Android Went Semantic: Time for Evaluation

Carlos Bobed, Fernando Bobillo, Roberto Yus, Guillermo Esteban, and Eduardo Mena

Dept. of Computer Science & Systems Engineering, University of Zaragoza, Spain
{cbobed,fbobillo,ryus,gesteban,emena}@unizar.es

Abstract. Applications for mobile devices could often show a more intelligent behavior by using a semantic reasoner to discover new knowledge. Unfortunately, using Description Logic reasoners on Android devices is not trivial. In this paper we continue our previous work on investigating the use of semantic reasoners on mobile devices. In particular, we port some new OWL 2 EL reasoners to Android and analyze the results of some experiments measuring the performance of several OWL 2 DL and OWL 2 EL reasoners on Android smartphones and tablets.

1 Introduction

Mobile devices are continuously increasing their importance in our daily lives. Currently there are lots of interesting applications (or *apps*) that take advantage of the context of the user (for instance, the geographical location). The incorporation of semantic technologies to enhance such applications seems promising, combining ontological background information with extensional data obtained from mobile sensors. For example, recommender systems (e.g., SHERLOCK [17]) can use semantic reasoners to infer interesting location-based services for mobile users. In our previous work [16], we analyzed whether smartphones and tablets are ready for this challenge. We focused our research on Android [3] devices due to its diffusion (80% of the smartphones according to a recent estimation[1]), its openness, and the existence of a native virtual machine that makes it easier to reuse existing Java applications such as most of the semantic APIs and reasoners.

While some Description Logic (DL) reasoners have been specially designed for mobile devices (Mini-ME [12], Delta [11], and Pocket KR Hyper [9]), our previous work shows that it is often possible to reuse existing semantic APIs and DL reasoners on Android [16] (the use of ELK on Android has also been studied [7]). Using semantic reasoners locally installed on the mobile device can be useful to preserve the privacy of users or minimize network connections.

This paper continues this line of research. In Section 2 we summarize our experiences porting semantic APIs and reasoners to Android. In particular, we consider more recent versions of already considered OWL 2 DL reasoners and some new reasoners specific for the OWL 2 EL profile. Next, in Section 3 we evaluate the performance of these reasoners on Android devices trying to investigate if there are differences between the behavior for these two OWL 2 fragments. Finally, Section 4 sets out some conclusions and ideas for future work.

[1] http://www.engadget.com/2013/10/31/strategy-analytics-q3-2013-phone-share

2 Reasoning on Android

Android uses a Java-like virtual machine called Dalvik which runs *dex-code* (Dalvik Executable). Java bytecodes can be converted to Dalvik-compatible .dex files to be executed on Android. However, Dalvik does not align to Java SE and so it does not support Java ME classes, AWT, or Swing. Thus, although most of the semantic APIs and DL reasoners are implemented in Java, porting them into Android requires some human intervention. This section reports our experiences with newer versions of the APIs and reasoners already considered in our previous work [16], and with two new reasoners.

Using Semantic Web APIs. OWL API [5] 3.4.10 can be imported into and used in an Android project directly. This is not the case of *Jena*, but it can be replaced by *Androjena*[2], a port of Jena to the Android platform. The last released version is version 0.5, which contains all the original code of Jena 2.6.2.

Revisiting already working reasoners. In [16] we presented how to get JFact, HermiT [13], Pellet [14], and CB [6] working on Android. Since then, new versions of some of them have been released. We revisited the reasoners to check if it is still possible to port them to Android. *JFact*, a port of FaCT++ [15] to Java, has reached its version 1.2.1, and can still be imported directly into an Android project. In our previous work, we used version 0.9.1, yet we have experienced problems using version 1.0.0. In Section 3 we will also describe some problems when using JFact 1.2.1. *Pellet* and *HermiT* reasoners have reached their versions 2.3.1 and 1.3.8, respectively. They can still be ported to Android similarly as we described in [16] (we omit the details here due to space restrictions). *CB* [6] has not received any update, so the approach we adopted is still valid: compiling the OCaml reasoner code to Android native code, and making it accessible using the Java Native Interface (JNI) and Android NDK.

Porting new reasoners. JCEL [10] is based on the CEL reasoner [2] and supports a subset of the OWL 2 EL profile. Indeed, some of the OWL 2 EL ontologies that we tried are not supported (see Section 3 for details); we used version 0.19.1, which can be imported into an Android project directly. *ELK* [8] reasoner supports the OWL EL profile. Version 0.4.0 cannot be directly used in Android projects: The *Log4j* library imported by the reasoner is not supported on Android and has to be replaced by the *log4j-over-slf4j*[3] library.

3 Experimental Evaluation

In this section we firstly detail our ontology dataset and then we present the results of our experiments.

[2] `https://code.google.com/p/androjena`
[3] `http://www.slf4j.org/legacy.html`

3.1 Ontology dataset

To evaluate the performance of the studied reasoners, the first idea was to use directly the ORE 2013 ontology set [4], with 200 ontologies from the NCBO BioPortal, the Oxford Ontology Library, and the Manchester Ontology Repository. Every ontology has at least 100 logical axioms, and 10 named concepts, and ontologies are grouped, according to the number of logical axioms, as *small* (≤ 500), *medium* (between 500 and 4999), and *large* ontologies (≥ 5000).

In our case, we had to take into account the restrictions that mobile devices suffer from when compared to a desktop computer, specially the limited CPU, memory, and battery. For this reason, we selected a subset of the ORE 2013 ontology set carrying out the following steps:

1. We order the ontologies according to the size of the file instead of the number of logical axioms. We cannot ignore annotation axioms as they are problematic in our scenario: They also have to be loaded by the OWL API and, thus, they consume memory.
2. The maximum heap size per application usually (currently) provided by Android is 256 MB. This could be the theoretical maximum size of an ontology loaded on Android, but we must build instances of the OWL API class OWLReasoner in the device's memory. So, we applied a first filter to each ontology dataset to take out the ontologies whose files occupied more than 128 MB. This resulted in 186 DL ontologies and 193 EL ontologies.
3. To keep the number of tests manageable, we sampled these filtered ontologies to get just 10 of each profile OWL 2 DL and OWL 2 EL (the supported languages of our ported reasoners). To do so as fair as possible, we ordered them by size of their OWL file, and took one ontology every 186/10 or 193/10, respectively. We did it this way as the number of axioms of the ontology and the size of the file are directly related.

Our DL ontology dataset has 3 small (DL1–DL3), 5 medium (DL4–DL8), and 2 large ontologies (D9–D10), whereas our EL ontology dataset has 5 small (EL1–EL5), 3 medium (EL6–EL8), and 2 large ontologies (E9–E10). The detailed list of ontologies in our datasets can be found at [1].

3.2 Experiments and discussion

Checking the ported versions. Our first type of experiments was designed to test if the reasoners produce the same results in Android devices as in desktop computers. On the one hand, we wanted to check the behavior of the Android-compatible libraries we used to replace unsupported ones (for HermiT, Pellet, and ELK). On the other hand, we wanted to check the behavior of the reasoners that can be directly imported in Android projects (JFact and JCEL). We considered three devices: a Galaxy Nexus smartphone (Android 4.2.1, 1.2 GHz dual-core, 1 GB RAM, denoted A1), a Galaxy Tab 2.7.0 tablet (Android 4.1.2, 1 GHz dual-core, 1 GB RAM, denoted A2), and a desktop computer (PC, Windows 64-bits, i5-2320 3.00 GHz, 16 GB RAM).

We computed the classification of the 20 ontologies in our dataset checking if every reasoner obtained the same sets of subsumption relations. All of the reasoners except one indeed had the expected behavior but, unfortunately, it turned out that JFact 1.2.1 produced different results in the Android devices and in the PC, although this does not happen with JFact 0.9.1[16]. Firstly, ontology DL4 is correctly identified as consistent by the PC version, but is inconsistent according to the Android version. Secondly, in ontology DL7, the Android version misses two subclasses of the class `http://www.loa-cnr.it/ontologies/DUL.owl#Role`, namely `http://www.loa-cnr.it/ontologies/DUL.owl#Entity` and `http://cidoc.ics.forth.gr/cidoc_v4.2.owl#E1.CRM_Entity`.

Comparing previous versions. The next step was to check whether the new versions of HermiT, Pellet, and JFact have a similar performance to the older ones [16]. To this end, we tested the classification performance (i.e., time needed to compute the class subsumption hierarchy) for 5 well-known ontologies not included in our current ontology dataset but tested in [16]: Pizza[4], Wine[5], DBpedia 3.8[6], GO[7], and NCI[8]. Indeed, different versions of HermiT (1.3.8 and 1.3.6), Pellet (2.3.0 and 2.3.1), and JFact (1.2.1 and 0.9.1) perform similarly.

Comparing profiles for ontology classification. Finally, we investigated the performance on the different profiles by classifying the ontologies in our dataset for every reasoner supporting the given profile. We repeated the experiments three times for every ontology and computed the mean value. We noticed that the time variance is small except for large ontologies. Some of the reasoners were not able to classify some ontologies (DL4, DL6, DL7, DL8, DL10, EL6, EL8, and E10). In these cases, we did not consider that test for any of the devices where the test was run. Apart from the problems with JFact in ontologies DL4 and DL7, we found the following problems:

- Loading error in A1, A2, and PC: JCEL (in EL6 and EL8).
- Stack overflow in A1 and A2: JFact (in EL8) and Pellet (in DL6 and DL7).
- Out of memory in A1 and A2: Pellet (in E10).
- Time out (after 30 minutes) in A1 and A2: ELK (in E10), Pellet (in DL8), and all the reasoners (Pellet, HermiT, and JFact) in DL10. It is worth noting that Pellet also produced a time out (after 10 minutes) in PC.

The results are summarized in Figures 1 and 2 for OWL DL and OWL EL, respectively. Note that the scale is logarithmic. Due to space limitations, we show the mean of the classification times for the groups of small, medium, and large ontologies, but the results for every single ontology are available online [1].

The figures indicate, for every reasoner and ontology size, the number of ontologies that are considered in the displayed chart. Unsurprisingly, we can see

[4] `http://www.co-ode.org/ontologies/pizza/pizza.owl`
[5] `http://www.w3.org/TR/owl-guide/wine.rdf`
[6] `http://dbpedia.org/Ontology`
[7] `http://www.geneontology.org`
[8] `http://bioportal.bioontology.org/ontologies/NCIT`

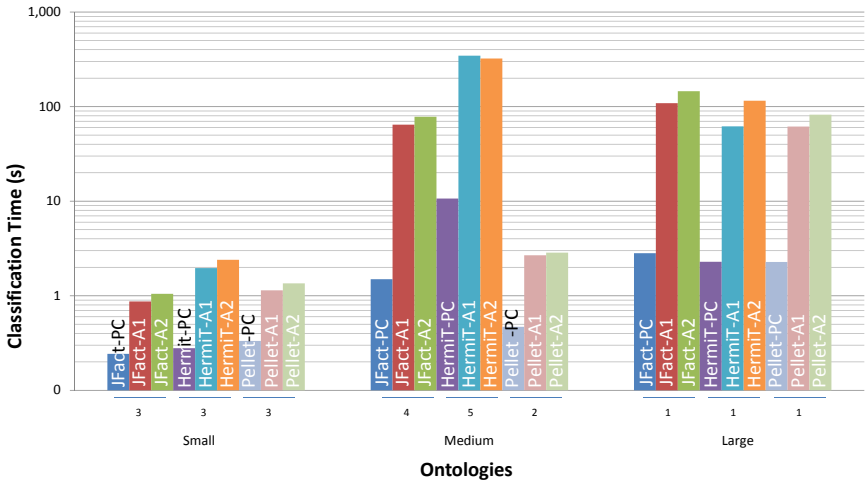

Fig. 1. Results of the experiments in OWL 2 DL ontologies.

that reasoners are much faster in the PC. Table 1 shows the number of times of the PC version being faster than the Android ones. It is worth noting that Pellet seems faster for medium OWL 2 DL ontologies, but it is due to the fact that the chart is only displaying the data for two ontologies. We can also see that the smartphone (A1) is a 7-34 % faster than the tablet (A2) in all cases except one. Furthermore, the larger the ontology and the more expressive profile, the more significant difference with respect to the PC version. In these cases time outs and stack overflows are more frequent in Android and we have displayed less data in the charts (see details it [1]).

We would like to add some final remarks. We considered measuring memory and battery usage as well, but we found several difficulties to do that accurately on Android devices. Moreover, we have not included CB reasoner in the experiments because the comparison with the other reasoners would not be fair: We have focused on evaluating the reasoners running directly within Dalvik virtual machine while CB would execute native code through the use of JNI.

Table 1. Number of times of the PC version being faster than the Android ones.

Reasoner		ELK		JCEL		JFact		HermiT		Pellet	
Device		A1	A2	A1	A2	A1	A2	A1	A2	A1	A2
OWL 2 DL	Small	-	-	-	-	3.6	4.3	7.0	8.6	3.4	4.0
	Medium	-	-	-	-	43.0	52.2	32.4	30.3	5.7	6.0
	Large	-	-	-	-	38.6	51.7	27.0	50.5	27.0	36.0
OWL 2 EL	Small	1.5	2.0	11	11	3.7	4.1	15	22	2.7	3.1
	Medium	4.4	5.3	16	17	11	12	18	22	15	17.5
	Large	11	17	39	79	47	54	102	299	11	22.4

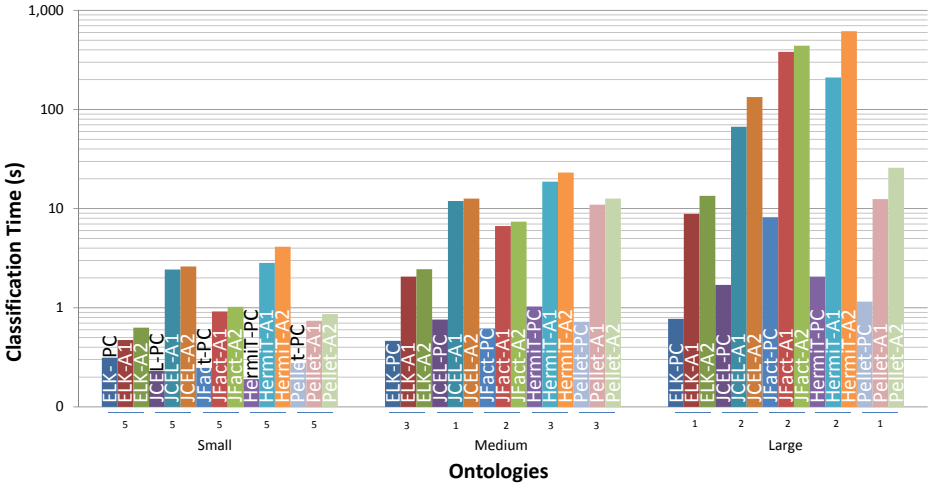

Fig. 2. Results of the experiments in OWL 2 EL ontologies.

4 Conclusions and Future Work

Our ongoing work shows that using semantic reasoners on mobile devices is far from being trivial. We have discussed how to support some reasoners on Android devices, being able to port some new ones (JCEL and ELK). Unfortunately, it turns out that newer versions of some reasoners (HermiT and Pellet) are not easier to port than the older versions we considered one year ago [16]. We have also experienced some new errors with the latest version of JFact.

Our experiments show a worse performance on Android devices, with frequent time outs and out of memory errors that are more usual in the OWL 2 DL profile and in larger ontologies, where there are less ontologies such that every reasoner is able to complete the classification. We also noticed important differences (34 % in some cases) in the performance of the two analyzed Android devices.

The complete results of our experiments can be found at the webpage [1], together with a detailed description of all the changes needed to port the semantic reasoners and, if the licenses make it possible, a download link.

Future work will mainly be focused on a more complete evaluation of the performance of semantic reasoners on Android devices. We plan to consider more semantic reasoners (we are trying to port TReasoner, TrOWL, MORe, Snorocket, and Quest), reasoning tasks, and ontologies (those mentioned at Section 3). Finally, we would like to understand better the reasons of the differences in the performance. To do so, we have designed some experiments to evaluate the role of the amount of memory (by restricting the PC to work with the same memory as the mobile devices) and the alternative libraries (by testing them in the PC).

Acknowledgments. This research work has been supported by the CICYT project TIN2010-21387-C02-02 and DGA-FSE.

References

1. Android goes semantic! `http://sid.cps.unizar.es/AndroidSemantic`.
2. F. Baader, C. Lutz, and B. Suntisrivaraporn. CEL - A polynomial-time reasoner for life science ontologies. In *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR 2006)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 287–291. Springer-Verlag, 2006.
3. E. Burnette. *Hello, Android: Introducing Google's Mobile Development Platform*. The Pragmatic Programmers, LLC., 2010.
4. R. S. Gonçalves, S. Bail, E. Jiménez-Ruiz, N. Matentzoglu, B. Parsia, B. Glimm, and Y. Kazakov. OWL Reasoner Evaluation (ORE) workshop 2013 results: Short report. In *Proceedings of the 2nd International Workshop on OWL Reasoner Evaluation (ORE 2013)*, volume 1015, pages 1–18. CEUR Workshop Proceedings, 2013.
5. M. Horridge and S. Bechhofer. The OWL API: A Java API for OWL ontologies. *Semantic Web Journal*, 2(1):11–21, 2011.
6. Y. Kazakov. Consequence-driven reasoning for Horn $\mathcal{SHIQ}$ ontologies. In *Proceedings of the 21st International Joint Conference on Artificial intelligence (IJCAI 2009)*, 2009.
7. Y. Kazakov and P. Klinov. Experimenting with ELK reasoner on Android. In *Proceedings of the 2nd International Workshop on OWL Reasoner Evaluation (ORE 2013)*, volume 1015 of *CEUR Workshop Proceedings*, pages 68–74, 2013.
8. Y. Kazakov, M. Krötzsch, and F. Simančík. The incredible ELK. *Journal of Automated Reasoning*, 53:1–61, 2014.
9. T. Kleemann. Towards mobile reasoning. In *Proceedings of the 2006 International Workshop on Description Logics (DL 2006)*, 2006.
10. J. Mendez. jcel: A modular rule-based reasoner. In *Proceedings of the 1st International Workshop on OWL Reasoner Evaluation (ORE 2012)*, volume 858 of *CEUR Workshop Proceedings*, 2012.
11. B. Motik, I. Horrocks, and S. M. Kim. Delta-reasoner: A Semantic Web reasoner for an intelligent mobile platform. In *Proceedings of the 21st World Wide Web Conference (WWW 2012), Companion Volume*, pages 63–72, 2012.
12. M. Ruta, F. Scioscia, G. Loseto, F. Gramegna, and E. D. Sciascio. A mobile reasoner for semantic-based matchmaking. In *Proceedings of the 6th International Conference on Web Reasoning and Rule Systems (RR 2012)*, 2012.
13. R. Shearer, B. Motik, and I. Horrocks. HermiT: A Highly-Efficient OWL Reasoner. In *Proceedings of the 5th International Workshop on OWL: Experiences and Directions (OWLED 2008)*, 2008.
14. E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, 5(2):51–53, 2007.
15. D. Tsarkov and I. Horrocks. FaCT++ description logic reasoner: system description. In *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR 2006)*, 2006.
16. R. Yus, C. Bobed, G. Esteban, F. Bobillo, and E. Mena. Android goes semantic: DL reasoners on smartphones. In *Proceedings of the 2nd International Workshop on OWL Reasoner Evaluation (ORE 2013)*, volume 1015, pages 46–52. CEUR Workshop Proceedings, 2013.
17. R. Yus, E. Mena, S. Ilarri, and A. Illarramendi. SHERLOCK: Semantic management of location-based services in wireless environments. *Pervasive and Mobile Computing*, In press.