo:ClimateInfo ;
         plo:temperature ?temp ;
         plo:happened ?event .
  ?addr rdf:type plo:Address ;
         plo:climate ?temp .
} WHERE {
  ?event rdf:type plo:Event ;
         plo:timestamp ?time .
  ?coord rdf:type plo:Coordinate ;
         plo:refersTo ?addr .
  ?lc rdf:type plo:LocationChanged ;
         plo:happened ?event ;
         plo:coordinate ?coord .
  ?bc rdf:type plo:BatteryChanged ;
         plo:temperature ?temp ;
         plo:happened ?event . }
```

Note that *_:ci* is a blank node, i.e., a placeholder, entailing a new URI that will be automatically generated by the SPARQL query engine when running this query.

III. Implementation and Evaluation

In the PLOMaR framework, a parser was developed in Java to map the PhoneLab measurement data in JSON format onto an ontology. Protege OWL APIs were used to programmatically create the ontology models, including creating classes, specifying relationships between classes, and defining object and data properties. SPARQL APIs were used to support queries and creation of new ontology for the derived context information. In this section, we present three examples in which SPARQL rules were used to derive new context information that reveals user behavior. The example SPARQL rules and the results shown in this section are presented to demonstrate the feasibility of our framework using the sample dataset. More interesting insights can be discovered when this framework is applied to user data.

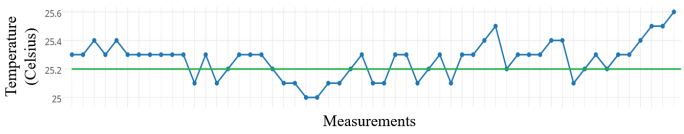
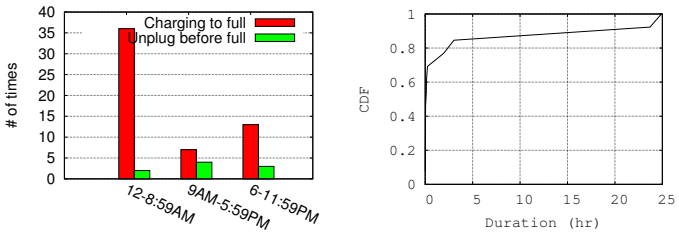


Fig. 3. Battery temperature measurements.



(a) (b)

Fig. 4. (a) Number of times battery is charged to full, (b) Distribution of the durations.

A. Inferring temperature at a location

As discussed in Section II-C, we construct SPARQL rules to correlate `plo:BatteryChanged` and `plo:LocationChanged` in order to infer the temperature at a location. Temperature is one of the measurements within battery properties recorded by PhoneLab instrumentations. If we assume the battery temperature is more or less an approximation of the temperature of the surroundings, we can estimate the temperature at the location. In this experiment, we ignore the instances where battery is charging since it might result in higher temperatures than normal. Using the previously discussed SPARQL rule, we can retrieve a list of locations and the corresponding average temperatures of days. Figure 3 shows the instantaneous battery temperature measurements recorded every few minutes and the average battery temperature on March 16, 2015. From the sample dataset, we know that the average temperature in the PhoneLab office on March 16, 2015 was 25.2 degree Celsius. This information indicates the phone has mostly been indoors, since temperature in March, at Buffalo, NY, should be a lot lower. Again, the consensus is that the data was collected from a demo phone in a lab.

B. Learning battery charging patterns

In the second example, we construct queries to investigate a user's battery charging patterns. By defining three segments in a day, such as `pld:BeforeDay` (12-8:59AM), `pld:DuringDay` (9AM-5:59PM) and `pld:AfterDay` (6-11:59PM), and by retrieving battery statuses that indicate `pld:Full`, we can learn how often users charge their phones and when. The SPARQL query below retrieves the number of times battery is charged to full during BeforeDay segment. Changing the `plo:status` value to "Charging", we can retrieve the number of times the phone has been unplugged before charging to full during the same time segment.

```
SELECT ( COUNT(?bc) AS ?count ) WHERE {
  ?bc rdf:type plo:BatteryChanged ;
      plo:status pld:Full ;
      plo:happend ?event .
  ?event plo:timeOfDay pld:BeforeDay . }
```

Figure 4(a) shows battery charging patterns during different segments of a day over twelve days. As expected, in most cases this phone was charged to full during the after work hours. During this measurement period, there were not many urgencies (such as meetings) that forced the user to disconnect the phone before its charged to full. Rate of interruptions of battery charging to full during work hours is high, which was just as expected. By modifying the query slightly, we can retrieve the average duration users spend on charging their phones in two situations: charging to full or unplugging the phone before fully charged. Table I shows the results. These statistics indicate that the phone was under constant use. The average number of times battery was charged to full is around five per day. This ratio implies that (i) The phone is likely plugged into charging whenever possible and (ii) On many occasions, the phone was unplugged for short periods of time and then reconnected. Hence, we observe a high average number within a short time frame. These behaviors are confirmed when we looked into the distribution of the charge durations, as shown in Figure 4(b). Around 70% of the data indicates that the charging duration is very less (about a minute). We also found the abnormally long charging duration of 24 hours.

TABLE I
AVERAGE HOURS USER SPENDS ON CHARGING THE PHONE.

	12-8:59AM	9AM-5:59PM	6-11:59PM
Charging to full	3.96	1.08	4.17
Unplug before full	4.12	0.49	0.19

In addition, we construct a SPARQL query to temporally correlate different battery charging statuses with location. The following query will return the list of locations where the battery has been charged to full.

```
SELECT  ?addr  WHERE {
  ?event rdf:type plo:Event ;
         plo:timestamp ?time .
  ?bc rdf:type plo:BatteryChanged ;
         plo:status pld:Full ;
         plo:happened ?event .
  ?coord rdf:type plo:Coordinate ;
         plo:refersTo ?addr .
  ?lc rdf:type plo:LocationChanged ;
         plo:happened ?event ;
         plo:coordinate ?coord .
}
```

From the above query, we found that the phone has always been charged to full in the computer science department at University of Buffalo. This verifies that the sample dataset is generated likely from a demo phone.

C. Determining network type for downloads

The general perception is that cellular network is slow and expensive. For this reason, users tend to defer any install/update of mobile application until a WiFi network is in range.

In the last example, we attempt to construct rules that allow researchers to investigate such user behavior directly from the ontology models. Note that `plo:PackageChanged` is an instrumentation that records package status changes, including installation, modification, and uninstallation from the user's phone. `plo:TelephonyStateChanged` instrumentation contains data related to cellular network activities, while `plo:WifiStateChanged` instrumentation contains data about WiFi. By temporally correlating information from these three classes, SPARQL queries can be constructed to retrieve instances indicating the type of network (WiFi or cellular) that has been used when a package is being installed or updated.

```
SELECT ?network ?pkgName WHERE {
  ?event rdf:type plo:Event ;
     plo:timestamp ?time .
  ?pc rdf:type plo:PackageChanged ;
     plo:packageName ?pkgName ;
     rdf:type plo:PhoneAction ;
     plo:action ?action ;
     plo:happened ?event .
  ?tc rdf:type plo:TelephonyStateChanged ;
     plo:state pld:Connected ;
     plo:happened ?event .
  ?wc rdf:type plo:WifiStateChanged ;
     plo:NetworkInfo ?netinfo ;
     plo:happened ?event .
  ?netinfo rdf:type plo:NetworkInfo ;
     plo:state pld:Connected .
  FILTER(
    (?action="android.intent.action.PACKAGE_CHANGED"
    || ?action="android.intent.action.PACKAGE_ADDED")
    && (?network=?tc || ?network=?wc) )
}
```

A SPARQL query is similar to a SQL query in the `SELECT` and `WHERE` clauses. Additional `FILTER` statement is used to limit the search space. Since we are only interested in the network bandwidth usage behavior, we modify the query to retrieve instances in which user performs package installation or update on either of the network types. Adding a `COUNT()` function, we get the number of packages being updated while the phone is connected to either of the network types. Figure 5 shows the results found from the sample dataset of three users. Interestingly, the sample dataset contains data that shows users were either using only WiFi or only cellular for the entire measurement duration. Also, User3 seemed to only use the cellular network for all application installations and updates. A short discussion with the PhoneLab team confirmed that due to the sparse network coverage, they sometimes disable WiFi even if the demo phone is stationary. If we apply PLOMaR to the actual user data, we can learn various aspects of user preferences towards network usage. For example, what type of mobile applications are more popular when people are on the move?

IV. Related work

The use of ontology models and SPARQL queries to facilitate information reuse, management, and retrieval has been well studied in the semantic web community. Examples include semantically correlating journals and articles [11], mapping sensors and their observation to address heterogeneity issues [12], capturing semantic relations between concepts [13]

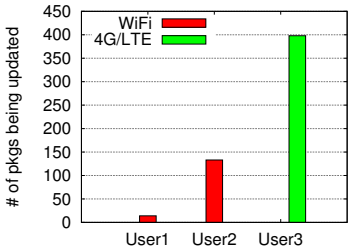


Fig. 5. Number of packages installed/updated over different networks.

and optimizing the data mining processes with detailed ontology [14]. In each of the existing work, an application-specific ontology model was developed. For example, Keet et al. [14] proposed Data Mining Optimization Ontology (DMOP). Although domain models for data mining exist, optimization of the data mining process was not possible due to the lack of detailed knowledge. DMOP was proposed to automate the selection of critical elements in the data mining process such as algorithms, models, and parameters. In contrast to seeing data mining algorithm as black boxes, focusing mostly on the input data and output hypotheses, DMOP conceptualizes the internals of the algorithm. It optimizes the data mining process by detailing knowledge of how components from different phases interact and how each component's characteristics influence the process performance.

Similarly, ontology has also been used extensively to model context information [2], [3], [9]. Wang et al. [15] marked one of the early attempts to model context using ontology. Henricksen et al. [16] described a hybrid model that combines an object-role model's ease of use and an ontology's support for well reasoning. Riboni et al. [17] attempted the use of OWL 2 for modeling and reasoning of complex human activities. However, there are limitations on OWL due to the lack of support for temporal reasoning and newly named individual assertions [18]. Meditskos et al. addressed these problems by leveraging SPARQL rules over the OWL context representations [18]. In their SP-ACT framework, OWL (or OWL 2) is used to capture conceptual information about the activities, whereas SPARQL rules are used to correlate atomic activities together to form complex activities. Temporal relations are also captured by these rules. By using the `CONSTRUCT` and `WHERE` clauses where the former defines new relation that should be added to the ontology upon the successful pattern matching of the condition defined in the latter clause. Our framework shares similar use of SPARQL rules when deriving new context information.

The main differences between our work and others are: (i) An ontology model was defined for a real-world crowd-sensing smartphone platform (ii) a parser was developed to automatically map JSON structured data onto the ontology model (iii) SPARQL rules were created to derive new information about user behavior. Most importantly, this framework can be extended to include in-depth operations within a smartphone. Together with the new inferencing rules, this framework can certainly make PhoneLab user data more accessible to

researchers.

V. Conclusion and Future work

In this paper, we proposed the use of ontology models and SPARQL queries to facilitate management, processing and retrieval of mobile crowd-sensing data. More specifically, we defined ontology models for capturing PhoneLab's raw measurement data, constructed new models for representing the correlations between data and created SPARQL queries for deriving context information that is embedded in the models. We evaluated our proposal with three examples: (i) Inferring the temperature of a location by correlating battery temperature with location, (ii) Learning a user's charging pattern and its correlation with location, and (iii) Finding correlation between package changes and network usage. We demonstrated the feasibility of retrieving this information from the ontology models through simple queries. This is not possible if the measurement data is in the original JSON format without additional processing. We argue that the benefits of our proposal go beyond these examples when more and more information is collected. By extending the models with new measurement data types, we can use this framework to derive implicit context information about the user, which is otherwise hidden within the complex JSON data objects.

As an extension to this work, we will extend the ontology to include other implicit correlations and construct more queries to facilitate the discovery of user behavior. When appropriate, we will incorporate our ontology with other well-established ontologies such as the LinkedGeoData.org for location referencing.

Acknowledgment

The authors would like to thank the PhoneLab team at University of Buffalo for their support with data processing and for providing valuable feedbacks about the paper.

References

[1] Cisco, Inc., "Cisco visual networking index: Global mobile data traffic forecast update, 2014–2019," http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html, May 2015.

[2] G. Chen and D. Kotz, "A survey of context-aware mobile computing research," Hanover, NH, USA, Tech. Rep., 2000.

[3] J. yi Hong, E. ho Suh, and S.-J. Kim, "Context-aware systems: A literature review and classification," *Expert Systems with Applications*, vol. 36, no. 4, pp. 8509 – 8522, 2009.

[4] H. E. BYUN and K. CHEVERST, "Utilizing context history to provide dynamic adaptations," *Applied Artificial Intelligence*, vol. 18, no. 6, pp. 533–548, 2004.

[5] A. K. Dey, "Understanding and using context," *Personal Ubiquitous Comput.*, vol. 5, no. 1, pp. 4–7, Jan. 2001.

[6] A. Nandugudi, A. Maiti, T. Ki, F. Bulut, M. Demirbas, T. Kosar, C. Qiao, S. Y. Ko, and G. Challen, "PhoneLab: A large programmable smartphone testbed," in *Proceedings of SENSEMINE'13*. Roma, Italy: ACM, 2013, pp. 4:1–4:6.

[7] J. Shi, Z. Guan, C. Qiao, T. Melodia, D. Koutsonikolas, and G. Challen, "Crowdsourcing access network spectrum allocation using smartphones," in *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*, ser. HotNets-XIII. Los Angeles, CA, USA: ACM, 2014, pp. 17:1–17:7.

[8] J. Shi, L. Meng, A. Striegel, C. Qiao, D. Koutsonikolas, and G. Challen, "A walk on the client side: Monitoring enterprise wifi networks using smartphone channel scans," in *Proceedings of INFOCOM'16*, 2016.

[9] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, and D. Riboni, "A survey of context modelling and reasoning techniques," *Pervasive and Mobile Computing*, vol. 6, no. 2, pp. 161–180, 2010.

[10] P. Korpipää, J. Häkkilä, J. Kela, S. Ronkainen, and I. Känsälä, "Utilising context ontology in mobile device application personalisation," in *Proceedings of MUM '04*. College Park, Maryland, USA: ACM, 2004, pp. 133–140.

[11] S. Peroni and D. Shotton, "FaBiO and CiTO: Ontologies for describing bibliographic resources and citations," *Journal of Web Semantics*, vol. 17, pp. 33 – 43, 2012.

[12] 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," *Journal of Web Semantics*, vol. 17, pp. 25 – 32, 2012.

[13] T. Baker, S. Bechhofer, A. Isaac, A. Miles, G. Schreiber, and E. Summers, "Key choices in the design of simple knowledge organization system (SKOS)," *Journal of Web Semantics*, vol. 20, pp. 35 – 49, 2013.

[14] C. M. Keet, A. Ławrynowicz, C. d'Amato, A. Kalousis, P. Nguyen, R. Palma, R. Stevens, and M. Hilario, "The data mining OPtimization ontology," *Journal of Web Semantics*, vol. 32, pp. 43 – 53, 2015.

[15] X. Wang, D. Q. Zhang, T. Gu, and H. Pung, "Ontology based context modeling and reasoning using owl," in *Proceedings of PerCom'04 Workshop (CoMoRea)*, Orlando, Florida, March 2004, pp. 18–22.

[16] K. Henricksen, S. Livingstone, and J. Indulska, "Towards a hybrid approach to context modelling, reasoning and interoperation," in *the First international workshop on Advanced Context Modeling, Reasoning and Management*, Nottingham, England, 2004.

[17] D. Riboni and C. Bettini, "OWL 2 modeling and reasoning with complex human activities," *Pervasive and Mobile Computing*, vol. 7, no. 3, pp. 379–395, 2011.

[18] G. Meditskos, S. Dasiopoulou, V. Efstathiou, and I. Kompatsiaris, "SP-ACT: A hybrid framework for complex activity recognition combining OWL and SPARQL rules," in *Proceedings of PerCom'13 Workshop (CoMoRea)*, San Diego, California, March 2013, pp. 25–30.