An Initial Investigation on Evaluating Semantic Web Instance Data

Li Ding
dingl@cs.rpi.edu

Jiao Tao
taoj2@cs.rpi.edu

Deborah L. McGuinness
dlm@cs.rpi.edu

Tetherless World Constellation, Computer Science Department
Rensselaer Polytechnic Institute
110 8th Street, Troy, NY 12180

ABSTRACT

Many emerging semantic web applications include ontologies from one set of authors and instance data from another (often much larger) set of authors. Often ontologies are reused and instance data is integrated in manners unanticipated by their authors. Not surprisingly, many instance data rich applications encounter instance data that is not compatible with the expectations of the original ontology author(s). This line of work focuses on issues related to semantic expectation mismatches in instance data. Our initial results include a customizable and extensible service-oriented evaluation architecture, and a domain implementation called PmlValidator, which checks instance data using the corresponding ontologies and additional style requirements.

Categories and Subject Descriptors: D.2.5 [Software/Software Engineering]: Testing and Debugging

General Terms: Design, Verification

Keywords: architecture, instance data evaluation, semantic web

1. INTRODUCTION

Increasingly semantic web applications are consuming data from different sources. It is desirable to automatically detect errors and potential issues in semantic web data before using the data. Semantic web instance data is growing and is dominating the Semantic Web on the web[2].

In this work, we identify a new research problem **semantic web instance data evaluation**. The work, while closely related to ontology evaluation, is different in that it focuses on potential and actual compatibility of the instance data with the term definitions in the corresponding ontologies. Unlike ontology evaluation which checks logical consistency of the term definitions, this effort includes issues of representation style in addition to provable errors of syntax and semantics.

We found no existing work dedicated to semantic web instance data evaluation. Previous related work aimed at general knowledge base environments mainly for ontology evaluation (e.g., [6, 4, 1, 3]). Related work on OWL DL ontology debugging (e.g., [7, 10, 9]) concentrates on diagnosing semantic consistency issues.

Semantic web instance data evaluation raises two challenges: (i) to identify issues in instance data; and (ii) to develop a customizable and extensible approach to meet the diverse evaluation requirements required by different semantic web applications.

2. EVALUATION ARCHITECTURE

WWW 2008, April 21–25, 2008, Beijing, China.
ACM 978-1-60558-085-2/08/04.

Figure 1 depicts our service-oriented architecture. The semantic web instance data evaluation process is composed from interactions of independent evaluation services that communicate using a simple data interface. Each service focuses on one evaluation task and generates the corresponding evaluation report. An evaluation report entry typically covers the severity, symptom diagnosis, and optional repair instructions.

In order to fulfill the separation requirement, we need to identify the scope of instance data and ontologies. Our evaluation architecture differentiated three collections of semantic web data: the *instance data* to be evaluated, the automatically-loaded *referenced ontologies* that define the classes and properties being instantiated in the instance data, and the user-provided *optional ontologies* that add definitions and restrictions to the referenced ontologies. Instance data evaluation considers only the issues introduced by the *instance data* and ignores the issues that previously existed within the *referenced ontologies* and/or the *optional ontologies*.

We identified six types of services under the three main categories suggested by previous work on ontology evaluation[7, 9] .

- structural (syntactic) issues: *RDF syntax parsing and validation* for parsing the *instance data* and the user-provided *optional ontologies*; *referenced ontology resolution* for collecting the *referenced ontologies* from the *instance data*; and the syntactic aspects of *OWL species classification*.
- logical (semantic) issues: *RDFS/OWL semantics validation* for verifying RDFS and OWL Lite/DL/Full semantics.
- user-defined (style) issues: *general style evaluation* for issues such as cardinality issues, and *domain specific style evaluation* for issues specific to a certain domain ontology.

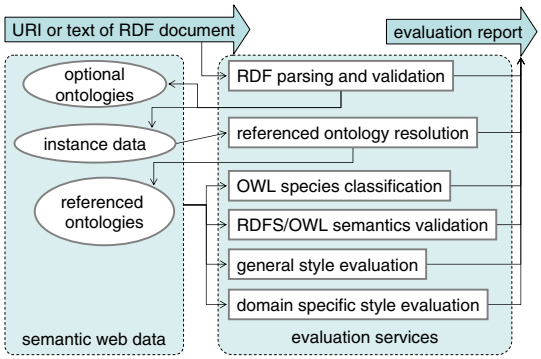

Figure 1: Evaluation Service Architecture

Our service-oriented architecture is designed to be customizable and extensible. The two topmost types of service in Figure 1 must be executed first in sequence, and the remaining services, which take the same input, can be executed in any combination and any order. Moreover, our architecture is extensible, for example, an *OWL species classification* service can be implemented using different OWL reasoners; and new user-defined services for *domain specific style evaluation* can be added as plug-ins.

3. IMPLEMENTATION: PMLVALIDATOR

In our Inference Web[5] project, we have implemented our architecture in a tool called *PmlValidator* for evaluating instance data encoded using the Proof Markup Language (PML)[8]. PmlValidator implemented and integrated evaluation services at the JAVA API level, and it is available online as a web service at `http://onto.rpi.edu/iw2api/doc_pmlvalidator`.

PML instance data is encoded using PML ontologies. Since the PML ontologies use only OWL DL, we can leverage many existing tools to provide syntax and semantics evaluation services. *RDF parsing and validation* is implemented using Jena (`http://jena.sourceforge.net/`), which is also used by W3C RDF Validation Service (`http://www.w3.org/RDF/Validator/`). *OWL species classification* and *OWL DL semantics validation* are implemented using the OWL DL reasoner Pellet (`http://pellet.owldl.com/`).

Our current implementation for *referenced ontology resolution* uses the following heuristics: the *referenced ontologies* are (i) ontologies linked by the namespace of classes and properties instantiated in the instance data, and (ii) ontologies recursively imported by the referenced ontologies.

PmlValidator also checks style issues beyond the reach of OWL DL reasoners. Figure 2 provides several examples to illustrate the following three issues checked by our *general style evaluation*.

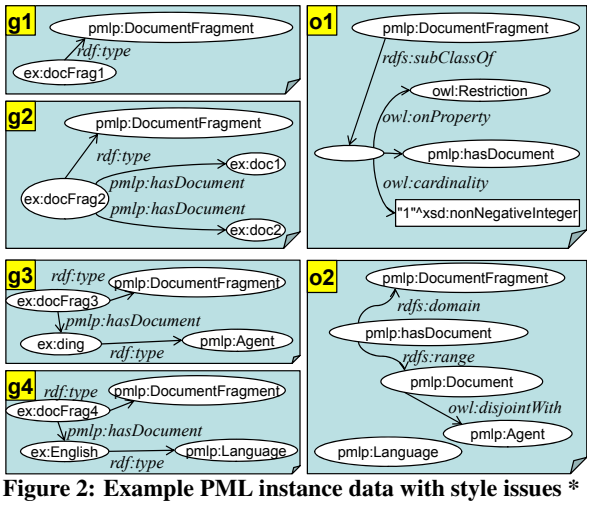

Figure 2: Example PML instance data with style issues *

* *g1*, *g2*, *g3* and *g4* are four independent collections of PML instance data; and *o1* and *o2* are the corresponding definitions from PML ontologies.

(i) **Issues related to min cardinality restrictions**. OWL DL reasoners, using the open world assumption, will not report any inconsistencies in *g1* based on *o1* because they assume the expected RDF triples may be found outside *g1*; but style issues (e.g. missing expected RDF triples) may be reported when *g1* is used by applications using the closed world assumption, e.g. database applications that require non-null fields.

(ii) **Issues related to max cardinality restrictions**. OWL DL reasoners will not report any inconsistencies in *g2* based on *o1* because they cannot find *different from* assertion in *g2*; but style issues (e.g., existence of unwanted RDF triples) may be reported when *g2* is used by applications using the *unique names assumption*, i.e. any two individuals with different names are different from each other unless they have been explicitly asserted to be the same.

(iii) **Issues related to multiple class-memberships**. OWL DL reasoners can, for example, infer that *ex:docFrag3* is a member of both *pmlp:Agent* and *pmlp:Document* in *g3* using the *rdfs:range* statement in *o2*. They will only confirm inconsistency in *g3* but not in *g4* because no relation between *pmlp:Language* and *pmlp:Document* has been found in *o2*. While it might be unnecessary to exhaust all pairs of classes without any asserted relations such as *rdfs:subClassOf* and *owl:disjointWith*, we have found it a useful precaution to check for relationships, particularly those generated from local property value restrictions and global domain and range restrictions.

Pml Validator also adds the following service for *domain-specific style evaluation*: (i) *existence of PML instance data*: whether the instance data contains at least one instantiation of a class or a property defined in the PML ontologies. (ii) *anonymous URI*: instances of some specific PML classes must not be anonymous.

4. CONCLUSION

Semantic web instance data evaluation brings unique and important benefits to semantic web applications: data publishers may detect and thus fix outstanding issues in the instance data; and data consumers may verify their expectations for incoming semantic web instance data and to customize and extend the domain specific evaluation services if necessary. The above style evaluation services have been implemented and tested in PmlValidator. Future work includes evaluation experiments and performance study of PmlValidator on large scale instance data, implementation extension and improvement to cover more style issues in instance data.

Acknowledgement: This work is supported by NSF #5710001895, DARPA #F30602-00-2-0579, #55-300000680 to-2 R2.

5. REFERENCES

[1] K. Baclawski, C. J. Matheus, M. M. Kokar1, J. Letkowski, and P. A. Kogut. Towards a symptom ontology for semantic web applications. In *ISWC*, pages 650–667, 2004.

[2] L. Ding and T. Finin. Characterizing the semantic web on the web. In *ISWC*, pages 242–257, 2006.

[3] A. Gangemi, C. Catenacci, M. Ciaramita, and J. Lehmann. Modelling ontology evaluation and validation. In *ESWC*, pages 140–154, 2006.

[4] A. Gómez-Pérez. Evaluation of ontologies. *International Journal of Intelligent Systems*, 16(3):391–409, 2001.

[5] D. L. McGuinness and P. P. da Silva. Explaining answers from the semantic web: the inference web approach. *Journal of Web Semantics*, 1(4):397–413, 2004.

[6] D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder. An environment for merging and testing large ontologies. In *KR*, pages 483–493, 2000.

[7] B. Parsia, E. Sirin, and A. Kalyanpur. Debugging owl ontologies. In *WWW*, pages 633–640, 2005.

[8] P. Pinheiro da Silva, D. L. McGuinness, and R. Fikes. A proof markup language for semantic web services. *Information Systems*, 31(4):381–395, 2006.

[9] P. Plessers and O. D. Troyer. Resolving inconsistencies in evolving ontologies. In *ESWC*, pages 200–214, 2006.

[10] H. Wang, M. Horridge, A. Rector, N. Drummond, and J. Seidenberg. Debugging owl-dl ontologies: A heuristic approach. In *ISWC*, pages 745–757, 2005.