Building a Mobile Applications Knowledge Base for the Linked Data Cloud

Primal Pappachan[1], Roberto Yus[2], Prajit Kumar Das[3],
Sharad Mehrotra[1], Tim Finin[3], and Anupam Joshi[3]

[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 Store[4] and the Apple App Store[5], 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 Mobipedia[6], 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.
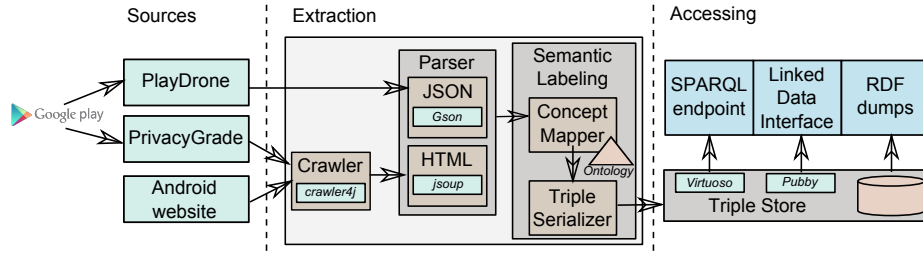

Fig. 1. Mobipedia overview.
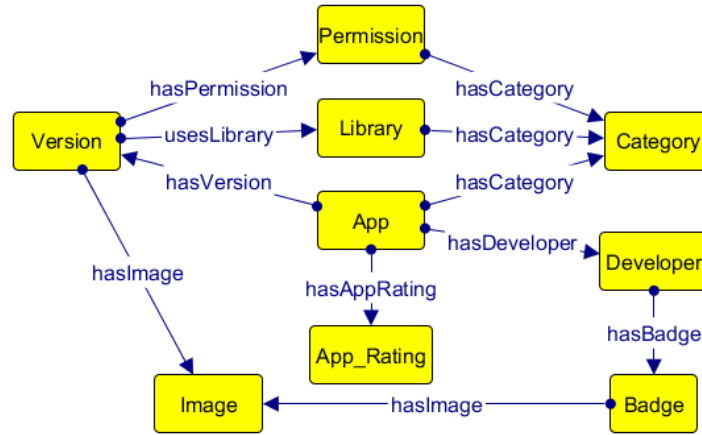
2.1   Ontology

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 ontology[7] 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 them[8]. 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

---

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 sources[9]. 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 dates[10]. 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 library[11], 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-201
          4-10-31-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 system[12] (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 library[13].

---

[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 website[14] (see Figure 5 for an example). To extract this information we developed a crawler based on the `crawler4j` library[15] 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.
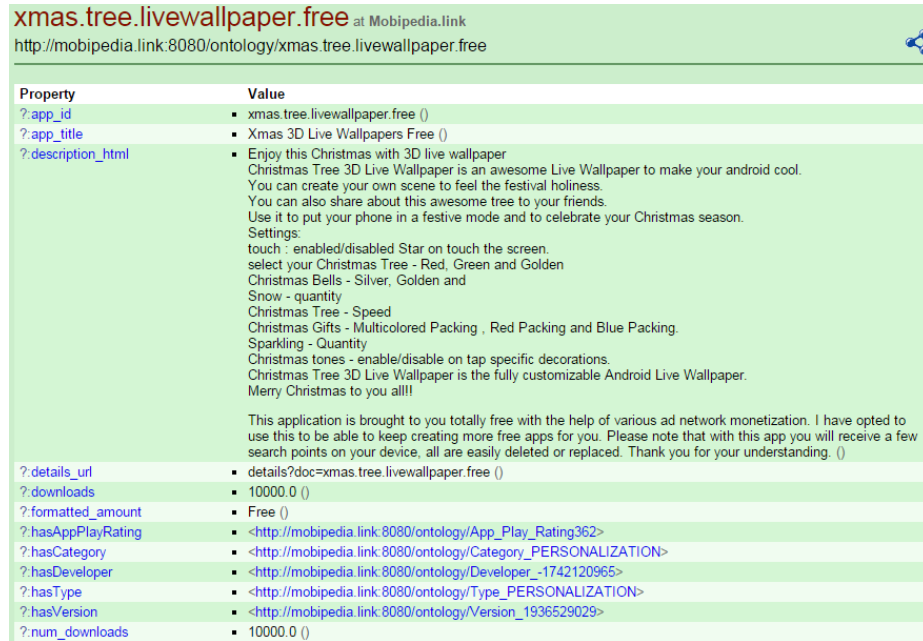
Linking Mobipedia with other Knowledge Bases. One of the requirements of Linked Data is to interlink the different knowledge bases available[16]. Based on this, we have interlinked Mobipedia with DBpedia [2], which is the nucleus of the Linked Data cloud. To achieve this, we executed queries against 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 project[17]. 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]).

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 server[19].

---

[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., 42matters[20]) 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 Data[21]. 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.

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)