

1st International Workshop on
Mobile Deployment of Semantic
Technologies (MoDeST 2015)



At the 14th International Semantic Web Conference
(ISWC 2015)
Bethlehem, PA, USA

October 2015

Preface

This volume contains the papers presented at the 1st Workshop on Mobile Deployment of Semantic Technologies (MoDeST2015) held on October 11th, 2015 in Bethlehem, Pennsylvania (USA) at the 14th International Semantic Web Conference (ISWC2015). MoDeST2015 aimed to bring together researchers, as well as developers and practitioners, who are interested in studying and deploying Semantic Web technology on mobile devices, as well as illustrating its useful deployment in particular application domains. More information about the workshop can be found at <http://modest.mobi>.

MoDeST2015 was a half-day workshop which included an invited talk by Jeff Z. Pan entitled “The Ubiquitous Semantic Web: The Story So Far” followed by the presentation of 4 full research papers and 2 short position papers. MoDeST2015 received 7 submissions from which the Program Committee selected 6 for presentation at the workshop and to be included in the proceedings.

We want to thank the members of the Program Committee for their help and support to make this first edition of MoDeST possible. We also thank the Deborah, Syed, and Eduardo for their help in the organization. We thank our invited speaker, Jeff Z. Pan, for his useful talk. We thank the organizers of the 14th International Semantic Web Conference for selecting MoDeST and hosting it at such important event. Finally, we thank all the participants of the workshop for their involvement and their contributions to the debate.

Evan Patton, William Van Woensel, and Roberto Yus

October 2015

Contents

1	The Ubiquitous Semantic Web: The Story So Far <i>Jeff Z. Pan</i>	1
2	Are Apps Going Semantic? A Systematic Review of Semantic Mobile Applications <i>R. Yus and P. Pappachan</i>	2
3	Building a Mobile Applications Knowledge Base for the Linked Data Cloud <i>P. Pappachan, R. Yus, P. K. Das, S. Mehrotra, T. Finin, and A. Joshi</i> .	14
4	Challenges for Semantic Technologies in Distributed Mobile Environments <i>E. W. Patton and A. Borgida</i>	26
5	Automating the Collection of Semantic Sensor Network Metadata in the Field with Mobile Applications <i>L. Kinkead, P. Pinheiro, and D. L. McGinness</i>	32
6	Linked Data and Mobile Application Privacy <i>E. W. Patton and I. Liccardi</i>	44
7	Mobile Semantic Query Distribution with Graph-Based Outsourcing of Subqueries <i>William Van Woensel</i>	50

MoDeST2015 Organization

MoDeST2015 Chairs

Evan Patton	Rensselaer Polytechnic Institute, USA
William Van Woensel	Dalhousie University, Canada
Roberto Yus	University of Zaragoza, Spain

MoDeST2015 Steering Committee

Deborah McGuinness	Rensselaer Polytechnic Institute, USA
Syed Sibte Raza Abidi	Dalhousie University, Canada
Eduardo Mena	University of Zaragoza, Spain

MoDeST2015 Program Committee

Sören Auer	University of Bonn, Germany
Carlos Bobed	University of Zaragoza, Spain
Fernando Bobillo	University of Zaragoza, Spain
Alex Borgida	Rutgers, USA
Sven Casteleyn	Universitat Jaume I, Spain
Mathieu d'Aquin	Open University, UK
Oscar Diaz	University of the Basque Country, Spain
Tim Finin	University of Maryland, Baltimore County, USA
Lalana Kagal	MIT CSAIL, USA
Yuan-Fang Li	Monash University, Australia
Jeff Z. Pan	University of Aberdeen, UK
Primal Pappachan	University of California, Irvine, USA
Michele Ruta	Politecnico di Bari, Italy
Oshani Seneviratne	Oracle, USA
Fuming Shih	Oracle, USA
Heiner Stuckenschmidt	University of Mannheim, Germany

The Ubiquitous Semantic Web: The Story So Far

Jeff Z. Pan

University of Aberdeen, UK
jeff.z.pan@abdn.ac.uk

Abstract of Invited Talk. Semantic Web technologies make it possible to represent, integrate, query and reason about structured online data. Recent years have witnessed tremendous growth of mobile computing, represented by the widespread adoption of smart phones and tablets. The versatility of such smart devices and the capabilities of semantic technologies form a great foundation for a ubiquitous Semantic Web that will contribute to further realising the true potential of both disciplines. In this talk, I will provide a brief overview of state-of-the-art research in this emerging area and try to conclude with a summary of challenges and important research problems.

Biography of the Speaker. Jeff Z. Pan received his Ph.D. in computer science from The University of Manchester in 2004, on the topic of Description Logics: Reasoning Support for the Semantic Web. He joined the faculty in the Department of Computing Science at University of Aberdeen in 2005. He is now the Deputy Director of Research of the department. His research focuses primarily on knowledge representation and reasoning, in particular scalable ontology reasoning, querying and reuse, and their applications (such as Semantic Web, Advertising, Healthcare, Software Engineering and Multimedia). He is a key contributor to the W3C OWL2 standard. He leads the work of the TrOWL Tractable OWL2 reasoning infrastructure. He is widely recognised for his work on scalable and efficient ontology reasoning; he gave tutorials on this topic in e.g. AAAI2010, ESWC2010, ESWC2011, SemTech2011 and the Reasoning Web Summer School (2010 and 2011).

Are Apps Going Semantic?

A Systematic Review of Semantic Mobile Applications

Roberto Yus¹ and Primal Pappachan²

¹ University of Zaragoza, Spain
ryus@unizar.es,

² University of California, Irvine, USA
primal@uci.edu

Abstract. With the wide-spread availability of cheap but powerful mobile devices and high-speed mobile Internet, we are witnessing an unprecedented growth in the number of mobile applications (*apps*). In this paper, we present a systematic review of mobile apps which use Semantic Web technologies. We analyzed more than 400 papers from proceedings of important conferences on Semantic Web and other venues. We give a brief overview of the 36 semantic mobile apps we identified by grouping them based on their specific functionalities. Our results show that usage of Semantic Web technologies on mobile devices is on the rise and there is a need for development of more tools to facilitate this growth.

Keywords: Semantic Web, semantic mobile applications, Android

1 Introduction

Mobile devices (such as smartphones and tablets) have been fast replacing other stationary devices as de facto medium for online browsing, social networking, and other applications. The widespread availability of high-speed mobile Internet and lowering prices of smartphones has accelerated this change. In fact, the most popular mobile application (*app*) stores crossed the one million apps mark in 2013. By using semantic technologies, applications on mobile devices can benefit from the advantages of the Semantic Web. For example, apps can use information from the Linked Data cloud, publish as well as subscribe to various data sources without worrying about app or device specific schemas, and reason over information to derive non-explicit facts.

The use of semantic technologies on mobile devices has been subject of interest from the early stages of the Semantic Web [1] and it has been recently invigorated with efforts to test and even port existing semantic technologies to mobile devices [2-5]. However, how many of the existing apps are using Semantic Web technologies is still unknown. Recently, Ermilov et al. performed a study on the field [6] by analyzing 172 relevant papers and coming up with guidelines for designing and developing effective semantic applications for ubiquitous devices.

However, the focus of their work was to find “the existing approaches for development of ubiquitous semantic applications” and thus slightly different from the goal of discovering how many semantic mobile apps have been presented.

In this paper we present a systematic review of semantic mobile applications which covers the breadth of semantic mobile apps and the depth of semantic data management. To this end, we analyzed more than 400 papers extracted from Google Scholar as well as proceedings of important Semantic Web conferences. From this set of studies we found 36 papers presenting semantic mobile apps for which we have extracted a brief summary with the focus on identifying “what” and “how” of semantic technology usage in these apps. We present information about common platforms and operating systems for mobile apps, Semantic Web technologies being used (locally on the device or in servers), and domains of the apps. We also outline some of the challenges and problems faced by researchers while developing semantic mobile apps. Our results show that number of semantic mobile apps has been steadily increasing over the years.

The rest of the paper is organized as follows. In Section 2, we present the methodology followed in performing this systematic review. In Section 3, we overview the semantic mobile apps we discovered grouped by specific domains. In Section 4, we answer the research questions proposed in our methodology. Finally, in Section 5, we discuss about findings of the review and outline future work.

2 Methodology

In this section we explain the methodology followed for the systematic review performed (based on the guidelines proposed in [7]).

Research Questions. The goal of this review is to find evidence to answer the following questions:

RQ1: How many semantic mobile applications have been developed?

RQ2: Is the increase on the number of mobile devices and their features motivating the development of more semantic mobile apps?

RQ3: Which domains are more suited for Semantic Web in mobile apps?

RQ4: What are the most used platforms and operating systems for semantic mobile apps?

RQ5: What are the most used Semantic Web technologies on mobile devices?

RQ6: Are Semantic Web technologies being used locally on mobile devices?

RQ7: What challenges/problems (specific to mobile devices) are researchers facing when developing semantic mobile apps?

Search Strategy and Study Selection Criteria. To find the studies to analyze we used the Google Scholar³ web search engine which indexes millions of research papers (and thus, includes documents from the major electronic libraries

³ <https://scholar.google.com>

–such as ACM, IEEE, and Springer–). We also went through the proceedings of the following Semantic Web conferences: the International Semantic Web Conference (ISWC) from 2002 to 2014 (last edition available), the Extended Semantic Web Conference (ESWC) from 2004 to 2015, and the International Conference on Semantic Systems (I-SEMANTICS) from 2009 to 2014. To query Google Scholar we used combinations of the following keywords: “Semantic Web”, “Mobile device”, “Android”, “SPARQL”, “RDF”, “OWL” and selected the first 400 results. Regarding the proceedings of the conferences we manually checked the title and abstract of each paper to select those focusing on mobile computing.

After this step, we manually went through each paper to check if there was a semantic mobile system described in it. We use the definition of ubiquitous semantic application as an app which is designed and developed for ubiquitous devices and uses semantic data in any way during its execution by Ermilov et al. [6]. We excluded papers which only presented an architecture or an ontology but did not describe an implementation of a semantic mobile app.

Data Extraction Strategy. For each paper remaining after the selection process we extracted the following information required to answer the research questions explained earlier: year, app functionality, mobile platform, local vs. remote handling of semantic data, Semantic Web technologies, and specific challenges related to development of the app.

3 Semantic Mobile Apps

In this section we give a brief overview of the 36 semantic mobile apps identified in the selection process. We have grouped the selected apps into disjoint domains. Therefore, even though many of the apps can be classified as Location-Based Services, we have chosen a more specific functionality for the purpose of this classification.

Map based and Augmented Reality. *mSpace Mobile* [1] provides information about topics of chosen interest to mobile users related to their location. The client app accesses the knowledge about topics and the location based information by calling remote web services. The server queries RDF KBs, using RDQL (RDF Data Query Language) [8], and returns the results to the client app. *DBpedia Mobile* [9] extracts information about POIs in the surroundings of the user from DBpedia [10] and displays them on a map. The client app, developed as a web app, displays the information obtained by a server which queries DBpedia using SPARQL. *PediaCloud* [11] displays tag clouds with information related to the user geographical location. The app executes SPARQL queries against the DBpedia endpoint to obtain POIs around the user and computes the tag cloud with the information retrieved. The app presented in [12] offers nearby POIs (e.g., cultural attractions) which match a user request or user interests. The app uses a semantic matchmaker (using non-standard reasoning services) on the device to match POIs (extracted from OpenStreetMap and enhanced semantically by a back-end) with given requests.

LOD4AR [13] displays POIs around the user by using augmented reality (AR). The client app, developed as a web application, obtains POIs as JSON by querying a server and displays it using an AR library. The server manages an OpenRDF Sesame RDFS store with information gathered from several sources (DBpedia, LinkedGeoData.org, Romanian Government Open Data portal). *Alive Cemeteries* [14] combines AR with Semantic Web to navigate through a cemetery in Hungary. The client app uses Androjena⁴ to handle the POIs returned by executing SPARQL queries against a knowledge base (it is not clear whether the KB is stored in the device or in a server). *ARSemantic* [15] offers personalized POIs using AR considering the user profile. The app uses a semantic reasoner (Mini-ME [16]) to infer POIs which might be interesting for a user regarding her profile. The reasoning tasks of subsumption, satisfiability, concept abduction, and concept contraction are used on the mobile device with enhanced OpenStreetMap data to do the matching between services and profile.

Disaster, Health Management and Collaborative. *WeReport* [17] allows people to report the situation during an emergency and relief workers to obtain continuously disaster feed. *Donate-N-Request* [17] (Android) matches requests for resources with their availability in the context of disaster scenarios. Both apps use a ported version of Jena to manage semantic data (reports or requests near the user) retrieved from an external server by executing SPARQL queries. Also, the app semantically annotates user generated reports and requests and sends them to the server. *VGSAndroidApp* [18] allows users to submit and browse volunteering requests. The client app uses the Triploid API, realized on top of Androjena, to parse the information returned by a RESTful web service. The web service executes SPARQL queries against the Jena triplestore in the server.

Patient Self-Management App [19] helps patients to develop self efficacy to overcome barriers for the self-management of cardiac risk factors. The app uses an OWL ontology to model the patient profile but it is not explicitly mentioned whether the app manages the OWL ontology directly or not. *Rafiki* [20] helps community health-workers in remote areas in the diagnosis of diseases. The app manages OWL ontologies on the device which are defined using the OWL API⁵. The app also uses SWRL rules and a DL reasoner (HermiT [21]) on the device to infer the most probable diseases for a patient given her symptoms and context.

ParkJam [22] helps users to find parking using crowdsourced geographic data (from external sources and users of the system). The client app gets the parking information from a back-end system on a server which integrates information from Linked Data sources (such as OpenStreetMap). *Urbanapoly* [23] uses Human Computation (minigames to engage users) to enrich and validate geo-spatial Linked Data (POIs). The client app obtains information about POIs around the user and the server validates information received from multiple users and publishes it. *FaceBlock* [24, 25] allows users to define their context-aware privacy policies regarding pictures taken by others (e.g., “do not allow strangers to take

⁴ <https://github.com/lencinhaus/androjena>

⁵ <http://owlapi.sourceforge.net>

my picture”) and implements it on devices around. The app, which has been tested on smartphones and Google Glass, uses the OWL API to handle OWL ontologies for the representation of the user context and SWRL rules for the definition of the policies. The app also uses a DL reasoner (JFact⁶) to infer the policy to be executed depending on the user context. *csxPOI* [26] enables the collaborative creation, sharing, and modification of semantic POIs. The client app receives POIs from the server, which stores them in a triplestore, and allows users to modify them or create new ones.

Semantic Web Browsers and Endpoints. *mSWB* [27] is an effort to develop an endpoint-agnostic mobile Semantic Web browser which can connect to any of the available endpoints to retrieve maximum information. The client app allows user to perform keyword based search and visualize the results in table and map view. The federated middleware, on an external server, takes care of running semantic queries in parallel on different endpoints and returning the results to the app. *OntoWiki Mobile* [28] is a mobile version of the free and open-source semantic wiki application “OntoWiki” which facilitates knowledge acquisition in a collaborative manner. The app is a HTML5 application which uses RDFauthor⁷ (Javascript-based system) for data authoring and utilizes HTML5 cache functionality to support offline work. Persistence of data is provided at the server which has advanced conflict resolution and replication features built-in which allows concurrent editing of same resources.

RDFBrowser [29] is an RDF browser which provides a generic layer for accessing device information making it independent of specific application schemas. The app uses Androjena to manage RDF data on the device and there is a remote RDF server which exposes the device information for outside world consumption. *RDF On the Go* [30] is an RDF storage and SPARQL query processor for Android devices allowing them to query data collected on the devices locally. They have adapted Jena and ARQ toolkits for mobile to handle RDF data and the app stores the triples from LinkedGeoData collection. The data is indexed using R-trees to support spatial SPARQL queries. The app presented in [31] makes it possible for mobile devices to publish the information from applications and sensors on them through a SPARQL endpoint. This data can be gathered by applications by querying the endpoints and federation through SPARQL queries. The app includes an RDF store and SPARQL endpoint based on the Sesame library (adapted to Android). *Linked Sensor Middleware* [32] provides wrappers for sensors on mobile device for the purpose of data collection and publishing. Through their web interface users can annotate and visualize the real world sensed data. They have also linked this sensor stream data to other Linked Data sources and the unified dataset can be queried through a SPARQL endpoint. *SHERLOCK* [33, 34] enables devices to automatically exchange knowledge about Location-Based Services in the geographic area of the user (e.g., a service to find taxis or to obtain pictures of monuments around). The

⁶ <http://jfact.sourceforge.net/>

⁷ <http://aksw.org/Projects/RDFauthor.html>

app manages OWL ontologies using the OWL API. Also, it uses a Description Logics reasoner on the device (JFact) to infer services which might be interesting for the user.

Social Networks and Recommendation. *Person Matcher* [35] obtains FOAF profiles from persons around the user (via Bluetooth) and calculates a “compatibility” score with the user for each profile discovered. The app handles RDF data using the MicroJena library⁸. *Mobile Social Semantic Web* [36] offers a distributed social network based on Semantic Web technologies. The app queries various triple stores (e.g., FOAF) and transforms the RDF, by using the Androjena, into a format that is suitable for social applications such as contact information based and FOAF based. *Who’s Who* [37] enables users to access and visualize Linked Data by linking physical world with virtual with the help of contextual information (e.g., location). To address the potential latency problems due to low bandwidth or no network connection, the app includes a light weight triple store on the device, using the RDFquery library⁹, which stores knowledge from the remote RDF server.

Mobile Wine agent [38] offers descriptions of wines and dishes, and recommendations to the user regarding her location. The app manages an ontology and supports partial reasoning on the device and exhaustive reasoning over the information collected is performed on the Jena server. *Cinemappy* [39] computes contextual movie recommendations for users by using their spatial and temporal position. The app executes SPARQL queries against the DBpedia endpoint to obtain information related to movies which is combined with information from semistructured sources. *Krishi-Mantra* [40] offers suggestions and alerts to farmers to improve productivity regarding the crops being cultivated. The client app sends information introduced by the user in forms to the server through RESTful web services and displays the results. The server translates the information to SPARQL and queries the KB with information about cotton. *RealFoodTrade* [41] allows farmers and fishermen to sell their products directly to the end-buyer. The client app sends user keywords (related to a particular type of fish) to the server which obtains a matching product in its ontology and finds announcements regarding fishermen selling it.

Travel. *GetThere* [42] provides users from rural areas with details about public transport (buses). The client app invokes web services which execute SPARQL queries against the dataset managed by the back-end. The server integrates information from Linked Data points with crowdsourced locations of buses shared by the clients. *LinkedQR* [43] enables users to scan QR codes attached to pieces of art in a museum to obtain further information. The client app creates and executes a SPARQL DESCRIBE query against the server using the URI contained in the QR code. The returned RDF is parsed by the app, using the Sesame library, and shown to the user. The server manages a KB with information about an art

⁸ http://poseidon.ws.dei.polimi.it/ca/?page_id=59

⁹ <https://code.google.com/p/rdfquery>

gallery enriched with information from DBpedia. *HDTourist* [44] helps tourists visiting a foreign city by displaying urban data from DBpedia. The app executes SPARQL queries against a local RDF/HDT file (which contains information extracted from DBpedia) using a Java library¹⁰. *Touristguide* [45] offers personalized tourist information to users after profiling them through questions. The client app obtains the information from a server which maintains the information about places in an ontology. *CURIOS* [46] offers personalized information to tourists based on their preferences and activity history. The client app uses RESTful services to obtain the information from the KB (generated by their previous system CURIOS CMS from Hebridean Connection dataset). Additionally the client also provides semantic (semantic relevance) and location-based (euclidean) caching to overcome connectivity issues. *Mobile Cultural Heritage Guide* [47] helps in finding interesting cultural material for a tourist using her location. The client app sends user information (location, heading, and facets) to the server which queries the KB (containing data from the Eculture data cloud, LinkedGeoData.org, and DBpedia, among others) and returns POIs.

4 Results

In this section, we answer the research questions posed in Section 2:

RQ1: How many semantic mobile applications have been developed? At least 36 semantic mobile apps have been presented in the literature based on our survey of publications over the last 10 years. Given that there is no central repository for authors to publish their semantic mobile apps, the process of finding them relies on quality of indexing mechanisms, appropriateness of keywords used in the search, and effectiveness of the selection of the studies. In our effort to avoid a possible bias, the studies were evenly split between the authors for the purpose of reviewing them.

RQ2: Is there an increase on the number of semantic mobile apps? Figure 1(a) shows the number of papers presenting a semantic mobile app per year (the figure do not include one app published in 2015¹¹). Notice that there is a gap between 2005 and 2009, we believe that this might be related to two milestones: the release of the iPhone in June 29, 2007, and the release of the first commercial version of Android in September 23, 2008. With the more powerful and affordable devices, high speed Internet, and better tools available, the number of mobile semantic web apps doubled in 2010 and 2014 whereas it remained stable in between.

¹⁰ <https://github.com/rdfhdt/hdt-java>

¹¹ This study has been finished in May 2015 so more semantic mobile apps might be presented in 2015.

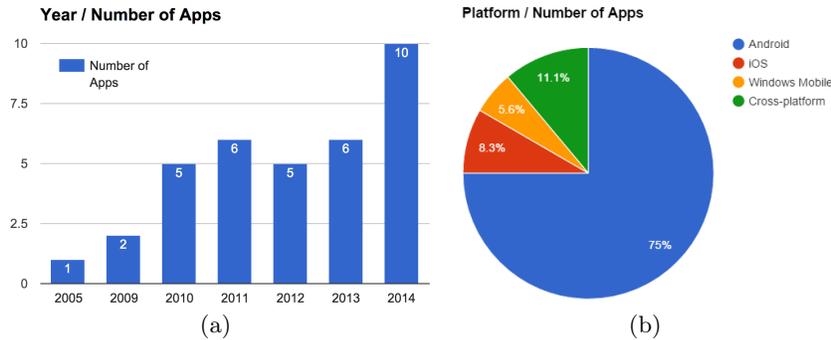


Fig. 1. Number of semantic mobile apps per year (a) and number of semantic mobile apps per platform (b).

RQ3: Which domains have the most number of Mobile Semantic Web apps? The majority of the apps reviewed, 27 apps out of 36 (i.e., [1, 9, 11–15, 17, 18, 20, 22, 23, 25, 26, 34, 35, 37–42, 44–47]), can be classified as Location-Based Services (LBSs). This was as expected as mobile devices are equipped with sensors which are able to obtain the location of the user in real-time. Among these LBS apps, the most common functionality is providing information about Points Of Interest (POI), 14 apps (i.e., [1, 9, 11–15, 23, 26, 37, 44–47]).

RQ4: What are the most used platforms and operating systems? In general all the apps are deployed on smartphones, except for [1, 35] which were deployed on Personal Digital Assistants (PDAs), as they were developed when PDAs were the most popular mobile devices. Figure 1(b) shows the distribution across different operating systems with Android being the most common choice for semantic mobile apps [11, 14, 15, 17, 18, 20, 22, 23, 25–27, 29–32, 34, 36, 37, 39–46] (27 out of 36). 3 of the apps were developed for iOS [19, 38, 47] whereas 2 are Windows Mobile apps [1, 35]. Also, there are 4 apps that have been developed as web applications [9, 12, 13, 28] and thus are cross-platform. The dominance of Android could be attributed to two factors: (1) it has the most number of users worldwide and it is based on Java as most of the popular semantic tools.

RQ5: What are the most used Semantic Web technologies? Figure 2 shows a wordcloud generated with the different semantic technologies that the apps reported using. For management of semantic data on the device, the most common libraries used are: Androjena (in [14, 18, 29, 36]), OWL API (in [20, 25, 34]), and Sesame API (in [31, 43]). Regarding Linked Data endpoints, apps use mainly the DBpedia (in [9, 11, 13, 39, 43, 44, 47]) and OpenStreetMap/LinkedGeoData (in [12, 13, 15, 22, 30, 47]) KBs. With regards to semantic reasoning, the following reasoners have been reported: Mini-Me (in [15]), JFact (in [25, 34]) and Hermit (in [21]).



Fig. 2. Wordcloud with the semantic technologies used by the different apps.

RQ6: Are Semantic Web technologies being used locally on the device? Most of the apps, 23 out of 36 (i.e., [1, 9, 13, 14, 17, 18, 22, 23, 26–28, 32, 37, 38, 40–47]), use a client-server approach in which the mobile app itself acted as an interface to present the results returned by the server. These type of client apps were also reported to be majority in [6] (where they were called “thin client apps”). However, 9 of these 23 apps (i.e., [14, 17, 18, 28, 37, 38, 43, 44]) processed Semantic Web languages on the device. 13 apps (i.e., [11, 12, 15, 19, 20, 25, 29–31, 34–36, 39]) do not follow the client-server approach and manage semantic data on the device (which can obtain from other devices or directly from Linked Data sources). Also, just 6 apps (i.e., [12, 15, 20, 25, 34]) use a semantic reasoner/matcher on the device to infer facts.

RQ7: What challenges/problems (specific to mobile devices) are researchers facing when developing semantic mobile apps? The challenges reported by apps can be broadly categorized into two: general Semantic Web challenges and those specific to mobile apps. In the first category we have long term issues such as helping users to define content in RDF or integrating information from different source. While the challenges specific to mobile apps include: the need for efficient mobile RDF triplestores [44], scalability issues from the perspective of storage and processing while increasing number of triples (e.g., [43] reported that increasing the number of triples in 20 multiplied the processing time almost by 4), lack of semantic reasoners for certain mobile operating systems (such as iOS), better local storage to minimize bandwidth and battery consumption for retrieving and processing semantic data [34, 46].

5 Conclusion and Next steps

The results of our systematic review, with more than 400 papers analyzed, show that mobile semantic apps have been presented since the early days of the Se-

semantic Web, 2005. However, it was not until 2009 that the number of mobile semantic apps started to steadily increase. The increment seen in 2014 motivates us to believe that in the following years this number will continue growing with the adoption of new mobile devices (such as smartwatches, smartglasses, and even “smart cars”). However, Semantic Web research should focus on dealing with the problems related to this new scenario (e.g., devices with limited capabilities which generate large amounts of highly-dynamic data) to popularize the use of semantic technologies in apps on these existing and future devices.

Our results show that most of the apps act as clients which rely on external servers for the handling of semantic data. This means that although they consume data which comes from Linked Data points and ontologies, this data is preprocessed on a server which returns the data in a semistructured format (JSON) or just as strings. However, this has been changing recently with few apps exploiting the capabilities of current mobile devices to handle semantic data locally. We believe that this trend would continue with work performed in porting existing semantic technologies (such as the Jena port Androjena, and semantic reasoners [2–4]) or creating new technologies specifically for mobile devices (e.g., semantic reasoners such as Mini-Me [16]).

Finally, based on the results and apps discovered in this review, we think it would be useful to formally define what a “semantic application” is, irrespective of whether it is mobile or not, by further studies. This would help in coming up with methodologies for systematic reviews and recommendations for semantic app development. The different scenarios presented in this paper, such as apps consuming data from a server in a non-standard format while the server obtains this information from the Linked Data cloud, apps handling data in a semantic format (i.e., RDF and OWL) on the device, or apps using a semantic reasoner to handle the data, would have to be studied to determine an specification of semantic apps.

In future, we want to extend this work by considering semantic mobile apps published in app stores (such as Google Play or Apple App Store). Based on a preliminary look at the Google Play Store¹² we found that number of commercial semantic mobile apps are indeed meager. Also, we are planning to build a website which can act as central repository of mobile semantic apps which would be updated periodically to keep track of all the latest apps.

Acknowledgments. This research work has been supported by RADICLE project CNS-1059436, CNS-1212943, CNS-1118127 and CNS-1450768, CICYT project TIN2013-46238-C4-4-R and DGA FSE.

References

- [1] Wilson, M.L., Russell, A., Smith, D.A., Owens, A., et al.: mSpace mobile: A mobile application for the Semantic Web. In: 2nd International Workshop on Interaction Design and the Semantic Web. (2005)
- [2] Patton, E.W., McGuinness, D.L.: A power consumption benchmark for reasoners on mobile devices. In: 13th International Semantic Web Conference (ISWC). (2014) 409–424

¹² <https://play.google.com/store>

- [3] Van Woensel, W., Haider, N.A., Roy, P.C., Ahmad, A.M., Abidi, S.S.R.: A comparison of mobile rule engines for reasoning on semantic web based health data. In: 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technologies (WI-IAT). (2014) 126–133
- [4] Yus, R., Bobed, C., Esteban, G., Bobillo, F., Mena, E.: Android goes semantic: DL reasoners on smartphones. In: 2nd International Workshop on OWL Reasoner Evaluation (ORE). (2013) 46–52
- [5] Valincius, E., Nguyen, H.H., Pan, J.Z.: A power consumption benchmark framework for ontology reasoning on Android devices. In: 4th International Workshop on OWL Reasoner Evaluation (ORE). (2015) 80–86
- [6] Ermilov, T., Khalili, A., Auer, S.: Ubiquitous semantic applications: A systematic literature review. *International Journal on Semantic Web Information Systems* **10**(1) (2014) 66–99
- [7] Kitchenham, B.: Procedures for performing systematic reviews. *Keele University* **33**(2004) (2004) 1–26
- [8] Seaborne, A.: RDQL a query language for RDF. W3C Member submission (2004)
- [9] Becker, C., Bizer, C.: Exploring the geospatial semantic web with DBpedia Mobile. *Journal of Web Semantics* **7**(4) (2009) 278–286
- [10] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.G.: DBpedia: A nucleus for a web of open data. In: 6th International Semantic Web Conference (ISWC). (2007) 722–735
- [11] Tessem, B., Johansen, B., Veres, C.: Mobile location-driven associative search in DBpedia with tag clouds. In: 9th International Conference on Semantic Systems (I-SEMANTICS). (2013) 6–10
- [12] Ruta, M., Scioscia, F., Ieva, S., Loseto, G., Sciascio, E.D.: Semantic annotation of OpenStreetMap points of interest for mobile discovery and navigation. In: 1st International Conference on Mobile Services (MS). (2012) 33–39
- [13] Vert, S., Dragulescu, B., Vasiu, R.: LOD4AR: exploring linked open data with a mobile augmented reality web application. In: 13th International Semantic Web Conference (ISWC). (2014) 185–188
- [14] Matuszka, T., Kiss, A.: Alive cemeteries with augmented reality and Semantic Web technologies. *International Journal of Computer, Information Science and Engineering* **8** (2014) 32–36
- [15] Ruta, M., Scioscia, F., Filippis, D.D., Ieva, S., Binetti, M., Sciascio, E.D.: A semantic-enhanced augmented reality tool for OpenStreetMap POI discovery. *Transportation Research Procedia* **3** (2014) 479 – 488
- [16] Ruta, M., Scioscia, F., Sciascio, E.D., Gramegna, F., Loseto, G.: Mini-ME: the mini match-making engine. In: 1st International Workshop on OWL Reasoner Evaluation (ORE). (2012)
- [17] Shih, F., Seneviratne, O., Liccardi, I., Patton, E., Meier, P., Castillo, C.: Democratizing mobile app development for disaster management. In: Joint Workshop on AI Problems and Approaches for Intelligent Environments and Workshop on Semantic Cities (AIIP). (2013) 39–42
- [18] Saveliev, A., Xu, S., Janowicz, K., Mülligann, C., Thatcher, J., Luo, W.: Volunteered geographic services: developing a linked data driven location-based service. In: 2011 International Workshop on Spatial Semantics and Ontologies (SSO). (2011) 25–31
- [19] Abidi, S.R., Abidi, S.S.R., Abusharek, A.: A Semantic Web based mobile framework for designing personalized patient self-management interventions. In: 1st Conference on Mobile and Information Technologies in Medicine. (2013)
- [20] Pappachan, P., Yus, R., Joshi, A., Finin, T.: Rafiki: A semantic and collaborative approach to community health-care in underserved areas. In: 10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom). (2014) 322–331
- [21] Shearer, R., Motik, B., Horrocks, I.: HermiT: A highly-efficient OWL reasoner. In: 5th OWLED Workshop on OWL: Experiences and Directions, collocated. (2008)
- [22] Kopecký, J., Domingue, J.: ParkJam: Crowdsourcing parking availability information with linked data. In: 9th Extended Semantic Web Conference (ESWC). (2012) 381–386
- [23] Celino, I., Cerizza, D., Contessa, S., Corubolo, M., Dell’Aglia, D., Valle, E.D., Fumeo, S.: Urbanopoly - A social and location-based game with a purpose to crowdsource your urban data. In: 2012 International Conference on Social Computing (SocialCom). (2012) 910–913
- [24] Pappachan, P., Yus, R., Das, P.K., Finin, T., Mena, E., Joshi, A.: A semantic context-aware privacy model for FaceBlock. In: 2nd International Workshop on Society, Privacy and the Semantic Web - Policy and Technology (PrivOn). (2014)
- [25] Yus, R., Pappachan, P., Das, P.K., Mena, E., Joshi, A., Finin, T.: FaceBlock: Privacy-aware pictures for Google Glass. In: 12th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys). (2014) 366–366
- [26] Braun, M., Scherp, A., Staab, S.: Collaborative semantic points of interests. In: 7th Extended Semantic Web Conference (ESWC). (2010) 365–369
- [27] Matuszka, T., Gombos, G., Kiss, A.: mSWB: Towards a mobile Semantic Web browser. In: 11th International Conference on Mobile Web Information Systems (MobiWIS). (2014) 165–175
- [28] Ermilov, T., Heino, N., Tramp, S., Auer, S.: OntoWiki Mobile - knowledge management in your pocket. In: 8th Extended Semantic Web Conference (ESWC). (2011) 185–199
- [29] David, J., Euzenat, J.: Linked data from your pocket: The Android RDFContentProvider. In: 9th International Semantic Web Conference (ISWC). (2010) 129–132

- [30] Phuoc, D.L., Parreira, J.X., Reynolds, V., Hauswirth, M.: RDF on the go: RDF storage and query processor for mobile devices. In: 9th International Semantic Web Conference (ISWC). (2010)
- [31] d'Aquin, M., Nikolov, A., Motta, E.: Building SPARQL-enabled applications with android devices. In: 10th International Semantic Web Conference (ISWC). (2011) 23–27
- [32] Le-Phuoc, D., Quoc, H.N.M., Parreira, J.X., Hauswirth, M.: The linked sensor middleware—connecting the real world and the semantic web. In: 10th International Semantic Web Conference (ISWC). (2011)
- [33] Yus, R., Mena, E., Ilarri, S., Illarramendi, A.: SHERLOCK: A system for location-based services in wireless environments using semantics. In: 22nd International World Wide Web Conference (WWW). (2013) 301–304
- [34] Yus, R., Mena, E., Ilarri, S., Illarramendi, A.: SHERLOCK: Semantic management of location-based services in wireless environments. *Pervasive and Mobile Computing* **15** (2014) 87–99
- [35] Van Woensel, W., Casteleyn, S., Troyer, O.D.: Applying Semantic Web technology in a mobile setting: The Person Matcher. In: 10th International Conference on Web Engineering (ICWE). (2010) 506–509
- [36] Tramp, S., Frischmuth, P., Arndt, N., Ermilov, T., Auer, S.: Weaving a distributed, semantic social network for mobile users. In: 8th Extended Semantic Web Conference (ESWC). (2011) 200–214
- [37] Cano, A.E., Dadzie, A., Hartmann, M.: *Who's Who* - A linked data visualisation tool for mobile environments. In: 8th Extended Semantic Web Conference (ESWC). (2011) 451–455
- [38] Patton, E.W., McGuinness, D.L.: The mobile wine agent: Pairing wine with the social Semantic Web. In: 2nd Workshop on Social Data on the Web. (2009)
- [39] Ostuni, V.C., Noia, T.D., Mirizzi, R., Romito, D., Sciascio, E.D.: Cinemappy: a context-aware mobile app for movie recommendations boosted by DBpedia. In: International Workshop on Semantic Technologies meet Recommender Systems & Big Data. (2012) 37–48
- [40] Kumar, V., Dave, V., Nagrani, R., Chaudhary, S., Bhise, M.: Crop cultivation information system on mobile devices. In: IEEE Global Humanitarian Technology Conference: South Asia Satellite (GHTC-SAS). (2013) 196–202
- [41] Cali, A., Virgilio, R.D., Noia, T.D., Menichetti, L., Mirizzi, R., Nardini, L., Ostuni, V.C., Rebecca, F., Ungania, M.: Semantic search in RealFoodTrade. In: 8th Alberto Mendelzon Workshop on Foundations of Data Management. (2014)
- [42] Corsar, D., Edwards, P., Baillie, C.C., Markovic, M., Papangelis, K., Nelson, J.D.: GetThere: A rural passenger information system utilising linked data & citizen sensing. In: 12th International Semantic Web Conference (ISWC). (2013) 85–88
- [43] Emaldi, M., Lázaro, J., Laiseca, X., López-de-Ipiña, D.: LinkedQR: Improving tourism experience through linked data and QR codes. In: 6th International Conference on Ubiquitous Computing and Ambient Intelligence (UCAmI). (2012) 371–378
- [44] Hervalejo, E., Martínez-Prieto, M.A., Fernández, J.D., Corcho, Ó.: HDTourist: Exploring urban data on Android. In: 13th International Semantic Web Conference (ISWC). (2014) 65–68
- [45] Dodwad, P.R., Lobo, L.: A context-aware recommender system using ontology based approach for travel applications. *International Journal of Advanced Engineering and Nano Technology* **1** (2014)
- [46] Nguyen, H.H., Beel, D.E., Webster, G., Mellish, C., Pan, J.Z., Wallace, C.: CURIOS mobile: Linked data exploitation for tourist mobile apps in rural areas. In: 4th Joint International Conference on Semantic Technology (JIST). (2014) 129–145
- [47] van Aart, C.J., Wielinga, B.J., van Hage, W.R.: Mobile cultural heritage guide: Location-aware semantic search. In: 17th International Conference on Knowledge Engineering and Management by the Masses (EKAW). (2010) 257–271

Building a Mobile Applications Knowledge Base for the Linked Data Cloud

Primal Pappachan¹, Roberto Yus², Prajit Kumar Das³,
Sharad Mehrotra¹, Tim Finin³, and Anupam Joshi³

¹ University of California, Irvine, USA
{primal, sharad}@uci.edu,

² University of Zaragoza, Zaragoza, Spain
ryus@unizar.es,

³ University of Maryland, Baltimore County, Baltimore, USA
{prajit1, finin, joshi}@umbc.edu

Abstract. The number of mobile applications (*apps*) in major app stores exceeded one million in 2013. While app stores provide a central point for storing app metadata, they often impose restrictions on the access to this information thus limiting the potential to develop tools to search, recommend, and analyze app information. A few projects have circumvented these limitations and managed to create a dataset with a substantial number of apps. However, accessing this information, especially for the purpose of an integrated view, is difficult as there is no common standard for publishing data. We present Mobipedia, an effort to gather this information from various sources and publish it as RDF Linked Data. We describe the status of Mobipedia, which currently has information on more than one million apps that has been extracted from a number of unstructured and semi-structured sources. This paper describes the ontology used to model information, the process for fact extraction, and an overview of applications facilitated by Mobipedia.

Keywords: Semantic Web, Linked Data, SPARQL, Knowledge Base, Android

1 Introduction

The incredible penetration of mobile devices (e.g., smartphones and tablets) in our lives in the last few years has been accompanied by an overwhelming growth in the number of mobile applications (also called *apps*) available for various platforms. The Google Play Store⁴ and the Apple App Store⁵, which are the main app stores currently, achieved the 1 million apps milestone in 2013, and, as of May 2015, both stores offer more than 1.4 million apps. Therefore, today's users have an array of choices while installing apps of any kind for entertainment, utility, or education. This has resulted in smart phones replacing other devices

⁴ <https://play.google.com/store>

⁵ <https://www.apple.com/itunes/charts>

as de facto medium for online browsing, social networking, and other activities, and mobile apps replacing traditional desktop applications and web sites.

Most of the popular app stores, which are used for showcasing as well as downloading apps, are proprietary and only offer a limited set of search functionalities. Some of them also restrict crawlers from downloading the metadata associated with apps and thus developers and researchers do not have access to this huge data set. While there have been industrial and academic efforts to gather information from app stores, the former is usually not freely available and the latter is difficult to access as they use different methods to release the datasets (e.g., websites, dumps, or databases) and various formats (from unstructured to semi-structured data). As a result a new project in this domain has to start either with a small data set of apps to analyze or repeat the process of gathering information which might be already available.

If developers and researchers are able to access information about apps easily, it would facilitate interesting analyses. For instance, the information could be used to warn users about malicious apps or apps that might request sensitive data. Also, such information would be useful to help users in the difficult task of selecting apps taking into account different parameters: from a purely technical one (such as the version of the operating system supported, the device requirements, or the installation size), to app credibility (such as ratings, privacy grade), among others. Therefore, we believe that having a centralized online knowledge base (KB) integrating information about mobile apps from various datasets and publishing it in an standard format would be very useful.

In this paper we present Mobipedia⁶, an evolving KB which integrates mobile app information from different sources and publishes it following the principles of Linked Data [1]. In its current version, Mobipedia contains metadata of around 1 million Android apps, including permissions and libraries used, extracted from two research projects and the official Android website. The information in Mobipedia, published in the standard Resource Description Framework (RDF) language, can be accessed through web browsers, programs, and query interfaces. To summarize, the major contributions presented in this paper are:

- Definition of an ontology to model app metadata information.
- Development of tools to extract facts from different sources and label the information semantically with the Mobipedia ontology.
- Creation of multiple access methods for Mobipedia data (Linked Data interface, SPARQL endpoint, and RDF dumps).

We present some example of interesting applications that can be developed using Mobipedia focused on the domains of app recommendation and privacy (based on our expertise). These applications can be developed agnostic of a specific mobile platform as we are using Semantic Web technologies and standard languages for representation. In addition, it is also possible to access the knowledge in Mobipedia locally on the device to draw inferences, for example, suggesting apps to their users depending on their context.

⁶ <http://mobipedia.link>

The rest of the paper is organized as follows. In Section 2 we present Mobipedia by explaining the ontology developed, the extraction of facts, and the accessing mechanisms. In Section 3 we show some motivating applications that can be developed using Mobipedia. Finally, in Section 4 we conclude and describe the future directions we are planning to take.

2 The Mobipedia Knowledge Base

The Mobipedia project is composed of an ontology which models concepts related to apps (see Section 2.1), an extraction module that generates facts from different sources (see Section 2.2), and three mechanisms to access the information stored (see Section 2.3). Figure 1 gives a high-level overview of the Mobipedia project including its different modules and external libraries used.

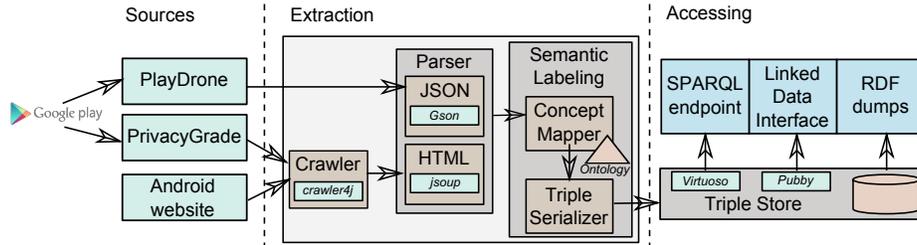


Fig. 1. Mobipedia overview.

2.1 Ontology

Mobile apps are programs designed to run on mobile devices (e.g., smartphones and tablets) and, as any other program, have different *versions* and use external *libraries*. Mobile devices are usually equipped with different sensors such as location, accelerometer, or gyroscope. Apps can make use of these sensors to, for example, offer personalized services depending on the user context. However, the information generated by the sensors on mobile devices could be sensitive (e.g., the location of the user). Mobile operating systems have a *permission* system which lets the user decide whether to grant an app request to access a specific sensor information.

In Mobipedia’s ontology⁷ we have modeled this information related to apps which is independent of the mobile operating system. We considered using existing ontologies such as Dublin Core Metadata Initiative (DCMI) and Description of a Project (DOAP) which are used to describe web resources and software projects respectively. But as neither of them is focused on mobile development,

⁷ <http://mobipedia.link/ont/mobipedia.owl>

the concepts and properties in those vocabularies do not match the requirements for modeling of mobile apps completely. Nevertheless, we linked some of the terms in DCMI ontology with Mobipedia terms using `owl:subClassOf`. For example, “DCMI:Creator” (<http://dublincore.org/documents/2012/06/14/dcmi-terms/?v=terms#terms-creator>) was linked to “Mobipedia:Developer” (<http://mobipedia.link/ontology/Developer>).

Figure 2 shows an excerpt of the ontology including the most important classes and the object properties that relate them⁸. The main classes in the ontology are: `mobipedia:App`, which represents mobile apps; `mobipedia:Version`, for the different versions of each app; `mobipedia:Permission`, for the permissions that can be requested by apps; `mobipedia:Library`, for the libraries imported by each version of an app; `mobipedia:Developer`, for the developers of each app; `mobipedia:Badge`, for badges assigned to developers; `mobipedia:Image`, for photos of the app or images of the badges; `mobipedia:Category`, for categories of apps, libraries, and permissions; and `mobipedia:App_Rating`, for the rating which users gave to the app. Table 1 shows some of the data properties in the Mobipedia ontology including a brief description, their domain, and range. In total, the current Mobipedia ontology includes 12 classes, 9 object properties, and 50 data properties.

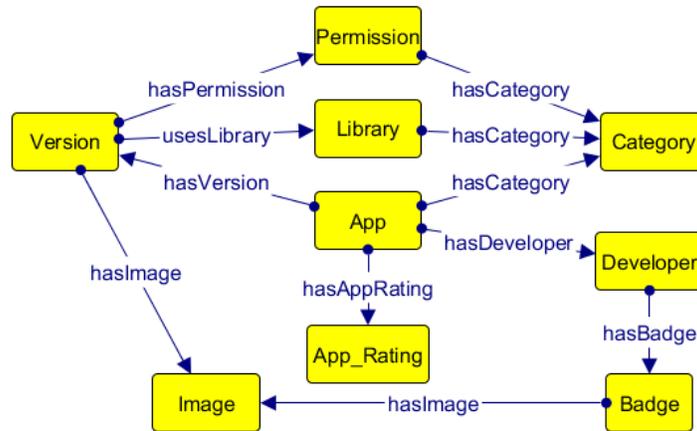


Fig. 2. Excerpt of the Mobipedia ontology.

2.2 Extraction Process

We focused on Android apps to populate the ontology and the current version of the Mobipedia KB by incorporating information from datasets released

⁸ The figure has been generated using the Graffoo specification [4].

Data Property	Description	Domain	Range
apk_url	URL to the APK	Version	xsd:anyURI
app_title	Name of an app	App	xsd:string
badge_title	Name of a developer badge	Badge	xsd:string
category_name	Name of a category	Category	xsd:string
comment_count	Number of comments for an app	App_Rating	xsd:string
description_html	Description of an app	App	xsd:string
developer_email	Email of a developer of an app	Developer	xsd:anyURI
developer_name	Name of a developer	Developer	xsd:string
developer_website	Website of a developer	Developer	xsd:anyURI
downloads	Number of downloads of an app	App	xsd:int
formatted_amount	Price of an app	App	xsd:string
image_url	URL of an image	Image	xsd:anyURI
installation_size	Size of an installed app	Version	xsd:int
library_description	Description of a library	Library	xsd:string
library_name	Name of a library	Library	xsd:string
major_version_number	Version number of an app	Version	xsd:int
package_name	Name of the package of an app	App	xsd:string
permission_description	Description of a permission	Permission	xsd:string
permission_name	Name of a permission	Permission	xsd:string
permission_reference	URL of the permission	Permission	xsd:anyURI
privacy_grade	PrivacyGrade of an app	Version	xsd:string
recent_changes_html	Change log of an app	Version	xsd:string
reviews_url	URL with the reviews of an app	App	xsd:anyURI
snapshot_date	Date when PlayDrone crawled the information of an app	App	xsd:dateTime
star_rating	User rating of an app	App	xsd:int

Table 1. Most important data properties in the Mobipedia ontology.

from two research projects as well as the Android website. Table 2 gives some statistics about these data sets. As the information in these sources was mainly unstructured or semi-structured (including HTML and JSON) we had to develop modules to extract structured information from these sources and label it with the Mobipedia ontology. As we had to filter out some of the information included in the dataset in PlayDrone which was not relevant for Mobipedia, we decided not to directly add a JSON-LD context. Instead, we developed ad hoc parsers to determine the datatypes of the data extracted as well as associate them with other entities in the dataset. For labeling the data we used the OWL API [6]. We have made crawlers and parsers available to help creating new tools for other data sources⁹. Having the parser developed, extracting the app metadata and converting it to RDF was easy.

PlayDrone. Google does not provide any mechanism to automatically extract information about the apps available in the Google Play store. Instead Google

⁹ <http://github.com/primalpop/MobipediaProject>

Android Permissions	Number of Permissions	152
PlayDrone	Number of Apps	1,402,894
	Number of Categories	24
PrivacyGrade	Number of third party libraries used	246
	Number of Apps	1,173,265

Table 2. Mobipedia Data Sources.

imposes restrictions to prevent the crawling of Google Play store data. However, researchers from Columbia University built the PlayDrone, a scalable Google Play store crawler which extracted information of over 1,100,000 apps [9]. The PlayDrone project publishes dumps with the information extracted for different dates¹⁰. The format used for the information published is JSON (see Figure 2.2 for an example). In our case we downloaded the latest dump available (October 31, 2014) and developed a parser to translate the JSON format, using the Gson library¹¹, to RDF.

```
{
  "app_id": "com.google.android.youtube",
  "title": "YouTube",
  "developer_name": "Google Inc.",
  "category": "MEDIA_AND_VIDEO",
  "free": true,
  "version_code": 51405300,
  "version_string": "5.14.5",
  "installation_size": 10191835,
  "downloads": 1000000000,
  "star_rating": 4.08009,
  "snapshot_date": "2014-10-31",
  "metadata_url": "https://archive.org/download/playdrone-metadata-2014-10-31-c9/com.google.android.youtube.json",
  "apk_url": "https://archive.org/download/playdrone-apk-c9/com.google.android.youtube-51405300.apk"
}
```

Fig. 3. Excerpt from PlayDrone dataset.

Android Permissions. We extracted information about the Android permission model from the website of the operating system¹² (see Figure 4). The website includes information about 152 official permissions that Android apps can request to access information from the user. For each permission the website contains its name (key), the code that has to be added to the manifest of the app (value), and a brief description. For extracting the content we developed an HTML parser using the jsoup library¹³.

¹⁰ <http://systems.cs.columbia.edu/projects/playdrone>

¹¹ <https://github.com/google/gson>

¹² <http://developer.android.com/reference/android/Manifest.permission.html>

¹³ <http://jsoup.org>

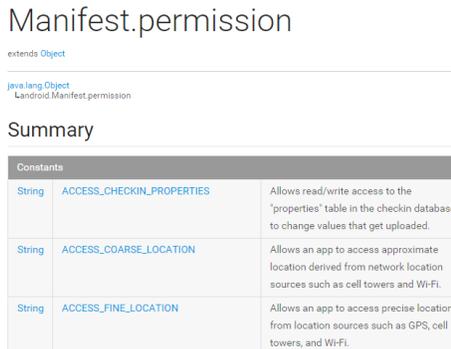


Fig. 4. Android permissions web site.

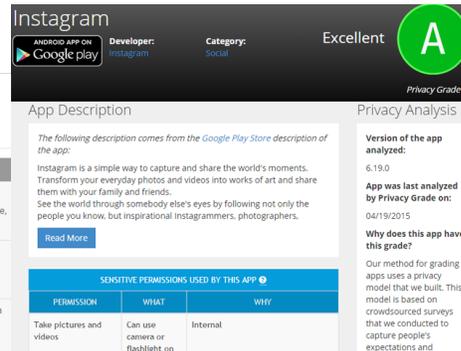


Fig. 5. PrivacyGrade for an app.

PrivacyGrade. A team of researchers from Carnegie Mellon University developed a method to grade Android apps based on the analysis of people’s expectations of an app’s behavior and app’s actual behavior [7]. For this purpose, they used static analysis of sensitive data usage by an app and crowdsourcing. At the end of analysis, apps are given a grade based on a 4.0 scale where “A+” means apps have no privacy concerns. The privacy grade for each app, along with other information such as the libraries used by the app, is published at their website¹⁴ (see Figure 5 for an example). To extract this information we developed a crawler based on the `crawler4j` library¹⁵ and parsed the HTML obtained. In addition to the Android permissions (152 in total) mentioned earlier, this dataset includes custom permissions (e.g., permission to access Facebook data) which are created by apps and used to restrict access to app data from other apps.

Linking Mobipedia with other Knowledge Bases. One of the requirements of Linked Data is to interlink the different knowledge bases available¹⁶. Based on this, we have interlinked Mobipedia with DBpedia [2], which is the nucleus of the Linked Data cloud. To achieve this, we executed queries against the DBpedia SPARQL endpoint to obtain entities which are already in Mobipedia. We analyzed the DBpedia category hierarchy and found two categories related to mobile apps. In this way, we obtained instances of the DBpedia categories *Android_(operating_system)_software* and *Mobile_software*, 409 and 221, respectively. After filtering for duplicates, we checked the names of the 600 remaining DBpedia entities against entities in Mobipedia and linked them by using the `owl:sameAs` property (for each entity we automatically obtained a list of possible links based on the name and manually selected the most appropriate ones).

¹⁴ <http://privacygrade.org>

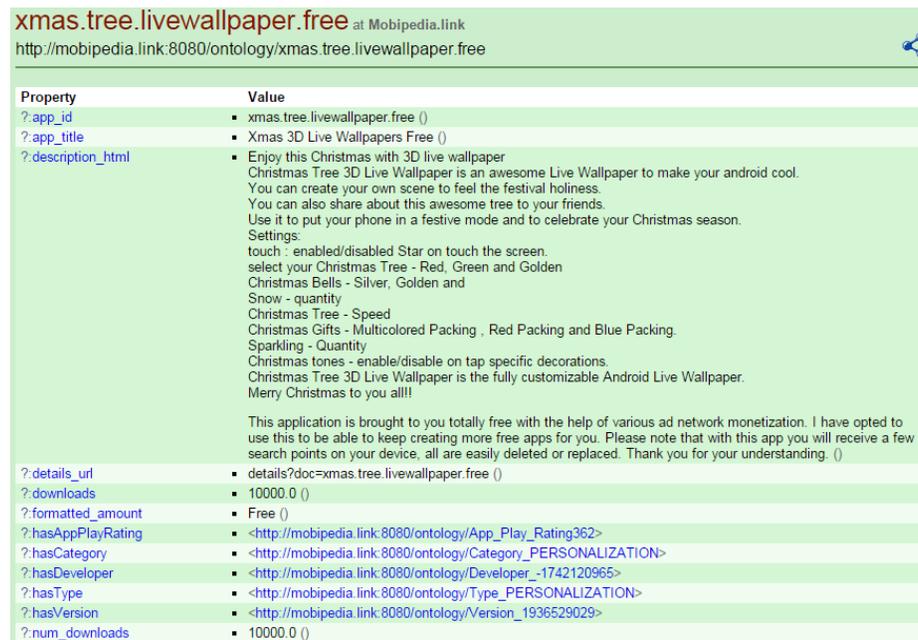
¹⁵ <https://github.com/yasserg/crawler4j>

¹⁶ <http://www.w3.org/DesignIssues/LinkedData.html>

2.3 Accessing Mobipedia

Access to Mobipedia is royalty-free under the terms of GNU free documentation license. Similarly to DBpedia [2], we provide three mechanism of accessing the Mobipedia dataset:

Linked Data. It uses HTTP protocol to retrieve entity information which contains all the triples associated with the entity. This can be accessed using web browsers, Semantic Web browsers, and crawlers. We generated the Linked Data interface for the SPARQL endpoint by using the Pubby project¹⁷. Pubby is a Java web application which translates URIs which are not dereferenceable to dereferenceable URIs by connecting to the SPARQL endpoint. For example, Figure 6 shows the web page created by Pubby for one of the entities in the Mobipedia KB (an app called “Xmas 3D Live Wallpapers Free”¹⁸).



Property	Value
?app_id	▪ xmas.tree.livewallpaper.free ()
?app_title	▪ Xmas 3D Live Wallpapers Free ()
?description_html	▪ Enjoy this Christmas with 3D live wallpaper Christmas Tree 3D Live Wallpaper is an awesome Live Wallpaper to make your android cool. You can create your own scene to feel the festival holiness. You can also share about this awesome tree to your friends. Use it to put your phone in a festive mode and to celebrate your Christmas season. Settings: touch - enabled/disabled Star on touch the screen. select your Christmas Tree - Red, Green and Golden Christmas Bells - Silver, Golden and Snow - quantity Christmas Tree - Speed Christmas Gifts - Multicolored Packing , Red Packing and Blue Packing. Sparkling - Quantity Christmas tones - enable/disable on tap specific decorations. Christmas Tree 3D Live Wallpaper is the fully customizable Android Live Wallpaper. Merry Christmas to you all!! This application is brought to you totally free with the help of various ad network monetization. I have opted to use this to be able to keep creating more free apps for you. Please note that with this app you will receive a few search points on your device, all are easily deleted or replaced. Thank you for your understanding. ()
?details_url	▪ details?doc=xmas.tree.livewallpaper.free ()
?downloads	▪ 10000.0 ()
?formatted_amount	▪ Free ()
?hasAppPlayRating	▪ <http://mobipedia.link:8080/ontology/App_Play_Rating362>
?hasCategory	▪ <http://mobipedia.link:8080/ontology/Category_PERSONALIZATION>
?hasDeveloper	▪ <http://mobipedia.link:8080/ontology/Developer_-1742120965>
?hasType	▪ <http://mobipedia.link:8080/ontology/Type_PERSONALIZATION>
?hasVersion	▪ <http://mobipedia.link:8080/ontology/Version_1936529029>
?num_downloads	▪ 10000.0 ()

Fig. 6. Linked Data interface of Mobipedia as seen in a web browser.

SPARQL Endpoint. We have also setup a SPARQL endpoint at <http://mobipedia.link/sparql> which can be used for querying the Mobipedia dataset. This endpoint is hosted using open source edition of Virtuoso server¹⁹.

¹⁷ <http://wifo5-03.informatik.uni-mannheim.de/pubby>

¹⁸ <http://mobipedia.link:8080/ontology/xmas.tree.livewallpaper.free>

¹⁹ <https://github.com/openlink/virtuoso-opensource>

RDF Dumps. Larger versions of the Mobipedia dataset in the form of serialized triples can be downloaded from the Mobipedia website as well. These dumps can be used as annotated datasets in research or for the purpose of running various analyses locally.

3 Mobipedia Application Usecases

In this section we present some of the applications that can benefit from Mobipedia's data and its standard format for representing it. As Mobipedia provides an easy access point for various app data, it could be useful for facilitating app development as well as mobile computing research.

3.1 For Application Development

As today's app stores are densely populated its not an easy task for either users to find the right app or the developers to compare their app with what's out there already. We propose different applications to tackle this problem.

Semantic Search. With Linked Data in Mobipedia, it is possible to perform a semantic search making it easier to find the right set of apps. In the case of users, currently they are limited to keyword search in order to find the application they are looking for. This makes it difficult to find applications for a unique set of users (e.g., superhero games with parental control) or applications with special features (e.g., todo list with location reminder) or an application which does not collect unnecessary user data (e.g., the flash light app requesting the less number of permissions). Developers have only the category information provided by app stores to discover apps which are similar to theirs and. They have to rely on external agencies or organizations (e.g., 42matters²⁰) to have detailed information about app markets. A semantic search engine for apps could be developed, for example as a website, translating the user/developer queries to SPARQL and executing them against the Mobipedia endpoint. For example, Figure 7 shows the SPARQL query needed to extract the list of flash light apps and the number of permissions they request.

App Recommendation. Mobipedia can also facilitate development of an app recommendation system. By using simple distance measures and user history of apps installed and rating, an app recommender could suggest which one of the app should be installed for a particular requirement based on comparison with apps with similar properties in Mobipedia. Also the user context could be used to infer apps that might be interesting for her. For example, a user visiting a new country could be presented with tourist apps for such a place. For that, it would be possible to develop a semantic mobile app which uses a reasoner locally on the device [10], an ontology defining the user context, and Mobipedia.

²⁰ <https://42matters.com/api>

```

PREFIX mobipedia: <http://mobipedia.link/ontology/>

SELECT DISTINCT ?App, (COUNT (?permission) AS ?numPerm)
WHERE {
  ?App mobipedia:description_html ?Desc.
  FILTER(contains(?Desc,"flashlight")).
  ?App mobipedia:hasVersion ?Vers.
  ?Vers mobipedia:hasPermission ?permission
}
GROUP BY ?App
ORDER BY ASC(?numPerm)

```

Fig. 7. An example of SPARQL query to return the list of flashlight apps and their number of permissions requested.

Permission Suggestion. Similar to the app recommender, we can also develop a permission chooser which helps developers to make an informed decision about the permissions to be used in the app so that it would be compliant with regulations and user expectations. While building an app, a developer can verify what permissions and external libraries data are typically used by other apps in the similar category by using Mobipedia and make an informed decision on what permissions/data to ask from the user. We hope that this would result in evolution of a privacy guideline for developing apps of different categories and not adhering to it would mean that app would be ranked lower in the suggestions given by the previously mentioned app recommender.

3.2 For Research

Data from apps would be useful in many domains of mobile computing research for privacy studies, application recommendation, and various statistical analyses. For this purpose, researchers would have to comb through websites and papers to find out datasets (if publicly available) and develop tools for data transformation. With Mobipedia, researchers can focus on building their system without worrying about accessing data from different sources as well as linking them to real world entities. Similar to DBpedia Query builder, we intend to provide various such examples of ready-made SPARQL query snippets for accessing various kinds of information from Mobipedia in the form of a query builder. It will also consist of queries written by Semantic Web researchers thus making it easier for developers to find information inside Mobipedia without the prerequisite of learning SPARQL.

Linking application user experiences. The information of user experiences while using an application is fragmented across various sources such as app ratings, reviews, blog articles, forums and so on. We intend to develop a mobile app library which developers could embed in their applications for capturing various user experiences in a systematic manner. These user experiences data can be pushed to Mobipedia and linked to app information which we already have in

the KB and further the relevance of recommendations possible in the sample applications given above.

Mining app reviews. In app stores, the user feedback is captured using reviews which can help developers to improve user satisfaction. Previous work [3] has been done in mining these reviews and visualizing them. By adding more classes to Mobipedia, we can directly link the concepts from app reviews to apps itself thus capturing the user sentiment in the knowledge base.

Policy Representation. Semantic Web technologies have been used significantly in context-aware systems for security purposes [8]. Rule languages such as Semantic Web Rule Language (SWRL) has been used to represent policies which capture user preferences on sensitive data access to context-aware services or apps. The Mobipedia ontology can be used to represent concepts related to apps. Also, researchers can leverage various meta-data (e.g., privacy grade, developer rating, permissions requested, etc.) about apps in the KB to augment their context-aware policies.

4 Discussion and Next Steps

Mobipedia is an effort to store information related to mobile apps from multiple sources and present it in a structured format accessible by humans and machines. In the current version of Mobipedia we focused on Android apps and incorporated three important data sources which contain information for more than 1 million apps: PlayDrone, PrivacyGrade, and the Android permissions website. We enabled three mechanisms to access the information in Mobipedia: Linked Data interface, SPARQL endpoint, and RDF dumps. Therefore, users can access Mobipedia from web browsers, query interfaces, or their own applications. We also interlinked the KB with DBpedia to fulfill the good practices for publishing Linked Data²¹. This allows users to access information about apps which is not available in DBpedia (e.g., Android permissions requested by an app). Finally, we have shown several applications that can benefit from using the content in Mobipedia and the standard representation format used (RDF). Until now, developing such applications would require an effort to find and integrate the information which is split in different sources and published with different formats. Some of the applications are mobile semantic apps which would benefit from accessing the knowledge in the Mobipedia KB locally.

Mobipedia is an evolving project due to the dynamic nature of mobile apps: New apps or versions of existing apps are published every day. The next steps of the project involve the integration of other published data sets such as the *Android Malware Genome Project* [11], which contains information about malware apps in the Android Play store, and the *BlueSeal* project [5], which analyzed the flow of malicious apps. Second, we want to incorporate data from other app stores such as the Amazon.com or GetJar and possibly app stores with different

²¹ <http://www.w3.org/TR/ld-bp>

permission model such as Apple App store. We are also hoping over time we would be able to incorporate apps portals in other languages (e.g., Baidu store, Tencent App Gem in China) as well. Currently contributions to Mobipedia can be only approved by the developers. However, to tackle ever growing app stores community participation would be essential. We are hoping to open for community contributions. For this to be a reality, we need to develop mechanisms to vet the quality of information submitted as well as make it easy to contribute without relying on users knowledge of RDF and SPARQL.

Acknowledgments. This research work has been supported by RADICLE project CNS-1059436, CNS-1212943, CNS-1118127 and CNS-1450768, CICYT project TIN2013-46238-C4-4-R and DGA FSE, U.S. National Science Foundation awards 0910838 and 1228198.

References

1. Bizer, C., Heath, T., Berners-Lee, T.: Linked data-the story so far. *Semantic Services, Interoperability and Web Applications: Emerging Concepts* pp. 205–227 (2009)
2. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: DBpedia - a crystallization point for the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web* 7(3), 154–165 (2009)
3. Chen, N., Lin, J., Hoi, S.C.H., Xiao, X., Zhang, B.: Ar-miner: Mining informative reviews for developers from mobile app marketplace. In: *36th International Conference on Software Engineering (ICSE)*. pp. 767–778 (2014)
4. Falco, R., Gangemi, A., Peroni, S., Shotton, D., Vitali, F.: Modelling OWL ontologies with Graffoo. In: *11th Extended Semantic Web Conference (ESWC)*. pp. 320–325 (2014)
5. Holavanalli, S., Manuel, D., Nanjundaswamy, V., Rosenberg, B., Shen, F., Ko, S., Ziarek, L.: Flow permissions for Android. In: *2013 IEEE/ACM 28th International Conference on Automated Software Engineering (ASE)*. pp. 652–657 (2013)
6. Horridge, M., Bechhofer, S.: The OWL API: A java API for OWL ontologies. *Semantic Web* 2(1), 11–21 (2011)
7. Lin, J., Liu, B., Sadeh, N., Hong, J.I.: Modeling users’ mobile app privacy preferences: Restoring usability in a sea of permission settings. In: *Symposium On Usable Privacy and Security (SOUPS)*. pp. 199–212 (2014)
8. Truong, H.L., Dustdar, S.: A survey on context-aware web service systems. *International Journal of Web Information Systems* 5(1), 5–31 (2009)
9. Viennot, N., Garcia, E., Nieh, J.: A measurement study of Google Play. In: *The 2014 ACM International Conference on Measurement and Modeling of Computer Systems*. pp. 221–233. SIGMETRICS (2014)
10. Yus, R., Bobed, C., Esteban, G., Bobillo, F., Mena, E.: Android goes semantic: DL reasoners on smartphones. In: *2nd International Workshop on OWL Reasoner Evaluation (ORE)*. pp. 46–52 (2013)
11. Zhou, Y., Jiang, X.: Dissecting android malware: Characterization and evolution. In: *2012 IEEE Symposium on Security and Privacy*. pp. 95–109 (2012)

Challenges for Semantic Technologies in Distributed Mobile Environments

Evan W. Patton¹ and Alexander Borgida²

¹ Dept. of Computer Science, Rensselaer Polytechnic Institute, Troy, NY USA

² Dept. of Computer Science, Rutgers University, Piscataway, NJ USA

Abstract. We present a scenario for a mobile wine recommendation agent where semantic technologies can provide value to end users. The wine recommender uses an ontology in the $SHOIN(d)$ description logic, and is extended to consider energy consumption of the device, intensional social contexts, and reasoning in a privacy-aware manner. We discuss the challenges semantic technologies, especially ones based on Description Logics, face in this scenario and formulate key issues for the mobile semantic technology community to address.

Keywords: reasoning, mobile, resource constraints, distributed, description logics

1 Introduction

Recent work has demonstrated the feasibility of reasoning on mobile phones [12,13], but resource constraints lead to poor performance of description logics (DLs). Due to space constraints, we will focus discussion on the profiles of OWL 2 (i.e., EL, QL, and RL) rather than RDF(S), and consider mostly conjunctive query answering rather than standard reasoning such as subsumption, given that many mobile applications are data/instance oriented. Specifically, we present a scenario of a mobile wine recommender system based on the $SHOIN(d)$ description logic. We highlight some challenges presented when combining traditional description logics with social structures and distribution in resource constrained devices.

2 Contextual peer reasoning in a Recommender System

Consider a mobile wine agent (e.g., [7]), which is a semantic recommendation engine. The user enters a restaurant as part of a group, and the agent, running on the phone of the user, is tasked with recommending wines to go with the meal(s) chosen by the group. The agent uses its GPS together with search engines to identify the restaurant and retrieve its menu and wine list. In addition, it obtains other information about the restaurant from social media (e.g., reviews). The agent has access to a global knowledge base of foods and wines in an expressive DL (e.g., $SHOIN(d)$), with recommended pairings. The agent

also has access to a personal knowledge base consisting of the likes/dislikes and past experiences of the specific user. This personal knowledge base also contains private information concerning the health status of the user, which could affect the choice of wines. Note that it is not necessary for the recommender agent to return *all* possible wine pairings for the desired food, or even the *best* one; a wine that is available locally is sufficient. Due to the use of description logics, which can describe concepts intensionally not just extensionally (i.e., by enumeration), it is possible for the agent to return an answer that *describes* appropriate wines, such as “a light red wine from Spain,” which could be communicated to the waiter or sommelier in the absence of machine-readable wine descriptions or in case no instances on the wine list match the given class expression. Intensional information also allows the agent to be told via DLs things about the group, like the fact that they are celebrating someone’s birthday (the identity of the person need not be known), or an important visitor/superior is present, in which case the user’s personal preferences should be discounted. The incomplete information that can be modeled by DLs is useful, for example, if the GPS information is not sufficient to identify just one restaurant (GPS resolution is insufficient in case the restaurants are located above one another or in dense urban areas with limited access to open sky). Note therefore that recommendation is very contextual (based on both the location and the group eating together). Throughout, the system will need to balance the benefit of generating recommendations with the amount of energy used to do so.

3 Discussion

First, let us observe some of the key features of this recommender system:

1. The recommender need not give the best answer, nor must it give complete answers as patrons cannot try every wine/food pairing combination.
2. Recommendation context comes by both intensional and extensional means. For example, GPS or other geolocation techniques can extensionally describe user location. Individuals attending the meal can be described by concept expressions, e.g. $\exists hasAttendee.(clientOf.\{user1\})$.
3. Private data may not be transmittable due to non-technological (e.g., legal) reasons. For example, health data (e.g., allergies) used to make a recommendation cannot be transmitted under the HIPAA law in the United States. This information can only be used in a distributed inference scenario.
4. The agent’s distributed query answering can exploit advances in both distributed/mobile databases as well as techniques from federated SPARQL. Further, using information about classes and role hierarchies, as well as least common subsumers in DL, we could gain further knowledge about which nodes are likely to have information relevant to answering a query.
5. Personal knowledge bases at each node evolve to be “egocentric” as the user makes their preferences known. This provides an inherent locality that can be exploited (e.g., user 2’s device does not know user 1’s preferences, so there is little need to query user 2’s knowledge base vis-à-vis user 1).

6. Since queries can be answered in an intensional manner (i.e., by class expressions) under open world semantics, answers are not restricted to concrete instances and thus allow for more flexibility in query answering compared with closed world approaches such as mobile databases.

We now briefly discuss some challenges related to the aforementioned features and highlight some related work for the interested reader.

Challenge 1: Pervasive energy awareness. Due to significant resource requirements, mobile semantic technologies will need to be aware of available space and energy for computation and communication. Predictive models of energy use will enhance query cost estimation and planning. Intelligent migration of computation will also be useful for balancing computation with user requirements to maintain privacy or reserve sufficient energy in case of emergencies. Early work has been done by [8,11] to measure power consumed by semantic technologies but this area is still underexplored.

Challenge 2: Query, data, and knowledge migration. During query evaluation the engine may generate one or more states relevant to other nodes. [5] explores using ontology metrics to select an appropriate reasoner at runtime to minimize T-box classification time. If the cost (e.g., time, energy) of computing these states is greater than the cost to transmit, sharing them with other nodes could save resources. Another case for migration is when the device is at a point where computing a solution will require more than its available energy, but sending a portion of its T-box or A-box (e.g., sending most specific concepts [1] or summarizing [2,3]) might allow other nodes to answer sufficiently well. Energy limits and privacy constraints may be defined by the user (e.g., *always ensure that I have at least 30 minutes of battery life left*). [9] gives a sound algorithm for query answering in an \mathcal{EL} KB with secrecy.

Challenge 3: Query, data, and knowledge partitioning. In a highly distributed world, partitioning semantic queries so that nodes only receive portions appropriate to the local knowledge base, especially under security and privacy constraints, will be challenging. Traditional approaches have included publishing link-sets between nodes or making statistics about the underlying RDF data available as a part of a SPARQL service description [4]. However, publishing such information may expose users to data-harvesting by malicious agents. Knowledge and data may be naturally partitioned/modularized based on mobile application design, such as wine preferences for a user stored on a device versus living in the cloud. What are alternate techniques for planning semantic queries over partitioned knowledge bases and can we take advantage of the semantics with respect to the advances made by the mobile database community?

Challenge 4: Knowledge summarization and compression. Sharing knowledge in a mobile environment costs time and energy, especially if network coverage is

spotty or a high speed, low power network such as WiFi is unavailable. Minimizing the energy required to share knowledge between nodes will improve our ability to parallelize inferences and reach consensus across many nodes.

Challenge 5: Optimizing tableaux concurrency as nodes enter/exit the network. Techniques to parallelize tableaux reasoning include having different nodes classify different concepts or explore non-deterministic choices introduced by disjunction and maximum cardinality constructs in parallel.

Challenge 6: Reasoning in the face of contradictory information in a dynamic network. Paraconsistent logics provide alternatives to the traditional description logics used in the semantic web. Due to their ability to avoid deducing everything in the presence of contradictory information, they are useful for reasoning over large datasets curated by large groups (e.g., crowdsourced data about disasters [6]) starting from different base assumptions about the world. In our mobile scenarios, clearly different people may have different preferences – something that would have to be modeled in a careful manner, possibly using beliefs.

Challenge 7: Resolving updates that are locally consistent but globally inconsistent. In a similar vein to challenge 6, updates consistent at a mobile node may not be globally consistent considering all nodes. While this is not specifically a challenge to mobile deployments of semantic technologies per se, one must consider the cost of performing this task from an energy perspective.

Challenge 8: Anytime/approximate reasoning for low energy. Another approach to limited energy resources is heuristics. If we can predict that an operation is going to require too much energy to compute (but not too much energy to communicate), the node could ask a peer to perform the calculation on its behalf. Approximate reasoning through less expressive DL reasoners will also increase throughput on mobile platforms when complete answers are not required.

Challenge 9: Dynamic cost functions based on context changes. Intensional descriptions of context brought by knowledge representation techniques may help reduce costs. For example, rather than having to have a user identify all of their clients, one could use class expressions to infer them. This can reduce the cost to transmit and store this information where memory and communication costs are at a premium. If we extend this further to location-based context, the savings of not enabling the GPS or WiFi to obtain user location data will improve battery duration over the course of the user’s day.

Challenge 10: Data Replication/Caching. As we have already mentioned, caching is an important technique in linked data. [14] provides a model of data replication of a knowledge base using contextual information. However, such approaches often do not include a cost model with respect to the device’s energy reservoir. It is possible that a sufficiently large enough cache would require so much energy to download, that not enough energy would remain to use it effectively.

Table 1. Feature summary of mobile, distributed knowledge bases and the challenges they present to reasoner designers, knowledge engineers, and application developers.

Feature	Challenges
Security & Privacy	<ul style="list-style-type: none"> • Preserving user privacy when performing distributed reasoning and query answering [9] • Sharing generalized class expressions to hide user preferences
Query processing & planning	<ul style="list-style-type: none"> • Exploit class expressions, relations to determine relevant endpoints
Distributed inference	<ul style="list-style-type: none"> • Limited resources prevent “pulling all data” for inferences
Preserving/forgetting history	<ul style="list-style-type: none"> • Context changes may require frequent truth maintenance; how much can be deferred?
Unreliable connections	<ul style="list-style-type: none"> • Truth maintenance when a node leaves the network
Summarization	<ul style="list-style-type: none"> • Reduce T-box or A-box size for limited resources [1,2,3] • Cost to summarize versus compress and transmit knowledge base elsewhere
Energy-awareness	<ul style="list-style-type: none"> • Different reasoning techniques result in different rates of energy consumption [8,11] • Cost to transmit portions of the T-box, A-box
Concurrency	<ul style="list-style-type: none"> • Reuse of partial models when answering multiple queries simultaneously
Replication	<ul style="list-style-type: none"> • Intermediate joins to reduce redundancy [10] • Semantics-driven data replication and caching [14]
Updates	<ul style="list-style-type: none"> • Simultaneous updates on devices may be locally consistent, but not globally consistent

4 Summary

Table 1 presents a summary of different capabilities mobile knowledge base systems might include for various application domains. We encourage the community to look at these challenges through the lens of four different resources: 1) *Memory*, which can be reused if free, otherwise it is limited by the amount of heap provided by the operating system; 2) *Time*, which can be reduced by parallelization locally (multicore) or remotely (radio); 3) *Energy*, which is monotonically decreasing until user connects phone to a power source; and 4) *User Attention*, whereby if the application is unresponsive for a period of time the user will lose interest.

This leads us to a number of open questions that may be of interest to the research community. What are techniques for efficiently migrating data to one or more nodes in the event the current node is going to run out of energy? How can we partition queries for distributed evaluation when reasoning over

private information? How can semantic descriptions be used to improve caching techniques between nodes? Can we use knowledge in addition to statistics to better partition data and query parts than has been accomplished in the mobile database community?

References

1. Baader, F.: Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In: Proc. 18th IJCAI. pp. 319–324. Morgan Kaufmann Publishers Inc. (2003)
2. Dolby, J., Fokoue, A., Kalyanpur, A., Kershenbaum, A., Schonberg, E., Srinivas, K., Ma, L.: Scalable semantic retrieval through summarization and refinement. In: Proc. 22nd AAAI Conf. vol. 7, pp. 299–304 (2007)
3. Fokoue, A., Kershenbaum, A., Ma, L., Schonberg, E., Srinivas, K.: The summary abox: Cutting ontologies down to size. In: Proc. 5th Int. Semantic Web Conf., pp. 343–356. Springer (2006)
4. Görlitz, O., Staab, S.: Splendid: Sparql endpoint federation exploiting void descriptions. COLID 782 (2011)
5. Kang, Y.B., Krishnaswamy, S., Li, Y.F.: A meta-reasoner to rule them all. In: Proc. 23rd Int. Conf. on Information and Knowledge Management. pp. 1935–1938. ACM, New York, NY, USA (2014)
6. Li, W., Adebayo, J., Shih, F., Kagal, L.: The role of mobile technologies in humanitarian relief. In: Proc. 12th Int. Conf. on Inf. Syst. for Crisis Response and Management (2015)
7. Patton, E.W., McGuinness, D.L.: The mobile wine agent: Pairing wine with the social semantic web. In: Proc. 2nd SDOW Workshop (2009)
8. Patton, E.W., McGuinness, D.L.: A power consumption benchmark for reasoners on mobile devices. In: Proc. 13th Int. Semantic Web Conf. (2014)
9. Tao, J., Slutzki, G., Honavar, V.: Secrecy-preserving query answering for instance checking in EL. Tech. rep., Iowa State University (2010)
10. Urbani, J., Kotoulas, S., Maassen, J., Van Harmelen, F., Bal, H.: Webpie: A web-scale parallel inference engine using mapreduce. *J. Web Semantics* 10, 59–75 (2012)
11. Valincius, E., Nguyen, H., Pan, J.Z.: A power consumption benchmark framework for ontology reasoning on android devices. In: Proc. 4th ORE (2015)
12. Van Woensel, W., Haider, N.A., Ahmad, A., Abidi, S.S.R.: A cross-platform benchmark framework for mobile semantic web reasoning engines. In: Proc. 13th Int. Semantic Web Conf. pp. 389–408 (2014)
13. Yus, R., Bobillo, F., Bobed, C., Mena, E.: The OWL reasoner evaluation goes mobile. In: Proc. 4th ORE (2015)
14. Zander, S., Schandl, B.: Context-driven rdf data replication on mobile devices. *Semantic Web Journal Special Issue on Real-time and Ubiquitous Social Semantics* 3(2), 131–155 (2012)

Automating the Collection of Semantic Sensor Network Metadata in the Field with Mobile Applications

Laura Kinkead¹, Paulo Pinheiro¹, Deborah L. McGuinness¹

Rensselaer Polytechnic Institute, Troy NY 12180, USA

Abstract. In the past few decades, the field of ecology has grown from a collection of disparate researchers who collected data on their local phenomenon by hand, to large ecosystems-oriented projects partially fueled by automated sensor networks and a diversity of models and experiments. These modern projects rely on sharing and integrating data to answer questions of increasing scale and complexity. Interpreting and sharing the big data sets generated by these projects relies on information about how the data was collected and what the data is about, typically stored as metadata. Metadata ensures that the data can be interpreted and shared accurately and efficiently. Traditional paper-based metadata collection methods are slow, error-prone, and non-standardized, making data sharing difficult and inefficient. Semantic technologies offer opportunities for better data management in ecology, but also may pose a challenging learning curve to already busy researchers. This paper presents a mobile application for recording semantic metadata about sensor network deployments and experimental settings in real time, in the field, and without expecting prior knowledge of semantics from the users. This application enables more efficient and less error-prone in-situ metadata collection, and generates structured and shareable metadata.

1 Introduction

Over the past few decades, the field of ecology has expanded immensely in scope. The field now tackles scientific questions that address wide ranges of complexity, scale, and time spans [12], [14]. Some of these questions are tackled by large project groups, and others are answered by combining data sources from multiple disparate projects. Despite the essential need for combining data from multiple heterogeneous sources, it is evident that there is no systematic approach for ecologists to share data and data workflows that takes into consideration the ecologist's in-situ knowledge about observations and experiments. More specifically, for projects utilizing sensor networks, there is no systematic approach to capture the ecologist's in-situ knowledge about sensor deployments and configurations such that the knowledge can inform data analysis and management decisions. The lack of contextual knowledge about in-situ scientific activities affects the way data should be further interpreted and analyzed. For example, the data generated by an improperly calibrated instrument or an instrument

not correctly placed on its platform should not be used in rigorous statistical analysis. Similarly, configuration settings of sensor network equipment may be changed in the field, which affects the resulting data product and the manner in which it should be analyzed.

We introduce a semantic approach to mediate data usage and sharing by encoding the knowledge about the sensor network that generated the data. In particular, we encode knowledge about the deployments of platforms, instruments, and detectors in support of the data products they generate. With this kind of knowledge, ecologists can better understand the context of their data collection. This information allows scientists unfamiliar with the original collection to make appropriate use of the data [12]. Thus, the availability of semantic annotations encoded as metadata increases the length of time that a data product is useful, as the data no longer relies on the presence of the people who were involved in the data collection process to explain how the data came to be. Whereas a file without metadata may become useless once its originator changes jobs or forgets some details, a file with metadata may be useful for many decades [12], [5]. Extending the longevity of data is especially important as ecological projects look to answer questions spanning long periods of time. In addition, metadata allows a data product to be used in the future to answer unanticipated questions [11]. In this paper, we describe a semantic technology-based mobile application that enables on-site capture of metadata in real time. By embedding the metadata capture system in a mobile device, much of the metadata capture can be completed automatically. The application uses a QR code-based sensor identification system, which automates the identification of sensor equipment and minimizes errors in data entry.

Our work overcomes three major challenges. First, our framework is a solution for overcoming the barrier of entry to semantic technologies for field scientists. This tool makes the collection of standardized and machine readable metadata efficient for the practicing scientist, thus enabling the practicing scientist to benefit from the advantages of semantic technologies without having any prior knowledge of the technology. Second, it describes and implements a framework for a method of using semantic technologies on a mobile platform to record data in real time in the field. This framework makes metadata capture more efficient and less error-prone compared to traditional recording methods. In addition, this method description and implementation has the potential to be broadly reused by a wide range of observational efforts. Finally, this application addresses context-specific challenges to make it more likely for valuable in-situ knowledge to be captured.

2 Challenges of Collecting In-Situ Contextual Knowledge

2.1 Challenges in Collecting Contextual Knowledge

The onset of automation in ecological data collection means that metadata about in-situ contextual knowledge, including the knowledge about the sensor network collecting the ecological data, is more important than ever before. A scientist

needs metadata not only for remembering how to use a data set that was collected years ago, but also for simply understanding how to use the data that was collected by an autonomous sensor earlier the same day. Automated sensor equipment collects data at an unprecedented rate, and it is necessary to have a well-structured system for capturing all of the contextual knowledge surrounding each data collection activity so that the data can be accurately used. Metadata is also critical for sharing data sets between research groups who are unfamiliar with the details of each others' work. For these reasons, metadata is widely recognized as a critical component to ecological data management [12], [5], [11].

Semantic technologies can be and are being used to support improved knowledge sharing: these technologies provide a standardized framework for storing and sharing metadata, and they make data interpretation and use more efficient [14]. However, creating semantic metadata is typically a slow and tedious process for a human to do manually. Generating semantic data often requires much technical knowledge, and the final product must be correct and complete if it is to be used effectively by applications. While semantic technologies ease the workload of the data consumer by improving integration and understandability, they are often perceived as a burden by the data generator. These difficulties present a large barrier to entry to many scientists, and many find that the challenges associated with adopting this new technology are not worth the benefits. These problems present a very real barrier for semantic technology adoption in non-computer science fields.

While very valuable, collecting well-structured and comprehensive contextual metadata can be expensive. At one point in time, requiring a scientist to take thirty minutes to create a thorough metadata document was thought to be worth the future value of that document [5]. However, the onset of large, automated data-collection systems that generate data sets by the minute renders such a value judgment unreasonable; such a researcher would be overwhelmed by the rate of data collected by the network [12].

2.2 The Additional Challenges of In-Situ Knowledge Collection

Much of the most valuable contextual knowledge needs to be collected in-situ: while a field scientist is actively collecting samples or making changes to automated equipment. Contextual knowledge such as the GPS points at which a sample was collected, the serial number of instrument was used to collect a sample, or what program was selected for an on-board computer is best written down immediately to ensure the information is accurate and to ensure that the knowledge is recorded at all.

Thus, while metadata recorded in the lab faces an efficiency problem at the level of requiring time of a researcher who is sitting at his desk with competing activities to do, collecting in-situ contextual knowledge has even more challenges. The field is not an ideal place to record data: flat, dry surfaces are difficult to come by, and a field scientist is often in a rush to complete a set of tasks not just before lunch break, but before the sun goes down or before it rains. Traditionally, field scientists have solved this problem by collecting metadata very quickly

with pen and paper, without always applying consistency or completeness to the process. This strategy leads to situations such as the one experienced by our group recently in which we received sixteen boxes of field notes containing handwritten metadata for a thirty year observational study. In its handwritten form, this data is challenging to utilize effectively. Thus, we need a compromise between efficiency for the knowledge recorder and usability for the knowledge receiver.

Fortunately, creating shareable, machine readable data is a forte of machines. The emergence of powerful yet affordable handheld devices, namely mobile phones, presents an opportunity for more efficient and less error-prone data collection in-situ. Such tools can validate input to ensure that metadata adheres to the structure of selected metadata standards. In addition, rather than recording one’s own metadata in the field on paper and then later trying to fit that data into a standard (and perhaps losing information in the transition), semantic tools help scientists collect the right information in real time and on site. These software tools may also include error-checking mechanisms, such as checking for out-of-bounds values, to ensure that collected metadata is sensible and to highlight mistakes for immediate correction [6].

3 Ontology-driven Contextual Knowledge Capture

This work is performed in the context of the Jefferson Project, a collective effort of Rensselaer Polytechnic Institute, IBM, and The FUND for Lake George. The Jefferson Project studies Lake George in New York state as a model ecosystem and aims to apply the findings to freshwater resource management worldwide [10]. One component of the Jefferson Project is a large network of sensors stationed around the lake and its watershed. These sensors collect data on a variety of attributes, such as the region’s weather, the lake’s chemistry and currents, and even populations at the lowest levels of the food web. The state and arrangement of these sensors will change many times over the course of the project: instruments will need to be taken back to the lab for calibration, fixed when malfunctioning, upgraded, or may be deployed elsewhere. Throughout all of this activity, the metadata about the position of all of the sensor equipment will need to be recorded so that the data sets the sensors generate can be correctly interpreted. Therefore, we decided to focus our metadata capture application on collecting metadata about sensor deployments because this metadata will be collected many times over the course of the project, thus it will benefit greatly from being automated.

3.1 The Human Aware Sensor Network Ontology

We use the Human Aware Sensor Network Ontology [13], or HASNetO, to encode our metadata. By using a semantic approach, we make the interpretation of the metadata less subject to misleading interpretations, and make it possible for machines to read and leverage the knowledge in the process of managing the data.

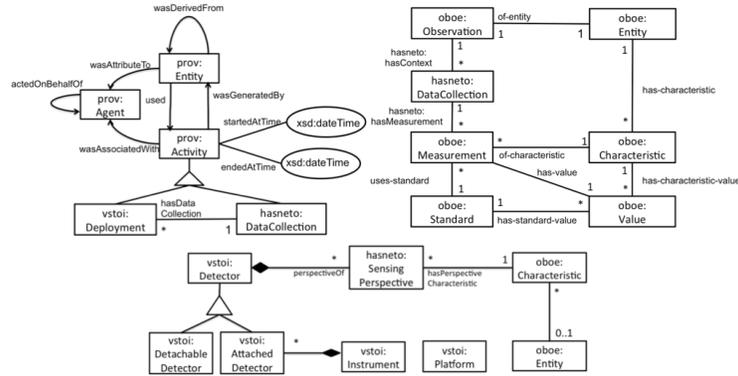


Fig. 1. The Human Aware Sensor Network Ontology (HASNetO) [13]

In related work, concepts from the Extensible Observation Ontology (OBOE) [8], the PROV Ontology (PROV-O) [7], and the Virtual Solar Terrestrial Observatory Ontology (VSTO) [9] were leveraged to build the Human-Aware Sensor Network Ontology, or HASNetO [13]. Notably, we did not incorporate the Semantic Sensor Network Ontology (SSN) into HASNetO because although SSN uses similar concepts, it is not suited to our work because it does not talk about human agents and their involvement in the process of managing sensor networks. HASNetO groups sensor network equipment into three types: detectors, instruments, and platforms. *Detectors* are the objects that do the sensing: they convert the physical signals about the characteristic of interest into a (most often electric) signal that can be read by a computer or human. *Instruments* are the objects that support the detectors. They do not do any sensing themselves, but they provide the framework in which the detector captures signals, and convert the detector’s signal into a data point. A *platform* is the object that determines the location of the instrument, whether it be the point of a stationary platform or the path of a mobile one. A platform may also provide overhead services, such as providing the instrument with power, a data connection, and protection against natural and human hazards. In HASNetO, a *deployment* is composed of one platform, one instrument, and one or more detectors. A deployment also has a start time and an end time.

3.2 MOCCASN: Mobile Context Capture for Sensor Networks

Our solution for collecting in-situ contextual knowledge is MOCCASN, an Android application. MOCCASN makes collecting metadata very quick, and it does not require the scientist using the application to have any knowledge of the underlying semantic technology. Using the phone’s camera, the MOCCASN identifies sensor network objects via QR codes that are affixed to each object. An object’s QR code contains the URI of the object’s instance of a HASNetO concept. From the URI, one can retrieve instance properties such as the serial

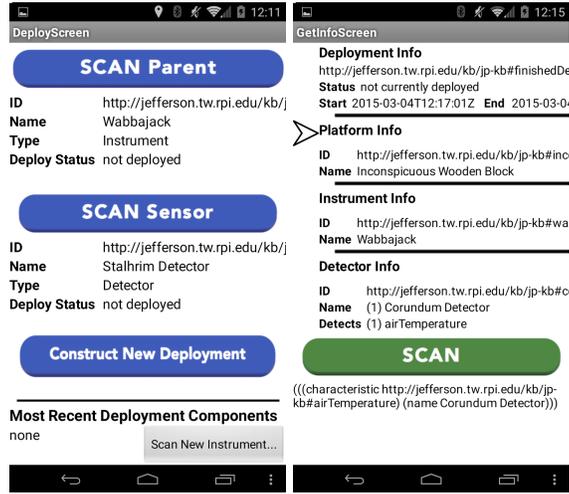


Fig. 2. Constructing a new deployment and getting information about a deployment

number and type of sensing device. The scientist can use MOCCASN to scan an object’s QR code and retrieve information about the object, to start a deployment with the object, or to end the object’s current deployment. When starting and ending deployments, MOCCASN uses the phone’s built-in GPS to automatically assign latitude and longitude coordinates. Time and user log-in information are automatically documented as well. All of this information is automatically recorded in accordance with the HASNetO ontology. The scientist is guided through the process of creating and ending deployments via a dynamic graphical user interface. MOCCASN accomplishes these tasks by communicating with the online knowledge base through a cellular or wifi connection. When the scientist does not have a data connection, records may be saved locally to the phone for later submission. While in communication with the knowledge base, MOCCASN initiates error checks on the scientist’s input based on the semantics encoded in the ontology.

The GitHub page for MOCCASN may be found at [1], and a static demo page at [3].

3.3 Software Development Tools

MOCCASN was developed with MIT App Inventor, an online tool with an intuitive interface for developing Android applications quickly and easily [2]. MIT App Inventor made it possible for MOCCASN to be developed in a matter of weeks, and we believe that MIT App Inventor can be used to rapidly create similar specialized applications to match the needs of a variety of observation-based efforts.

4 Results

MOCCASN meets a number of requirements of in-situ metadata collection, including deployment-specific requirements. First, deployments must be able to be assembled in a piecemeal fashion. For example, the user may want to connect the instrument-to-be-deployed with its associated detectors while still on shore, and then connect the instrument on the platform at a later time. We use an existing term in the OBOE ontology, “hasCode”, to “tag” partial deployments in the knowledge base as “under construction”. With this method, a scientist may connect an instrument and detector in the field and enter this partial deployment in the knowledge base as a “deployment under construction”. When that instrument is later brought into the field to be connected to a platform, the tool automatically finds that the instrument is part of a “deployment under construction”, and adds the platform to the same deployment. When the user starts the deployment, the “under construction” tag is removed from the deployment. This method ensures that all of the application users and the knowledge base always have the same information.

Second, a data or wifi connection cannot be relied upon. It is common for field sites to be located in areas with no data connection. Therefore, all data must be recorded in a sensible fashion even when the scientist is unable to connect to the knowledge base. MOCCASN meets this requirement by offering a “Save Deployment Data” button, which copies the recorded information to local storage, such as the scanned URIs, the username, and the time. The scientist can view this saved information in the “View Un-Submitted Records” screen, and the scientist may re-submit any of these records when he or she returns to cell service by clicking on the record and clicking “Re-Submit”. Upon resubmission, the application will initiate all necessary error-checking routines and attempt to write the data to the knowledge base.

Reasoning in support of error checking is performed when annotated data arrives in the data repository. Reasoning capabilities are limited on the device itself as much of the error checking involves comparing the scanned information against information currently in the data repository. Therefore, when MOCCASN does not have a connection to the knowledge base, the application cannot execute error-checking on the recorded information in real time. Thus, it is possible that locally stored information may contain information that is inconsistent with the information in the knowledge base. These inconsistencies may only be identified when a data connection is regained, and will prevent the data record from being submitted. However, this information is likely still valuable even though it is not completely valid; portions of the data record are likely accurate. To make use of this partially correct information, the scientist is offered the option of emailing the record so that the data record can be corrected and entered manually. This feature ensures that no data that was collected in the field is ever lost.

In addition, by allowing the user to save deployment information, the application is more robust to errors caused by the knowledge base becoming out-of-sync with the true state of the sensor network. For example, consider a situation where a user attempts to end the deployment of an object, but upon scanning

the object, the application finds that the object is not currently deployed. This may happen because the scientist who set up the deployment forgot to click the “Start Deployment” button. The current scientist is correct to want to enter the deployment end time information, but the application will not let the scientist do so due to inconsistent information in the knowledge base. The missing deployment start information must be entered into the knowledge base by hand before the app will enter the deployment end information. To accommodate this type of situation, the user may save the end deployment information locally, and re-submit it once the error has been solved manually.

We now introduce a number of use cases for the ways in which MOCASSN has been used to read and write metadata while in the field.

4.1 Use Case: Constructing a Deployment

To construct a deployment, the scientist scans two directly sensor objects that are to be directly attached to each other. For example, a scientist may scan an instrument and one of its detectors, or a platform and its instrument. Every time the scientist scans an object, the application queries the knowledge base to check that the object is in the knowledge base, that it is either a type of instrument, detector, or platform, and that it is not currently deployed (an object must be un-deployed before it may be re-deployed). When a pair of objects is scanned, the application queries to check that the types of the two objects are compatible (an instrument must have a parent of type platform, and a detector must have a parent of type instrument). It also checks that either one of the objects or neither of the objects is part of a deployment under construction.

According to HASNetO, a deployment is comprised of one platform, one instrument, one or more detectors, a start time, an end time, a location, and the deploying scientist. The platform, instrument, and detector objects of the deployment are assembled in a piecemeal fashion. During the time when the deployment is incomplete because all of the pieces of information have not yet been submitted, the deployment is tagged with the code “under construction” to denote that the deployment is not consistent with the HASNetO definition of a deployment. After the remaining deployment information is recorded, the scientist clicks “Start Deployment” to add the end time and remove the “under construction” tag.

4.2 Use Case: Ending a Deployment

To end an ongoing deployment, the scientist scans any of the objects associated with the deployment, and the application adds an end time to the deployment. The application will not allow the scientists to end a deployment that is not currently underway.

4.3 Use Case: Getting Information about a Sensor

The application may also be used to retrieve information about any piece of sensor network equipment. For scientists with no prior knowledge of linked data

and the SPARQL language that is used to query linked data, this feature offers a simple solution for viewing deployment data in the knowledge base in a very readable format. This screen presents a number of details about the most recent (or current) deployment of the scanned object. For example, this screen displays the deployment’s associated platform, instrument, and detectors, as well as the characteristics that are detected. In addition, the screen will show the deployment’s start and end times if applicable.

4.4 Reflections on Field Testing

MOCCASN was field tested with a Jefferson Project field scientist. This scientist has been performing ecological research on Lake George for many years, and is part of the team deploying Jefferson Project sensor network equipment. Testing was performed with an HTC One M7 phone. After reviewing the structure of the application, the scientist started and ended about a dozen deployments. A few mistakes were made at first, such as forgetting to click “Start Deployment” after connecting all pieces of the deployment, and not waiting for a confirmation of data submission before moving on to make the next entry. These problems could be mediated by explicitly warning the scientist of the incomplete data entry when they attempt to move on too quickly. We estimate that it takes about fifteen minutes to introduce how to use the application, and about an hour of practice for the field scientist to get a good understanding of how to use MOCCASN proficiently.

5 Discussion

MOCCASN addresses issues related to barriers to entry, automatic metadata capture, and in-situ context capture. The application removes the barrier-to-entry that researchers often face when presented with semantic technology solutions. With this tool, a researcher can generate machine readable metadata without any prior knowledge of semantics.

Historically, progress on improving the way we model metadata knowledge has come at the cost of increased time spent capturing metadata and increasingly advanced formats. For a long time, the extra time required of the data generator to standardize data was well worth the extra effort because it enabled efficient data sharing. However, with the onset of automated data collection, researchers will simply be overwhelmed by the amount of metadata that needs to be generated. Our work developing MOCCASN counters the trend of increased capture time for the sake of data usability, reducing a researcher’s metadata capture to a few QR code scans.

MOCCASN enables in-situ capture of contextual knowledge that would be lost otherwise. Researcher field time is incredibly valuable, and automated tools enable field scientists to quickly record valuable knowledge that would otherwise have been recorded on paper or not recorded at all in an easily usable and shareable format.

By automating metadata generation, we make a number of other advances in metadata capture. The application performs error-checking in real time by communicating with the data repository to help prevent erroneous data from being entered into the knowledge base. In addition, all metadata created with this tool conform to a standard vocabulary and are immediately accessible by anyone else on the team.

5.1 Value of Semantics

By automating the capture of contextual knowledge, we enable field scientists with no technical knowledge of semantic technologies to benefit from the value of semantic technologies. While the structure and capture of contextual knowledge is often standardized within a lab, it is not common for such metadata to follow broad community standards. This makes sharing datasets very difficult and time consuming, as a human must interpret each dataset to determine its usability and compatibility with other datasets. Semantic technologies turn the process of integrating datasets into a machine's task, which can be accomplished automatically, consistently, and thoroughly via the semantic comparison of dataset's contextual knowledge. For instance, machines can verify whether the contents of two datasets are semantically equivalent. Even if they are not equivalent, the machine can identify if any contextual difference is significant enough to enable or not the integration of the datasets.

A lab may feel confident in their current metadata practices for managing hand-collected data. However, for projects involving automated sensor networks, easily accessible and usable metadata is critical to harnessing the power of rapid data collection. Semantic technologies provide a solution for this new era in ecology because semantic metadata is structured and query-able, making it easy to access and use for data management and analysis.

5.2 Share-ability and Re-usability

MIT App Inventor makes it easy to share projects and to allow others to download the application to their phones. For those who wish to re-use this application, the source code file is available to reopen in App Inventor, from which point one can make changes to the interface and logic. Detailed instructions about sharing App Inventor projects are available here [4].

Since this application is based on a public set of ontologies, it may easily be re-used in projects that wish to use the same ontology to capture their deployment metadata. The application could be ready for a new use in just minutes by simply changing the endpoint URLs to a new project's knowledge base. Similarly, it would be relatively easy to make extensions to the application for small extensions required to the ontology. If a research team would like to collect metadata in a similar way, but with a different or modified ontology, it would still be useful to use this application as a starting point. Many of the queries that the app runs would likely need to be modified, but it may reduce development turnaround time to start with this app as a framework.

Our team has already found the need to extend MOCCASN and found the process to be very easy. While demonstrating MOCCASN to our field scientist teammates, we discovered that MOCCASN would be more useful to them if it collected information about samples in addition to information about deployments. After deciding on what terms from existing ontologies to use to represent sample-based knowledge, MOCCASN was extended to capture metadata about samples in just a few hours by adding “collect sample” and “analyze sample” screens.

5.3 Future Work

After having completed initial testing with a small collection of field scientists within our expanded team, we are beginning to deploy MOCCASN for real use in the Jefferson Project. The app will be in use among field scientists as they modify the arrangement of the equipment that autonomously monitors the lake, as well as researchers collecting samples in the field.

In addition, though this work focused on capturing metadata related to deployments, we plan to apply the same framework to rapidly develop additional tools for capturing a wide range of metadata. For example, we plan to build a similar tool to capture equipment calibration metadata in the lab, and for capturing the way sensor’s configuration parameters are set. Both of these activities will be performed routinely over the course of the Jefferson Project to maintain a well-functioning sensor network, and the calibration and configuration parameters are important for accurately comparing and combining datasets. In all sorts of human interventions, we are also planning to provide richer provenance knowledge about how sensor calibrations, deployments and configurations are decided.

More broadly, we think it would also be valuable to add to this tool the ability to preview the deployed object’s data stream. It is not uncommon to hear about a half of a day’s field work lost due to improperly set up equipment. Since many sensor network instruments stream their data back to a central hub, it should be possible for our tool, which is already connected to the knowledge base, to show the user what the instrument is streaming. This would help the researcher to correct mistakes quickly.

6 Conclusions

The advent of automation in data collection poses many opportunities for revolutionizing data analysis in ecology. However, the large volume of datasets will be difficult to use without improved metadata collection strategies. As more diverse data is collected with the aim of integration and analysis, it becomes more critical to thoroughly and accurately capture in-situ information concerning dataset collection. Simultaneously, we do not want to overly burden field researchers with inefficient or error-prone collection methods. We present a mobile application for automating the collection of in-situ metadata in an efficient, standardized, and error-free way.

7 Acknowledgements

We would like to thank Jeremy Farrell and Matt Schuler for their time beta testing and reviewing the application from the point of view of a field scientist. We also thank Katie Chastain for adopting this project after the primary developer moves on. We would also like to thank the Jefferson Project Group at large for their ongoing support and collaboration.

References

1. Github deployment metadata app, <https://github.com/lokijuhy/Deployment-Metadata-App>
2. Mit app inventor, <http://appinventor.mit.edu/> (Date last accessed: March 17, 2015).
3. Moccasin: A metadata collection app, <http://tw.rpi.edu/web/project/JeffersonProjectAtLakeGeorge/MetadataApp>
4. Sharing and packaging apps, <http://appinventor.mit.edu/explore/ai2/share.html> (Date last accessed: March 17, 2015)
5. Fegraus, E.H., Andelman, S., Jones, M.B., , Schildhauer, M.: Maximizing the value of ecological data with structured metadata: An introduction to ecological metadata language (eml) and principles for metadata creation. *Bulletin of the Ecological Soc. of America* 86(3), 158–168 (July 2005)
6. Jones, C., Blanchette, C., Brooke, M., Harris, J., Jones, M., Schildhauer, M.: A metadata-driven framework for generating field data entry interfaces in ecology. *Ecological Informatics* 2(3), 270–278 (2007)
7. Lebo, T., Sahoo, S., McGuinness, D., Belhajjame, K., Cheney, J., Corsar, D., Garijo, D., Soiland-Reyes, S., Zednik, S., Zhao, J.: Prov-o: The prov ontology. W3C Recommendation (April 2013), <http://www.w3.org/TR/prov-o/> (Date last accessed: March 18, 2005).
8. Madin, J., Bowers, S., Schildhauer, M., Krivov, S., Pennington, D., Villa, F.: An ontology for describing and synthesizing ecological observation data. *Ecological Informatics* 2(3), 279–296 (Oct 2007)
9. McGuinness, D., Fox, P., Middleton, D., Garcia, J., Cinquni, L., West, P., Darnell, J., Benedict, J.: Solar-terrestrial ontology develop. In: AGU Fall Meeting Abstracts. vol. 1, p. 0324 (2005)
10. McGuinness, D.L., Pinheiro, P., Patton, E.W., Chastain, K.: In21b-3712 semantic science for ecosystem understanding and monitoring: The jefferson project case study. In: Proceedings of AGU Fall Meeting 2014 (December 15-19 2014, Moscone Center, San Francisco, CA, US) (2014)
11. Michener, W.K.: Meta-inform. concepts for ecological data manage. *Ecological Informatics* 1(1), 3–7 (Jan 2006)
12. Michener, W.K., Brunt, J.W., Helly, J.J., Kirchner, T.B., Stafford, S.G.: Non-geospatial metadata for the ecological sciences. *Ecological Applicat.* 7(1), 330–342 (Feb 1997)
13. Pinheiro, P., McGuinness, D.L.: Provenance-enabled integration of sensor network data. In: AGU Fall Meeting 2014. San Francisco, CA, US (Dec 2014)
14. Reichman, O.J., Jones, M.B., Schildhauer, M.P.: Challenges and opportunities of open data in ecology. *Sci.* 331, 703–705 (2011)

Linked Data and Mobile Application Privacy

Evan W. Patton¹ and Ilaria Liccardi^{2,3}

¹ Dept. of Computer Science, Rensselaer Polytechnic Institute, Troy, NY USA

² Oxford e-Research Centre, University of Oxford, Oxford, UK

³ MIT CSAIL, Cambridge, MA USA

Abstract. The smartphone applications ecosystem lacks significant controls and accountability mechanisms to protect users' privacy from snooping applications, especially when context may significantly impact the decision to share sensitive information. In this paper, we discuss some of the existing privacy concerns of mobile applications. In particular we highlight how semantic web and linked data technologies have the potential to help improve privacy controls through rules, queries, and reasoning. These technologies have the potential to obtain and maintain the integrity of private data, but there are privacy challenges that may result from widespread deployment of mobile semantic technologies.

Keywords: privacy, mobile applications, linked data, accountability

1 Introduction

Mobile devices are a ubiquitous part of modern life, with over 2 billion estimated cellular connections in the world today [6]. The market for mobile applications, or 'apps', is significant. The increase in the number of apps has been accompanied by an increase in the amount of tracking and other forms of data collection on users, primarily for the purpose of generating advertising revenue to support application developers. A recent report by the United States Federal Trade Commission [4] shows that users are not always aware of the amount nor the extent to which companies track their behavior. We propose that Resource Description Framework (RDF) based technologies, such as rule engines, ontologies, and reasoners, can be used to assist mobile users in preserving their privacy by constructing a more robust platform for maintaining provenance about privacy-oriented data. Such tools will help users understand where their information is stored, how it is used, and create meaningful and informed decisions of how their information are shared. The paper is structured as follows:

- **People's reactions and attitudes towards current app access:** We highlighted related research that explores people's reaction when made aware of the kind of access apps have to their personal data. We highlighted possible reasons for developers wanting to access to these data (section 2).
- **Smartphone privacy scenarios:** We compared two hypothetical scenarios of apps accessing one's personal data. We described an implementation using semantic technologies that could grant or deny access to personal data based on context (section 3).

- **Possible solutions using semantic web technologies:** We presented possible implementations using semantic web technologies which are aimed at helping users’ preserve their privacy (section 4).

2 Related Research

Personal information is collected at a large scale from smartphone devices. Different operating systems have their own ways and mechanisms of presenting the access requested by different apps and of gaining consent from users.

Android and Windows devices present users a set of permissions requested by each app prior to installation. On these devices, users must accept all permissions before the app can be installed. Often users are not aware of the type of access, and even if they are able to understand they often do not know the reasons, motivations and purposes for accessing this information.

Apple devices use a different method for informing and gathering consent. An app that requires access to users’ personal information alerts the user to obtain permission the first time it is required. Users can decide whether to grant permission or not. While it is not always the case, denying access can result in the app failing to work properly or stopping working all together.

In all cases, apps can request access to personal data for monetization rather than core functionality. Mobile advertising is one of the most common and lucrative ways for app developers to monetize their apps [13]. Research on Android apps found that almost half of the apps (49% (473) of 964 apps) in the Google Play store requested multiple permissions only needed for advertising libraries [14]. Often users have no way of understanding differences in permissions, especially if personal information is legitimately needed by the app. For example, weather apps need the user’s location to gather and display location-based weather. However they can also send that information to advertisers to serve targeted ads. Users are often unaware of this access, particularly because only a small percentage of apps (fewer than 10%) have privacy policies specified on their description page [10]. Even when present, users often ignore privacy policies because they are often too vague or incomprehensible due to technical and legal jargon [11]. While Apple’s permission model allows users to have more fine-grained control over what data are accessed, users often grant access to all the information the app requires because they are unaware that the app uses their personal information for purposes other than the app’s functionality.

People are often unaware that apps may collect their personal data [3] as permission mechanisms are often difficult to understand [11] and part of this collection can happen silently in the background [18]. When users are made aware of this collection, they feel much less willing to share data which they perceive be extremely sensitive [15]. Some express shock and a desire to remove the app [1,12] or experience a sense of “creepiness” that results in a loss of trust [16]. The perceived sensitivity of data is subjective and can also vary within the individual’s context [15]. For example, a user might be willing to share when he or she is at a certain location or while engaging in a certain activity (e.g., relaxing),

but not when performing another (e.g., working). Users are also less willing to disclose personal data when more specific usage details are provided [15], which suggests that user awareness of how data are used is critical to preserving privacy. It is impossible to consent to data collection for every foreseeable purpose, given the incomplete, missing or difficult to understand information users receive when making the decision about whether to install an app [11,15].

3 Apps and Personal data: Access, sharing and consent

This section outlines a hypothetical, yet realistic scenario that depicts what happens when we carry and/or use smartphone devices. We then propose a possible improvement to the scenario using semantic technologies to grant or deny access to personal data using contextual information.

3.1 How currently is people’s data shared and access?

Alice needs to drive to work, but before she starts driving, she checks *routeInfo*, an app for real-time traffic updates (many such apps are present in app markets). To provide this information, *routeInfo* requests access to GPS. Alice is not aware, however, that other apps on her phone (e.g., a game she installed for her child) are also passively listening to these GPS updates to identify nearby businesses. These apps (like the majority of apps) use targeted advertising to generate revenue.

Alice arrives at work; her smartphone recognizes her work WiFi “Work-WiFi”. Alice’s company provides all employees with an app *openSesame* to navigate the building which grants access to rooms and personal offices.

Alice has another app, *remindME*, installed on her phone to help her with her errands, such as groceries and dry-cleaning. The app alerts Alice as she leaves work of her errands. She proceeds with all her to-dos while other apps are silently listening (e.g., apps gathering GPS updates for advertisement, *openSesame* knowing she is not longer at work). She runs her errands and drives back home. Her smartphone recognizes her home wifi “Home-WiFi”.

Developers who have collected Alice’s location can estimate that Alice lives at this address (Alice is there from evening to morning, everyday). Using public (possibly linked) data repositories, this information can be used to determine the average household income, home value, debt, number and length of car ownership, demographics, likely political affiliation, etc. All this information can be used by the developers for revenue (e.g. selling this information to data brokers), advertisement, etc. Alice never wanted or intended to supply such specific information, however. The lack of control over passive applications and types of information accessed has caused her device to leak significant private information about her without her **express** permission or intention.

3.2 How can semantic technologies improve smartphone privacy?

Alice needs to drive to work, but before she starts driving, she checks *routeInfo*, an app for real-traffic updates. To provide this information, *routeInfo* requests

GPS information and her context is set $\{\text{DRIVING}\}$. Alice only wants the foreground app to have access to GPS updates while driving. Since she has launched the app it receives GPS but other apps do not.

Alice arrives at work; her smartphone recognizes her work WiFi “Work-WiFi” and sets her context to $at.\{\text{WORK}\}$. Her company provides employees with an app, *openSesame*, that grants access to rooms and personal offices. Only *openSesame* is allowed to access information in the WORK context. She attends a meeting, her smartphone updates her context to $at.\{\text{WORK}\} \cap located.in.\{\text{BUILDING_A\#ROOM_304B}\}$, *openSesame* knows she is in the room.

Alice has another app, *remindME*, installed on her phone that helps her with her errands, such as groceries and dry-cleaning. The app alerts Alice of her errands after she leaves work. As she proceeds with all her to-dos, her smartphone updates her context to $intent.\{\text{RUNNING_ERRANDS}\}$. *openSesame* can not collect the location in this context, nor can other apps unless Alice has allowed it.

She runs her errands and drives home. Her smartphone recognizes her home wifi “Home-WiFi” and her smartphone updates her context to $at.\{\text{HOME}\}$; all apps are prevented from collecting location.

Developers have collected Alice’s location, but only for specific contexts. While Alice’s personal information can still be deduced, the information is more sparse than before. Alice is also aware of the possibility for developers to access her location when using an app. While her device may leak personal information, it is dependent on the context information set by Alice.

4 Employing semantic technologies for privacy

Given limitations of current app permission models, we propose that semantic technologies will improve the experience for users by enabling systems to:

1. Capture **provenance** of sensitive data to explain possible privacy violations.
2. Model user’s **intent** to share data (e.g., via description logic class expressions or rules).

Since many semantic technologies assume an open world, privacy preservation benefits from the fact that information can be *unknown*. We briefly explore some ways in which semantic technologies have been used to model privacy and how they might be adapted for mobile deployments.

4.1 Privacy information modeling

The Accountability in RDF (AIR) language was developed as a means of generating and tracking explanations on the Semantic Web [7]. Similar rule efforts include Rule Interchange Format (RIF) [8] and Semantic Web Rule Language (SWRL) [5]. PROV-O [9] provides an ontology for modeling provenance information in a linked way. Using a PROV middleware, one could track how private information are manipulated by different applications and shared with third parties. Users could also specify a custom privacy policy using intensional (i.e., described) contexts and/or rules to describe additional limits on applications.

4.2 Controlling access to private information

Tao et al. demonstrate query answering with privacy preservation in the \mathcal{EL} description logic [17]. They show that a set of secrets can be protected by inverting the application of \mathcal{EL} expansion rules to compute an “envelope of secrets.” Queries are answered using secrets so long as the system will not reveal sufficient information in the envelope that could infer secrets. Further, the computational complexity of both \mathcal{EL} and the envelope creation is polynomial, which makes mobile deployments feasible. It is an open question whether secrets expressible in a more expressive logic (e.g., \mathcal{ALC}) will face tractability issues.

The WebID⁴ and WebACL⁵ community groups at the W3C are iterating on frameworks for identification, authentication, and authorization built on RDF-based representations. While these technologies are primarily aimed at web-based deployments, one can see how their approach to access control using constraints structured in RDF could be repurposed for a mobile environment.

Chen et al. explore use of semantic technologies in a Context Brokering Architecture for mobile environments [2]. They use the architecture to prototype an intelligent meeting room that provides services by inferring when users enter the room from sensor data streams. While context can be used to customize an environment to the user’s needs, we note that devices might be unintentionally informing untrusted external systems about user context as well.

5 Conclusion

The contextual features provided by mobile devices are a double-edged sword for end users. They provide significant information to third-party applications to tailor behavior to one’s situation, but can also provide others, e.g. advertisers, a significant look at one’s behavior. Worse, the permissions models on mobile phones lack transparency and fine control over how context is used and disseminated. We have presented a scenario where semantic technologies could be used to capture provenance of private data and enforce user privacy policies.

Other techniques may be necessary to supplement linked data approaches to modeling dynamic user context. Markov models, for example, may provide a more compact, statistical representation of past context in a space and time complexity amenable to mobile devices compared with tableaux reasoning about context provenance modeled in PROV-O. Due to the error present in the various sensors used to infer a user’s context, probabilistic models and logics will also be relevant to enabling privacy-preserving technologies for mobile devices.

There are a number of open research questions for the community. Do we need to develop novel representation languages for context and privacy, especially in the face of probabilistic contextual features, or is RDF sufficient? What is the maximal feature space for context and privacy in description logics and is their evaluation on mobile devices sufficient? How can reasoning about context be made efficient on mobile devices to reduce energy drain?

⁴ <http://www.w3.org/wiki/WebID>

⁵ <http://www.w3.org/wiki/WebAccessControl>

References

1. Balebako, R., Jung, J., Lu, W., Cranor, L.F., Nguyen, C.: Little brothers watching you: Raising awareness of data leaks on smartphones. In: Proc. 9th SOUPS. pp. 12:1–12:11 (2013)
2. Chen, H., Finin, T., Joshi, A.: An intelligent broker for context-aware systems. In: Adjunct proceedings of Ubicomp. vol. 3, pp. 183–184 (2003)
3. Felt, A.P., Egelman, S., Wagner, D.: I’ve got 99 problems, but vibration ain’t one: A survey of smartphone users’ concerns. In: Proc. 2nd SPSM. pp. 33–44 (2012)
4. FTC Staff: Mobile privacy disclosures: Building trust through transparency. Tech. rep., Federal Trade Commission (2013)
5. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosz, B., Dean, M.: SWRL: A semantic web rule language combining OWL and RuleML. Tech. rep., W3C (2004)
6. ICT Data and Statistics Division: The world in 2014: ICT facts and figures. Tech. rep., International Telecommunications Union (2014)
7. Kagal, L., Jacobi, I., Khandelwal, A.: Gasping for air why we need linked rules and justifications on the semantic web. Tech. rep., MIT (2011)
8. Kifer, M., Boley, H.: RIF overview (2nd edition). Tech. rep., W3C (2013)
9. Lebo, T., Sahoo, S., McGuinness, D.L.: PROV-O: The PROV ontology. Tech. rep., W3C (Apr 2013)
10. Liccardi, I., Pato, J., Weitzner, D.J.: Improving Mobile App selection through Transparency and Better Permission Analysis. *J. Privacy and Confidentiality* 5(2), 1–55 (2014)
11. Liccardi, I., Pato, J., Weitzner, D.J., Abelson, H., De Roure, D.: No technical understanding required: Helping users make informed choices about access to their personal data. In: Proc. 11th MOBIQUITOUS. pp. 140–150 (2014)
12. Lin, J., Amini, S., Hong, J.I., Sadeh, N., Lindqvist, J., Zhang, J.: Expectation and purpose: Understanding users’ mental models of mobile app privacy through crowdsourcing. In: Proc. 14th Ubicomp. pp. 501–510 (2012)
13. Merisavo, M., Vesänen, J., Arponen, A., Kajalo, S., Raulas, M.: The effectiveness of targeted mobile advertising in selling mobile services: an empirical study. *Int. J. Mobile Commun.* 4(2), 119–127 (2006)
14. Pearce, P., Felt, A.P., Nunez, G., Wagner, D.: Addroid: privilege separation for applications and advertisers in android. In: Proc. 7th ASIACCS. pp. 71–82 (2012)
15. Shih, F., Liccardi, I.: Privacy tipping points in smartphones privacy preferences. In: Proc. 33rd CHI. pp. 807–816 (2015)
16. Shklovski, I., Mainwaring, S.D., Skúladóttir, H.H., Borgthorsson, H.: Leakiness and creepiness in app space: Perceptions of privacy and mobile app use. In: Proc. 32nd CHI. pp. 2347–2356 (2014)
17. Tao, J., Slutzki, G., Honavar, V.: Secrecy-preserving query answering for instance checking in EL. Tech. rep., Iowa State University (2010)
18. Zhang, F.: Assessing intrusiveness of smartphone apps. Tech. rep., Masters thesis, MIT (2012)

Mobile Semantic Query Distribution with Graph-Based Outsourcing of Subqueries

William Van Woensel

NICHE Research Group, Faculty of Computer Science,
Dalhousie University, Halifax, Canada
{william.van.woensel}@dal.ca

Abstract. While mobile computing domains have illustrated the usefulness of mobile semantic data, improvements in mobile hardware are paving the way for local semantic data access. To support this, a number of tools have been developed for storing, querying and reasoning over local semantic data. However, recent benchmarks have shown that mobile hardware still imposes limitations on efficient local data querying. Additionally, mobile scenarios pose unique challenges due to their dynamic nature; making it difficult to replicate semantic data a priori for local querying. In this paper, we propose a graph-based query distribution approach, which efficiently distributes query execution across configured remote datasets. Importantly, our approach aims to identify subqueries that can be outsourced to remote datasets, thus reducing local joining work.

Keywords: mobile applications, query distribution, resource constraints

1 Introduction

As shown by the Linked Open Data cloud [1], a staggering number of online, machine-readable and interconnected Semantic Web datasets are currently available. Multiple tools and techniques have been developed to access this wealth of data. Local replication [2] involves replicating relevant parts of semantic datasets locally, allowing for robust and efficient access. Virtual data integration, or query distribution, distributes queries over the remote datasets themselves, integrating the results locally [3, 4].

For some time now, mobile devices have met the hardware requirements for managing and querying Semantic Web data. Reflecting this evolution, various mobile computing domains currently leverage semantic data, including augmented reality [5], recommender systems [6], location-aware [7] and context-aware systems [8], mobile tourism [9] and m-Health [10]. Supporting these approaches, multiple tools have been developed for constructing, managing, querying and reasoning over local semantic data on mobile devices, including AndroJena [11], a port of the well-known Apache Jena framework [12], and Rdf On The Go [13], which was specifically developed for mobile systems. However, as shown by recent benchmarks [14, 15], mobile hardware limitations regarding processing power, memory and battery capacity, limit the scale of purely local solutions. Many mobile scenarios also pose unique challenges due to their highly dynamic nature; e.g., cases where semantic data related to the user's dynamic

context needs to be continuously accessible. Such scenarios makes a priori, local replication of relevant data on the device problematic. Virtual data integration solutions bypass this issue by executing queries directly on remote datasets. Moreover, by leveraging the capabilities of remote datasets, opportunities exist for dealing with mobile hardware limitations.

In particular, subqueries may be outsourced to relevant remote datasets, relieving the mobile client of join processing. We also note that server hardware hosting these datasets are better equipped, both hardware-wise and regarding data access optimizations (e.g., join indices), to execute these subqueries to begin with. Moreover, less intermediate results are returned to the device, reducing bandwidth usage. To allow identifying subqueries that are resolvable by a particular dataset, we propose indexing graph patterns (i.e., graph structure with only predicate edges) found in the dataset. For a given query and set of configured datasets, suitable subqueries are found by determining subgraph isomorphism between the query subgraphs and dataset graph patterns. Although subgraph checking is an NP-hard problem, it has reasonable execution times for many real-world scenarios and is often used in graph databases [16].

In this paper, we present a graph-based, semantic query distribution approach, which outsources suitable subqueries via graph pattern indexing and matching. We apply a custom, back-tracking subgraph isomorphism algorithm, which is able to identify subqueries suitable to be executed on a particular remote dataset. We present an evaluation comparing our system to a predicate-based approach, using a real-world dataset.

Section 2 discusses the indexing of dataset graph patterns. Section 3 presents our query distribution approach. Section 4 shows an initial evaluation of our approach, while Section 5 discusses related work. Section 6 presents conclusions and future work.

2 Indexing dataset graph patterns

To identify dataset graph patterns, our system first collects instance RDF graphs. Duplicate instance graphs and subgraphs are hereby ruled out by applying subgraph checks on the collected instance graphs, leaving only distinct graph patterns. Below, we show the pseudocode for this indexing step:

```

1.  $q \leftarrow \text{SELECT } * \text{ WHERE } \{ ?s ?p ?o . \}$ 
2.  $result \leftarrow \text{execute}(q, \text{dataset})$ 
3.  $graphs \leftarrow \emptyset$ 
4. for each  $t_1$  in  $result$ 
5.    $result = result - t_1$ 
6.    $graph_1 \leftarrow \emptyset$ 
7.    $triples \leftarrow [t_1]$ 
8.   for each  $curT$  in  $triples$ 
9.      $graph_1 \leftarrow graph_1 + curT$ 
10.    for each  $t_2$  in  $result$ 
11.      if  $matches(curT, t_2)$  then
12.         $result \leftarrow result - t_2$ 
13.         $triples \leftarrow triples + t_2$ 
14.      end if
15.    end for
16.    end for
17.    end for
18.     $exists \leftarrow false$ 
19.    for each  $graph_2$  in  $graphs$ 
20.      if  $is\_subgraph(graph_1, graph_2)$  then
21.         $exists \leftarrow true$ 
22.        break
23.      else if  $is\_subgraph(graph_2, graph_1)$  then
24.         $graphs \leftarrow graphs - graph_2$ 
25.      end for
26.      if  $\neg exists$ 
27.         $graphs \leftarrow graphs + graph_1$ 

```

Code 1. Algorithm for extracting dataset graph patterns.

Lines 1-2 obtain all triples from the dataset. For each distinct result triple (lines 4-5), a new graph pattern is created (line 6), as well as a list of candidates for expansion, initially containing the result triple (line 7). Each candidate for expansion is added to the graph pattern (lines 8-9), and other triples linking to the current candidate (lines 10-11) are themselves added as candidates for expansion (line 13). This process continues until no more new expansion candidates are found, meaning a disjoint instance RDF graph has been identified.

Subsequently, the algorithm checks whether the collected graph is a subgraph of another, previously identified graph, or vice versa (lines 18-27). In case it is found to be a (non-proper) subgraph, the newly found graph is ignored (lines 20-22). In case a previously indexed graph is a subgraph of the new graph, the previous is removed & the new graph is added (lines 24-25, 26-27). Else, the new graph is added to the index as a new graph pattern (lines 26-27).

In Section 3, we elaborate on the implementation of the *is_subgraph* function. We note that the *matches* function only considers certain types of links, to maximize the re-use of extracted graph patterns. Overall, the function may consider 4 links to extend an instance graph, as illustrated in Figure 1:

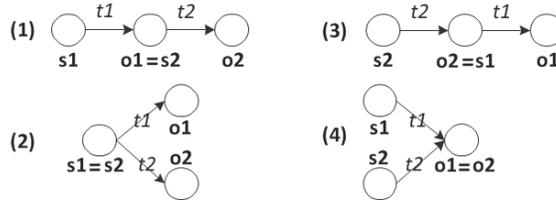


Figure 1. Potential links followed during indexing.

Two triples may be considered part of the same instance graph in case they represent a path (links (1) and (3)), and if they share the same subject (link (2)) or object (link (4)). In practice however, we found that considering link (4) typically leads to cases where only a single (huge) graph pattern can be extracted (i.e., about the same size as the dataset). For instance, most resources will often be typed with *owl:Thing*, resulting in only one instance graph. Currently, we follow a pragmatic solution to this problem, by simply ruling out link (4); thus maximizing re-use of dataset graph patterns, and significantly reducing the size of extracted graph patterns. On the other hand, we note that this will lead to problems if the shared object itself is involved in other triples as subject. Tackling this issue more effectively is considered future work.

After extracting the graph patterns, they are added to an index keeping the graph patterns for each dataset. Ideally, the resource-intensive indexing process occurs on the dataset server, ruling out the need to communicate the entire dataset to another location. Subsequently, indexed graph patterns are communicated to the mobile systems, and updated each time significant changes occur that alter the previously indexed patterns.

3 Graph-based Query Distribution

Based on the dataset graph pattern index (see Section 2), query execution will be distributed across matching datasets. To cope with mobile device limitations, our main goal is to distribute coherent subqueries to relevant datasets, thus outsourcing the resource-intensive join work to

database systems better equipped to execute the work, both hardware-wise and regarding internal optimizations (e.g., join indices).

We apply a graph-based approach to identify subqueries resolvable by remote datasets. To that end, we developed a custom subgraph isomorphism algorithm, discussed in Section 3.1. Next, we present our query distribution algorithm (Section 3.2).

3.1 Backtracking subgraph-isomorphism

A subgraph isomorphism algorithm checks whether graph g_1 is isomorphic to a (non-proper) subgraph of graph g_2 . To suit our purposes, we developed a slightly modified algorithm, which is able to identify the largest (non-proper) subgraph of g_1 that is isomorphic to a (non-proper) subgraph of g_2 . In doing so, our system can identify which subqueries, or query subgraphs, are resolvable by dataset subgraphs.

For this purpose, a listener traces the algorithm execution, and tracks the most successful comparison. Furthermore, certain optimizations, applicable when only checking for subgraph isomorphism, need to be dropped (indicated by (†)). Code 2 shows the pseudocode for this algorithm. In the *is_subgraph* function, lines 2-5 check whether the g_1 graph is subgraph-isomorphic to graph g_2 . In particular, for each combination of nodes n_1 and n_2 , the code checks whether the graph reachable from n_1 is a subgraph of the n_2 graph, using the *compare* function. In case the n_1 subgraph was compared successfully to g_2 (line 5), and all g_1 nodes were mapped (line 6), g_1 is a subgraph of g_2 . Else, previous mappings are undone (lines 10, 14) and another node combination is tried. If no complete match was found in any comparison, g_1 is not a subgraph of g_2 (line 20). We note that, in case no n_2 nodes were found that even partially match n_1 (line 13-16), g_1 cannot be a subgraph of g_2 ; and false could be returned (at line 17). However, a partial subgraph match, involving a subgraph of g_1 , may still occur; so this code is left out.

```

1. function is_subgraph( $g_1, g_2$ )
2.   for each node  $n_1$  in graph  $g_1$ 
3.     for each node  $n_2$  in graph  $g_2$ 
4.       map( $n_1, n_2$ )
5.       if compare( $n_1, n_2$ ) then
6.         if mapping_complete()
7.           listener :: done()
8.           return true
9.         else then
10.          mapping_rollback()
11.          listener :: compare_fail()
12.        end if
13.      else then
14.        clear_mapping( $n_1$ )
15.        listener :: compare_fail()
16.      end if
17.    end for (†)
18.  end for
19.  listener :: done()
20.  return false

21. function compare( $n_1, n_2$ )
22.  matches ← neighbor_matches( $n_1, n_2$ )
23.  if !matches then
24.    return false
25.  end if
26.  l1: for each matchi in matches
27.     $e_1$  ← matchi .  $e_1$ 
28.    l2: for each  $e_2$  in matchi .  $e_2$ s
29.      listener :: comparing( $e_1, e_2$ )
30.      map ← get_mapping( $e_1$  . toNode)
31.      if map = NULL
32.        map( $e_1$  . toNode,  $e_2$  . toNode)
33.        if compare( $e_1$  . toNode,  $e_2$  . toNode)
34.          break l2
35.        else clear_mapping( $e_1$  . toNode)
36.        end if
37.      else if map =  $e_2$  . toNode
38.        break l2
39.      end if
40.    end for (†)
41.  end for

```

¹ These optimizations are enabled while building the dataset graph pattern index.

```

42. listener :: done_comparing(n1, n2)
43. return true

```

Code 2. Custom, back-tracking subgraph isomorphism algorithm.

The *compare* function starts by checking for overlaps between the outgoing edges of n_1 and n_2 (*neighbor_matches*; line 22). In case n_1 is not a leaf node and no overlaps are found, false is returned (lines 23-25). For each overlap, the function checks whether the e_1 to-node is already mapped to a g_2 node (line 30; to avoid infinite loops). If so, and if it was already mapped to the e_2 to-node, edge e_2 matches e_1 (lines 37-38). If not, the e_1 to-node is mapped to the e_2 to-node (line 32), and the function recursively compares these two to-nodes (line 33). E.g., in case no overlapping edges are found, this call will return false; if n_1 turns out to be a leaf node, it will return true. In case n_1 and n_2 recursively match, edge e_2 matches e_1 (line 33-34). If not, the previously assigned mapping is removed (line 35) and another edge e_2 (if any) is tried². Again, at this point, the algorithm could return false if no matches are found for e_1 (at line 40), since all e_1 edges need to be matched for a subgraph match. However, to allow identifying partial matches (i.e., involving a subgraph of g_1), all e_1 edges need to be tried; even if some have already failed.

The *neighbor_matches* function returns true in case n_1 is a leaf node; since this means the n_1 subgraph has been checked completely. Else, it collects the overlaps between the outgoing edges of e_1 and e_2 : whereby two edges match in case both of them have the same label; either of them represents a variable; or the e_1 label represents a subproperty of e_2 . If no matches are found for any edge e_1 , the function returns false; else, it returns the overlapping edges.

To track the largest matching subgraph of g_1 , a listener is notified when two edges are being compared (line 29), when two nodes are finished comparing (line 42), when a g_1 node comparison failed (lines 11, 15) and when comparison is done (lines 7, 19). Upon finishing the subgraph comparison, the listener records the match by assigning dataset associated with the dataset edge to its matching query edge; together with an ID uniquely identifying the subgraph comparison³.

3.2 Query Distribution

To achieve virtual data integration, query execution is distributed across the configured datasets, and the results integrated locally. In its simplest form, this involves splitting up a query into its smallest units (i.e., triple patterns), executing them on each individual dataset, and combining the results. In doing so, a query distribution system ensures that all results are returned, even for queries that are not resolvable by any single dataset. Initially, such a *Query Distribution Plan (QDP)* consists of n^m *query sets*, each representing a particular result integration:

$$QDP = [\{ t_1 \rightarrow D_x, \dots, t_i \rightarrow D_y, \dots, t_m \rightarrow D_z \}, \dots]$$

$$n = \text{size}(\text{datasets}), m = \text{size}(\text{query}), 0 < i < m, 0 < x, y, z < n$$

Formula 1. Query Distribution Plan (QDP)

² Multiple e_2 matches for e_1 are possible, and vice-versa (e_2 is only matched to one e_1 at a time).

³ This unique ID is required by the query distribution algorithm (see Section 3.2).

Where $t_i \rightarrow D_x$ stands for an atomic subquery, i.e., executing a single triple pattern t_i on dataset D_x ; a set of subqueries between accolades forms a query set, standing for a particular integration of results; and the set of query sets make up the QDP, standing for all possible result integrations.

In our query distribution approach, graph patterns from an incoming query are compared to the set of dataset graph patterns (see Section 2), using subgraph isomorphism checks. After these checks, matching datasets are assigned to the query graph edges (see Section 3.1, last paragraph), indicating which query triples (each corresponding to an edge) are collectively resolvable by particular datasets. Based on these results, given a query set, multiple t_i matched to the same D_y (during the same subgraph check) can be grouped into the same subquery:

$$\begin{aligned} \text{query set} &= \{t_1 \rightarrow D_x, \dots, t_{i,j,k} \rightarrow D_y, \dots, t_m \rightarrow D_z\} \\ n &= \text{size}(\text{datasets}), m = \text{size}(\text{query}), 0 < i, j, k < m, 0 < x, y, z < n \end{aligned}$$

Formula 2. Grouping subqueries in query sets based on their shared dataset.

In the query shown in Figure 3, subqueries $t_1, t_2, t_3 \rightarrow A$, $t_2, t_4 \rightarrow B$, $t_1 \rightarrow C$ and $t_5 \rightarrow D$ can be distinguished into their respective query sets.

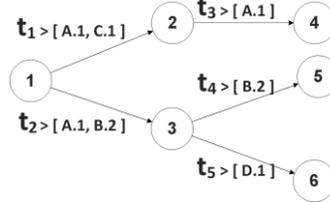


Figure 3. Example query matched to the configured datasets.

In this process, it is important to consider the particular subgraph check in which the matching dataset was found. For instance, consider the following cases:

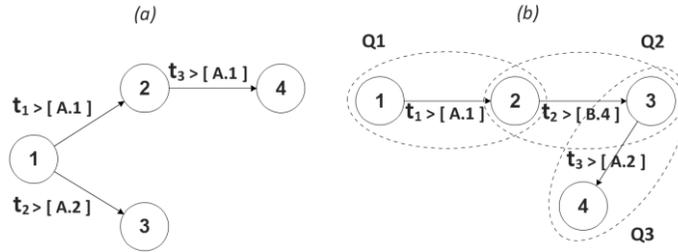


Figure 4. Distinct graph matches when constructing subqueries.

In case (a), part of the query (t_1, t_3) was matched to a particular graph pattern (1) from A during one subgraph check, while the remainder (t_2) was matched to a different graph (2) from A during another check. However, these two dataset graph patterns are disjoint; no single instance graph exists that covers both graph patterns⁴. As such, this particular query set will never yield any results, and need to be removed from the QDP. Case (b) illustrates that this reasoning is only valid when considering *connected* query triples (i.e., with shared variables).

⁴ Else, they would have been combined during the graph extraction process (see Section 2).

Here, an intermediate query triple t_2 is executed on dataset B, which may yield results that connect t_1 with t_3 ; in other words, an instance graph, integrated from both datasets, may exist that connects all 3 query triples.

Further, we note that triple patterns executed on the same dataset, but not sharing any variables, should ideally be kept separate. Putting these into the same subquery will lead to a Cartesian product, resulting in a huge number of results returned by the remote dataset; whereas the associated local computational work is comparatively low.

Below, we show the pseudocode for post-processing the QDP, based on subgraph matches:

```

1. l1: for each query set in qdp
2.   subqueries  $\leftarrow$  group on dataset and shared vars (query set)
3.   for subquery in subqueries
4.     l2: for each  $t_1$  in subquery
5.       for each other  $t_2$  in subquery
6.         if  $t_1$ .dataset.id  $\neq$   $t_2$ .dataset.id then
7.           remove query set from qdp
8.           break l2
9.         end if
10.    end for
11.  end for

```

Code 5. Processing the QDP based on subgraph matching results.

For each query set, query triples are grouped into subqueries based on assigned dataset and shared variables (lines 1-2). If one of these subqueries involves two query triples, assigned to the same dataset but associated with a different dataset graph pattern (lines 4-6), the query set is removed from the QDP (lines 7-8).

After generating a QDP, it is passed to the execution engine. For each subquery, the engine creates and executes a SPARQL query on the associated remote dataset. To integrate subquery results from a single query set, we apply a hash join. Results from multiple query sets are combined via a union operation. Since the same subquery-on-dataset combination will occur in multiple query sets (see Formula 1), the engine caches previous results for later re-use.

4 Evaluation

This section presents a preliminary evaluation of our query distribution approach. In our evaluation, a client app poses a query that requires data from two datasets to be integrated. To illustrate the usefulness of graph-based query distribution, we compare our approach to a straightforward predicate-based approach, which distributes incoming queries solely based on query predicates and indexed dataset predicates. For each query triple, the approach checks which datasets contain its concrete predicate; and then executes the query triple (potentially grouped in a subquery) on the found datasets.

Below, we elaborate on the evaluation setup, including current implementation components. Then, we discuss the results of each query distribution approach.

4.1 Setup

We ran all experiments 10 times, and took the average of the performance times. Below, we elaborate on other relevant aspects:

- Dataset & query

- *Dataset*

Using interlinks made available by DBPedia⁵, we extracted two small datasets from DBPedia (3846 triples; 487 Kb) and Geonames (1210 triples; 157 Kb). Both datasets supply different data on the same resources; whereby the extracted Geonames dataset uses DBPedia resource URIs to allow for data integration. Both datasets can be found online [19]. Although these datasets are relatively small, we will show that these a) already result in non-trivial execution times and b) indicate significant differences in performance between the evaluated approaches. Respectively, 7 and 2 distinct graph patterns were found in the extracted DBPedia and Geonames datasets.

- *Query*

Our evaluation executes the following query, selecting the label, type, coordinates and website of geographic entities (namespaces omitted for brevity):

```
1. SELECT * WHERE {
2.   ?subject rdfs:label ?label .
3.   ?subject rdf:type ?type .
4.   ?subject wgs:lat ?lat .
5.   ?subject wgs:long ?long .
6.   ?subject dbp:website ?website . }
```

Code 6. Evaluation query.

This query returns 51 results on the integrated dataset.

- Hardware

- *Dataset*

Both datasets were made accessible using the Apache Fuseki [17] SPARQL server, deployed on a Dell PowerEdge 2950 Server running Windows Server 32-bit, with (2) Intel Xeon 2.33 GHz and 64 Gb RAM. The datasets were indexed on a Lenovo Thinkpad, running Windows 7 64-bit, with Intel Core i7-3520M 2.90 Ghz and 8Gb RAM.

- *Mobile*

We used a LG Nexus 5 (model LG-D820) running Android 5.1.1 (Lollipop), with 2.26 GHz Quad-Core Processor, 2Gb RAM and 32Gb storage. The mobile device connects to the SPARQL server over an Internet connection⁶ (using WiFi).

- **Libraries:** to index datasets, we utilize Apache Jena 2.11.0 [12]. For performing query distribution on mobile systems, we rely on AndroJena 0.5 [18].

⁵ Such interlinks indicate resource equivalence with other major datasets.

⁶ To mimic real-life conditions, the SPARQL endpoint was not hosted on the same local network.

4.2 Results

Figure 5 shows the evaluation query graph after matching to indexed graph patterns, together with the resulting query sets. Although the DBpedia dataset (DB) contains individual triples matching all query triples, no single connected instance graph connects all 5 query triples (see DB.1 and DB.2 graph matches). Therefore, resolving this query requires data integration with the GeoNames dataset (GN). Based on graph matches shown in the query graph, two query sets are generated, each with a coherent subquery assigned to one of the datasets.

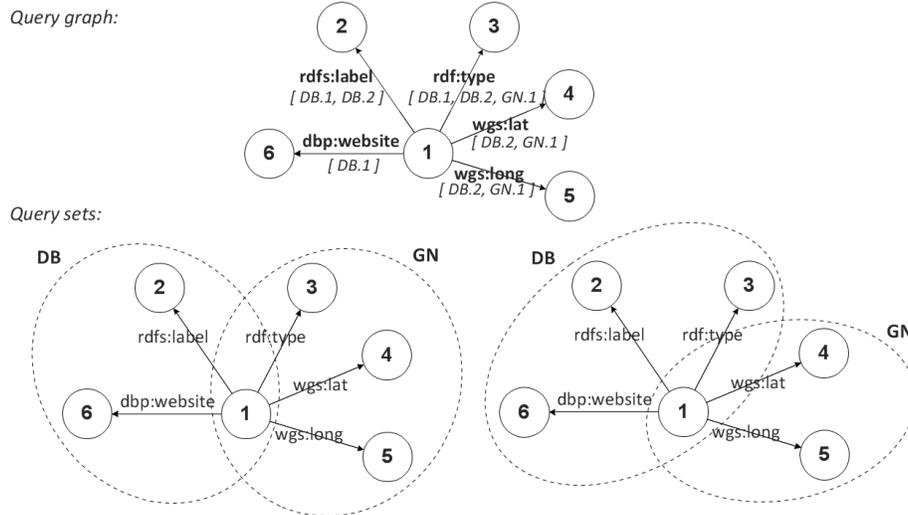


Figure 5. Query graph & subqueries for evaluation query.

For the predicate-based approach, we consider two configurations: a configuration where joins are not outsourced (*no-outsourced*); and a configuration where, per query set, query triples assigned to the same dataset are grouped into the same subquery (*outsourced*). In Table 1, we indicate the number of query sets, remote query executions and total number of individual query results to be joined.

	graph-based	predicate-based	
		<i>no-outsourced</i>	<i>outsourced</i>
# query sets	2	8	8
# query exec.	4	8	15
# indiv. results	151	1030	451

Table 1. Number of query sets and remote query executions.

The overall number of query executions is relatively low, since an internal cache is kept to avoid re-sending the same subquery (see discussion after Code 5). We also note that, when outsourcing queries, additional subqueries will be constructed. Therefore, the potential for re-using cached results is reduced, and the overall number of query executions is comparatively

increased (see *predicate-based* > *outsource* column). However, the local join work is reduced, as illustrated by the total number of individual query results.

Table 2 shows the performance results, where ID stands for identifying relevant datasets and QDP for constructing the dataset (see Code 5):

	graph-based	predicate-based	
		<i>no-outsource</i>	<i>outsource</i>
<i>parse query</i>	8		
<i>ID</i>	4	0.1	0
<i>QDP</i>	40	0	0.7
<i>execute</i>	529	2065	1856
<i>join</i>	16	1654	19
<i>total</i>	597	3728	1883

Table 2. Performance results (ms).

Since less queries are sent to the datasets, executing queries takes much less time for *graph-based*. Since *pred-based* > *outsource* and *graph-based* both outsource join work to the remote dataset, locally joining results is much faster as well. Despite its extra overhead when identifying relevant datasets and constructing the QDP, our graph-based query distribution approach outperforms either predicate-based approach.

Creating the graph index and predicate index takes ca. 4431ms and 80ms, respectively. We note that that the graph creation process only needs to be applied in case the dataset contents are updated significantly, causing a change in its graph patterns.

5 Related work

The Distributed ARQ (DARQ) [3] and Semantic Web Integrator and Query Engine (SemWIQ) [4] systems keep an index with summary dataset info. DARQ keeps so-called service descriptions, including found predicates, constraints on subjects and objects occurring with these predicates, and statistical data. The SemWIQ system maintains a catalog per data source, which keeps a list of classes and their number of instances, as well as a list of properties and their number of occurrences. Given a posed query, these indices are used to determine which triple patterns should be sent to which datasets. As such, these works do consider join outsourcing; which has the potential for large performance gains, as shown by our evaluation.

The approach in [20] resembles our work, as it focuses on indexing found predicate sequences or paths. This allows identifying datasets that can handle particular query predicate paths, with the goal of reducing local join work. In contrast, our approach supports outsourcing any kind of subquery, and is not just limited to path-based queries.

6 Conclusions & Future work

In this paper, we presented a graph-based query distribution approach, focusing on outsourcing subqueries to relevant remote datasets. We presented a mechanism for indexing graph patterns in remote datasets; a custom, backtracking subgraph isomorphism algorithm; and our graph-based query distribution mechanism. Our evaluation shows that our approach has the potential to significantly reduce the number of queries to be sent to remote datasets, as well as minimize the resulting local join work.

Many avenues for future work exist. By keeping summary data on graph pattern nodes (cfr. [3]), the “joinability” between graph patterns of different datasets can also be considered when ruling out query sets (see Figure 4 (b)). Edges in extracted graph patterns can be annotated with the number of associated instance graphs, to guide join optimizations. Currently, extracted graph patterns are kept per dataset. By keeping a single index, equivalent graph patterns from multiple datasets can be merged, thus reducing the number of isomorphism checks.

To allow identifying partial subgraph matches, our subgraph checking algorithm drops a number of optimizations that may result in serious performance gains. Studying other methods of efficiently determining partial query matches is future work. Furthermore, although subgraph isomorphism checking is known to be an NP-hard problem, many algorithms have been proposed over the years that solve it in a reasonable time [16]. In case our straightforward, custom algorithm leads to problematic performance for larger datasets, future work may involve studying and re-using other algorithms.

7 References

1. Cyganiak, R., Jentzsch, A.: The Linking Open Data cloud diagram, <http://lod-cloud.net/>.
2. Zander, S., Schandl, B.: A framework for context-driven RDF data replication on mobile devices. Proceedings of the 6th International Conference on Semantic Systems. pp. 22:1–22:5. ACM, New York, NY, USA (2010).
3. Quilitz, B., Leser, U.: Querying distributed RDF data sources with SPARQL. ESWC’08: Proceedings of the 5th European semantic web conference on The semantic web. pp. 524–538. Springer-Verlag, Berlin, Heidelberg (2008).
4. Langegger, A., Wöß, W., Blöchl, M.: A semantic web middleware for virtual data integration on the web. Proceedings of the 5th European semantic web conference on The semantic web: research and applications. pp. 493–507. Springer-Verlag, Berlin, Heidelberg (2008).
5. Reynolds, V., Hausenblas, M., Polleres, A., Hauswirth, M., Hegde, V.: Exploiting linked open data for mobile augmented reality. W3C Workshop: Augmented Reality on the Web (2010).
6. Ziegler, C.: Semantic web recommender systems. In Proceedings of the Joint ICDE/EDBT Ph.D. Workshop 2004 (Heraklion. pp. 78–89. Springer-Verlag (2004).
7. Becker, C., Bizer, C.: DBpedia Mobile: A Location-Enabled Linked Data Browser. In: Bizer, C., Heath, T., Idehen, K., and Berners-Lee, T. (eds.) LDOW. CEUR-WS.org (2008).
8. Van Woensel, W., Casteleyn, S., Paret, E., De Troyer, O.: Mobile Querying of Online Semantic Web Data for Context-Aware Applications. IEEE Internet Comput. Spec. Issue (Semantics Locat. Serv. 15, 32–39 (2011).
9. Keller, C., Pöhland, R., Brunk, S., Schlegel, T.: An Adaptive Semantic Mobile Application for Individual Touristic Exploration. HCI (3). pp. 434–443 (2014).

10. Puertas, E., Prieto, M.L., De Buenaga, M.: Mobile Application for Accessing Biomedical Information Using Linked Open Data. Proceedings of the 1st Conference on Mobile and Information Technologies in Medicine. , Prague, Czech Republic (2013).
11. AndroJena, <https://code.google.com/p/androjena/>.
12. Apache Jena, <https://jena.apache.org/>.
13. Le-Phuoc, D., Parreira, J.X., Reynolds, V., Hauswirth, M.: RDF On the Go: An RDF Storage and Query Processor for Mobile Devices. 9th International Semantic Web Conference (ISWC2010) (2010).
14. Patton, E.W., McGuinness, D.L.: A Power Consumption Benchmark for Reasoners on Mobile Devices. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C.A., Vrandečić, D., Groth, P.T., Noy, N.F., Janowicz, K., and Goble, C.A. (eds.) The Semantic Web - {ISWC} 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part {I}. pp. 409–424. Springer (2014).
15. Woensel, W. Van, Haider, N. Al, Ahmad, A., Abidi, S.S.R.: A Cross-Platform Benchmark Framework for Mobile Semantic Web Reasoning Engines. The Semantic Web - {ISWC} 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part {I}. pp. 389–408 (2014).
16. Lee, J., Han, W.-S., Kasperovics, R., Lee, J.-H.: An in-depth comparison of subgraph isomorphism algorithms in graph databases. Proceedings of the 39th international conference on Very Large Data Bases. pp. 133–144. VLDB Endowment (2013).
17. Apache Fuseki, <http://jena.apache.org/documentation/fuseki2/>.
18. AndroJena, <http://code.google.com/p/androjena/>.
19. Online Datasets, <https://niche.cs.dal.ca/materials/qd/>.
20. Stuckenschmidt, H., Vdovjak, R., Broekstra, J., Houben, G.: Towards distributed processing of RDF path queries. Int. J. Web Eng. Technol. 2, 207–230 (2005).