Received: 23 November 2018 | Revised: 14 December 2018 | Accepted: 14 December 2018

DOI: 10.1002/itl2.87

USE CASE LETTER

WILEY

Checking compliance of semantic web applications with RDFS-semantics

Simona Colucci[1] | **Francesco M. Donini**[2] | **Eugenio Di Sciascio**[1]

[1]DEI, Politecnico di Bari, Bari, Italy
[2]DISUCOM, Università degli Studi della Tuscia, Viterbo, Italy

Correspondence
Simona Colucci. DEI, Politecnico di Bari, Via Orabona 4, 70125 Bari, Italy.
Email: simona.colucci@poliba.it

Web applications calling themselves "Semantic" count in the hundreds, yet there is no clear definition about what this qualification should stand for. Semantic web applications may range from those just using Resource Description Framework (RDF) as a data interchange format, to those taking all RDF-S consequences into account. Moreover, even with the simplest semantics, RDF data may contain blank nodes, which should be treated appropriately as existential variables—but might not. In this paper, we propose a general framework of yes/no experiments whose result tell, in a black-box fashion, how "Semantic" can be considered a given Web application. Our experiments measure the sensitivity of the application to syntactic variations of data which are equivalent under RDF-S-semantics. We show how our experiments can be run on a real application, namely, RapidMiner with LODExtension. We show how RapidMiner passes most of the tests (but for blank nodes), but highlight a weakness of the workflow repository-retrieval-exploitation that may make the application fail in some cases.

KEYWORDS

RDF, RDF blank nodes, RDF-S semantics, semantic web

1 | INTRODUCTION

There is an obvious consensus about how to answer the question: "Does the application X uses technology Y?" for technologies like, say, Relational Databases, NoSQL, or Answer Set Programming. Maturity of such technologies brings off-the-shelf, monolithic solutions, which must be adopted from the design phase, qualifying the application as one using such a technology.

Semantic web (SW) technology, in our viewpoint, shows a much more "coastal wetland" boundary, in which an application can—to make a progressive list only for this paper:

1. access Resource Description Framework (RDF) data in the Web and treat them as pure database triples (skolemizing blank nodes).
2. treat blank nodes like SQL Null values.
3. add to the retrieved RDF triples all consequences derived using RDF-S semantics.
4. take RDF-S predicates into account without adding consequences.

A delicate point is how data can be accessed, or how they are made available by the data repository. Usually either an API is provided, or a SPARQL endpoint is accessible via Hyper Text Transfer Protocol. However, SPARQL endpoints tend not to use the RDF-S entailment regime,[1] which means that RDF triples are returned to an application without taking the consequences of RDF-S predicates into account. For instance, the two triples* "`a p b .`" and "`p rdfs:range C .`" imply in RDF-S -semantics the triple "`b rdf:type C .`", but many SPARQL endpoints would not retrieve such a triple—since they use the Simple Entailment regime.

This leaves to the application—if ever—the burden of taking RDF-S-consequences into account, in some way, in the results of the application itself. The question whether this should be done by the application is subtle, since it depends on whether the technologies for the next step in the workflow—namely, analyzing data—assume completeness of data, or do not. Available

Kernel methods,[3] for instance, assume complete information. This implies that both: (a) the lack of information about blank nodes is resolved somehow, and (b) all facts are explicit: for instance, to decide whether an element *b* belongs or not to a given class *C*, either "`b rdf:type C`." is retrieved, or the contrary can be assumed for the purpose of computing a kernel.

Hence, if RDF-S-semantics is not taken into account by the application, it may yield counterintuitive results when apparently redundant triples (as the one above) are added to the data. Such results lower the trust a user can have in the application itself.

We clarify that our purpose is not to point to faults either of SW applications, or of the workflow repository-retrieval-exploitation. Our point is measuring how much SW technologies are exploited. To this end, we propose a set of experiments that take the entire workflow as a black box, and assess whether some semantically equivalent syntactic variations produce differences in a SW application. Each experiment, with its yes/no answer, can add—or subtract—a bit about how much "Semantics" there is in the SW application.

In some way, we are paving the way to a maturity model for SW applications. Recently, maturity models have been proposed as a benchmark for best practices in heterogeneous fields, ranging from project and knowledge management to information technology (IT) and big data. Among the most recognized models in IT, the Richardson Maturity Model[4] classifies Web APIs in four maturity levels according to their design. A maturity model for Semantic RESTful Web APIs, called WS^3, has also been proposed.[5] WS^3 introduces a three-dimensional classification for Web APIs, embedding a so-called *Semantic dimension*. APIs are mature according to the Semantic dimension, if both resources and relationships among them are semantically described.

The paper continues with a general illustration of the rationale of each experiment, while in the subsequent Section 3, their implementation over a realistic SW application—extraction of some *DBPedia* triples, and use RapidMiner with LODExtension to compute a distance between similar resources—is conducted. Concluding section (Section 4) discusses the results.

2 | EXPERIMENTS DESIGN

The rationale of our experiments follows from an observation about the level of realization of the SW initiative. As a matter of fact, the SW literature gathers several applications of SW technologies. Some of them are recognized by the World Wide Web Consortium (W3C) as tools (https://www.w3.org/2001/sw/wiki/Tools) or case studies (https://www.w3.org/2001/sw/sweo/public/UseCases), and classified into categories, to make their retrieval easier. Nevertheless, the criterion according to which applications are evaluated as compliant with the SW is not explicit. In most cases, the use of any SW technology is enough for reaching the status of SW application. Yet, not all applications using SW technologies are really aware of the knowledge embedded in the data sources they manage.

As an example, many applications use RDF[2] data sets as data source, without exploiting their potential in terms of provided knowledge. In fact, RDF data sets are often used more like data structures than like knowledge bases: they are mined to extract information that are treated just as plain data. Of course RDF was not meant for this in the original SW vision, but it was conceived to ensure the interchange of data, together with the meaning embedded in them. Therefore, we believe that an application using RDF data sets as knowledge source should be put on a scale to test its compliance as a SW application.

In particular, at least the following features may be used to measure SW-compliance: (a) *Sensitivity to syntactic changes in the source RDF data set*: a SW application should produce the same results when syntactically different, but semantically equivalent, data sets are used as knowledge source; (b) *Handling of anonymous resources*: a SW application should treat blank nodes according to their definition in RDF—as existential variables whose scope is the file they appear in.

In order to check how well applications fulfill two requirements above, we designed a set of experiments, discussed below. From now on, we denote a generic SW application by *swa*, the RDF dataset by D and the result produced by the application by $res_{swa}(D)$. In the description of each experiment, we refer to a pair of data patterns D_1 and D_2. When implementing the experiments, the used data sources have to include triples matching such data patterns. All examples of RDF triples in the paper follow Turtle[6] syntax and the prefixes below:

rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs: http://www.w3.org/2000/01/rdf-schema#
dbr: http://dbpedia.org/resource/
dbo: http://dbpedia.org/ontology/

2.1 | Goal 1: Checking sensitivity to syntactic changes in RDF-S-equivalent input data sets

The experiment works by defining pairs of RDF data sets (D_1, D_2), such that D_1 and D_2 are equivalent with respect to (w.r.t.) RDF-S-semantics, even though they syntactically differ from each other. An application defined as "semantic" should be aware of this equivalence, and ignore any merely syntactical difference. We designed four experiments, choosing, for the

sake of brevity, the RDF-S rules we consider more significant. The experiments involve the RDF data patterns shown below:

Experiment 1 (rdfs:subClassOf)	D_1	x `rdfs:subClassOf` u. u `rdfs:subClassOf` z.
	D_2	x `rdfs:subClassOf` u. u `rdfs:subClassOf` z. x `rdfs:subClassOf` z.
Experiment 2 (rdfs:range)	D_1	x p u. p `rdfs:range` z.
	D_2	x p u. p `rdfs:range` z. u `rdf:type` z.
Experiment 3 (rdfs:domain)	D_1	x p u. p `rdfs:domain` z.
	D_2	x p u. p `rdfs:domain` z. x `rdf:type` z.
Experiment 4 (rdfs:subPropertyOf)	D_1	x p u. p `rdfs:subPropertyOf` q.
	D_2	x p u. p `rdfs:subPropertyOf` q. x q u.

In all the four experiments above, for any SW application *swa* it should hold: $res_{swa}(D_1) = res_{swa}(D_2)$.

2.2 | Goal 2: checking correct interpretation of blank nodes

Blank nodes are often managed by just giving them a unique symbolic name (a Skolem constant), forgetting about the logical nature of such special resources. Anonymous resources are, in fact, existential variables which could be bound to any resource. This kind of binding would lead in some cases to more accurate results, as we show later in our examples. In order to test applications w.r.t. this specific ability, we designed an experiment, involving the following RDF data patterns:

Experiment 5 (blank nodes)	D_1	x p u. y p u. z p w.
	D_2	x p `_:b1`. y p u. z p w.

Comparing resources x, y, and z in D_1, and assuming completeness of data, one can observe that x and y appear to be more similar to each other, than they are to z. In D_2 instead, while y and z are as dissimilar as before, x could be similar to y, or to z, or to none of them, depending on the intepretation of `_:b1`. Hence, for applications computing, say, pairwise similarity, or clusterization, one can expect that resource x is treated in $res_{swa}(D_2)$ in some intermediate way w.r.t. how it is treated in $res_{swa}(D_1)$. If instead x, y, and z are treated in $res_{swa}(D_2)$ as equally dissimilar to each other—as it would be in the case the blank node is just skolemized—the experiment assesses that the application does not treat blank nodes according to their semantics.

Observe that the treatment of blank nodes is completely orthogonal to RDF-S-semantics, since blank nodes appear already in the most basic semantics of RDF.

3 | EXPERIMENTS IMPLEMENTATION

We briefly show how the designed experiments may be implemented in an example application, working just as a showcase for the general validity of the proposed method.

3.1 | Dataset used

We introduce the pairs of data sources D_1 and D_2 used for each experiment. All triples were extracted from DBPedia (http://dbpedia.org/sparql/), as of September 11, 2018. In particular, Experiments 1 to 4 use the same set

```
D1={dbo:championInSingleFemale rdfs:subPropertyOf dbo:championInSingle;
                               rdfs:range dbo:Athlete;
```

```
                                    rdfs:domain dbo:SportsEvent .
dbo:TennisTournament rdfs:subClassOf dbo:Tournament .
dbo:Tournament rdfs:subClassOf dbo:SportsEvent .
dbr:2015_Citi_Open rdf:type dbo:TennisTournament;
   dbo:championInSingleFemale dbr:Sloane_Stephens;
   dbo:championInDoubleFemale dbr:Belinda_Bencic,
                              dbr:Kristina_Mladenovic .
dbr:2016_Citi_Open rdf:type dbo:TennisTournament;
   dbo:championInSingleFemale dbr:Yanina_Wickmayer;
   dbo:championInDoubleFemale dbr:Yanina_Wickmayer,
                              dbr:Monica_Niculescu .}
```

A different D_2 for each of the first four experiments is used:

- *Experiment 1*: $D_2 = D_1 \cup$ {`dbo:TennisTournament rdfs:subClassOf dbo:SportsEvent .`}
- *Experiment 2*: $D_2 = D_1 \cup$ {`dbr:Yanina_Wickmayer rdf:type dbo:Athlete .`}
- *Experiment 3*: $D_2 = D_1 \cup$ {`dbr:2016_Citi_Open rdf:type dbo:SportsEvent .`}
- *Experiment 4*: $D_2 = D_1 \cup$ {`dbr:2016_Citi_Open dbo:championInSingle dbr:Yanina_Wickmayer .`}

The reader can verify that in all the experiments, D_1 and D_2 are equivalent w.r.t. RDF-S-semantics.[2]

Experiment 5, instead, uses a different pair of data sources. In particular, D_1 is given below (all triples extracted from DBPedia, as of September 11, 2018):

```
{dbr:2015_Citi_Open dbo:championInDoubleFemale dbr:Belinda_Bencic,
                                               dbr:Kristina_Mladenovic .
dbr:2016_French_Open dbo:championInDoubleFemale dbr:Kristina_Mladenovic,
                                               dbr:Caroline_Garcia .
dbr:2016_Citi_Open dbo:championInDoubleFemale dbr:Yanina_Wickmayer,
                                               dbr:Monica_Niculescu .}
```

On the contrary, D_2 has been created for illustrative purposes, starting from the informative content in DBPedia.

```
{dbr:2015_Citi_Open dbo:championInDoubleFemale dbr:Belinda_Bencic,
                                                :winner2 .
dbr:2016_French_Open dbo:championInDoubleFemale dbr:Kristina_Mladenovic,
                                                dbr:Caroline_Garcia .
dbr:2016_Citi_Open dbo:championInDoubleFemaledbr:Yanina_Wickmayer,
                                                dbr:Monica_Niculescu .}
```

Data set D_2 differs from D_1 just in the winner team of the ladies' doubles for resource `dbr:2015_Citi_Open`: in D_2 one of the two winners is unknown and, thus, represented by an anonymous resource: `_:winner2`. Although built for example purpose, D_2 is realistic: for example, in data sets created by combining different sources, the presence of unknown information is rather frequent.

3.2 | Similarity between RDF resources

In our past research,[7] we described an application for computing the Euclidean distance[†8] of two RDF resources. The application is a process workflow in the well-known Machine Learning tool RapidMiner,[9] and adopts the Linked Open Data extension (LODExtension)[10] of RapidMiner to import data from RDF data sources. Without delving into implementation details (the interested reader may refer to our previous work), we here show in Table 1 the results of the application w.r.t. the pairs (D_1, D_2) introduced above.

Notably, choosing which triples form D_1 and D_2 is a highly subjective task, that requires a deep knowledge both of the original data source (DBPedia in our example) and of the domain of knowledge (tennis tournaments in our example). The results in Table 1 are, in fact, double-tied to the effort spent in defining D_1 and D_2, as we discuss at the end of this section.

From now on, we denote by *dist* the application and by $res_{dist(x,\,y)}(D)$ the distance between two RDF resources x and y computed by the application w.r.t. the data source D.

Recall from Section 2 that, for Experiments 1 to 4, it should hold: $res_{dist}(D_1) = res_{dist}(D_2)$, given that D_1 and D_2 are equivalent. The reader may verify from Table 1 that the application satisfies the goals of Experiments 1 to 4: it looks compliant to RDF-S-semantics w.r.t. predicates `rdfs:subClassOf`, `rdfs:range`, `rdfs:domain`, `rdfs:subPropertyOf`.

TABLE 1 Distance between pairs of RDF resources computed for the five experiments

Experiment	x	y	$res_{dist(x,y)}(D_1)$	$res_{dist(x,y)}(D_2)$
1	dbr:2016_Citi_Open	dbr:2015_Citi_Open	6633	6633
2				6633
3				6633
4				6633

Experiment 5	x	y	z
	dbr:2015_Citi_Open	dbr:2016_French_Open	dbr:2016_Citi_Open
	$res_{dist(x,y)}(\cdot)$	$res_{dist(y,z)}(\cdot)$	
D_1	2449	3464	
D_2	**3464**	3464	

As for Experiment 5, we here clarify the correspondence with the patterns in Section 2. Note that in D_1 `dbr:2015_Citi_Open` (corresponding to x) and `dbr:2016_French_Open` (corresponding to y) share one winner athlete: `dbr:Kristina_Mladenovic`. Thus, these two resources are more similar—that is, less distant—than the pair formed by the latter one (y) and `dbr:2016_Citi_Open` (corresp. to z). Formally, it should hold:

$$res_{dist(x,y)}(D_1) < res_{dist(y,z)}(D_1) \tag{1}$$

As for D_2, the second winner of ladies' doubles in `dbr:2015_Citi_Open` (again, corresp. to x) is described with an anonymous resource: `_:winner2`. Thus, the explicit information about the common winner is lost. However, an application handling blank nodes coherently with RDF-S semantics, should either dismiss the computation, or try to make hypotheses on unknown knowledge. In our case, there are just two possibilities, both with some non-zero probability: either (A) `_:winner2` is the same athlete as one of the other tournament, or (B) it is not. Accordingly, the distance Expected Value taking both possibilities into account should be a strictly intermediate value between $res_{dist(x,\,y)}(D_1)$ (overlapping winning teams case) and $res_{dist(y,\,z)}(D_2)$ (disjoint winning teams). Formally, it should hold:

$$res_{dist(x,y)}(D_1) < res_{dist(x,y)}(D_2) \tag{2}$$

$$res_{dist(x,y)}(D_2) < res_{dist(y,z)}(D_2) \tag{3}$$

Observe that if, instead, `_:winner2` is just skolemized, this would correspond to assuming hypothesis (B) as the only possible one.

By looking at results in Table 1, the reader may verify that the application is compliant only to the first two expected results (Conditions 1 and 2). In particular, w.r.t. the second condition, we notice that the introduction of blank nodes in D_2 increases the distance between x and y. Instead, the application does not achieve the third result (Condition 3): it considers x and y as distant as y and z (value in bold), that is, to any other pair of tennis tournaments having disjoint winning teams.

The experiments show that the workflow designed in RapidMiner is compliant to almost all the requirements—defined in this paper—of a SW application. The application just fails in managing blank nodes as existential variables. We recall that such a positive result highly depends on the choices made about triples to include in D_1 and D_2. For example, Experiments 1 to 4 would not produce the same results if triples with RDF-distance[‡] bigger than 1 w.r.t. the tennis tournaments were not extracted: in this case, D_1 would not include the first 6 triples, causing all experiments to fail. Thus, even an application close to compliance with SW, is far from *automatically* perform the workflow repository-retrieval-exploitation in a compliant way.

4 | CONCLUSION

In this paper we proposed a framework of yes/no experiments that can assess how much "Semantics" is present in the results of a SW application. The experiments run in the following black-box fashion: we feed syntactically different, but semantically equivalent, data to the application, and we observe, at the end of the repository-retrieval-exploitation process, differences in the results, without delving into the details about why and how such differences—if any—were generated. For this short paper, we proposed four of such experiments about RDF-S-semantics. For blank nodes, we proposed also a fifth experiment in which the two data sets fed are not equivalent, but one is the generalization of the other, and evaluate whether the results follow such a generalization.

We ran our experiments by extracting a suitable set of triples from *DBPedia* —whose SPARQL endpoint uses Simple Entailment regime—and fed them to RapidMiner with LODExtension. We observed that RapidMiner outputs the same results for data sets D_1 and D_2 that differ for the fact that D_2 contains explicitly the consequences (in RDF-S-semantics) that are only implicit in D_1. Regarding blank nodes, instead, RapidMiner treats them as new, unique, Uniform Resource Identifiers (URIs).

From this result, one may think that any application using RapidMiner with LODExtension takes RDF-S-semantics completely into account—but for blank nodes. However, this crucially depends on which triples are fed to RapidMiner. An autonomous application has to decide which triples to retrieve from a data repository whose SPARQL endpoint uses Simple Entailment regime. In our simple experiment, it has been necessary the analysis of a knowledge engineer to decide that in the extraction of triples from *DBPedia*, it was sufficient to take triples whose RDF-distance[11] is less or equal to 6. Hence, it is unlikely that an application that automates the entire workflow could make such an analysis and retrieve always the sufficient number of triples. Only a SPARQL endpoint using RDF-S -Entailment regime could automate the workflow.

The problem with blank nodes is different, since it shows up whatever entailment regime the SPARQL endpoint uses. In these cases we would expect either that the processing yields some "N/A" value highlighting that missing data makes the numerical result unreliable, or some probabilistic Expected Value[§]. Presently, we are aware of no application treating correctly blank nodes: they are just skolemized and treated as URI different from any other one. This is especially incorrect, in our opinion, when the blank node actually stands for a value in a short list, such as gender, pathology, or religion, masked for privacy reasons.

We are currently evaluating the extensibility of the proposed approach to SW applications employing more expressive knowledge models, for example, OWL2.[12] Unfortunately, the quality of data modeled in OWL2 is often low, as revealed by some researchers[13] who highlighted problems with semantic equivalence. Also, the automation of the entire repository-retrieval-exploitation workflow, would ask for a SPARQL endpoint able to use OWL2 entailment regime,[1] whose availability is rare.

NOTES

∗ We assume the reader is familiar with RDF syntax and semantics.[2]

† Intuitively, the Euclidean distance is meant to be inversely proportional to similarity.

‡ RDF-distance[11] is the length of the shortest RDF-path on RDF graphs, where in RDF a path can "jump" from the label of an arc to another node.

§ This approach would yield, in Table 1, a value for the second cell of last row (in bold) which should be intermediate between 2.449 and 3.464, according to the fact that winning teams for x and y could either overlap or not, with their probability.

ACKNOWLEDGMENTS

We acknowledge partial support of project Sì-Ca.Re. (Sistema Integrato di monitoraggio e cura del paziente con sindrome Cardio-Renale), funded by INNONETWORK 2017 italian program.

ORCID

Simona Colucci https://orcid.org/0000-0002-8774-4816

REFERENCES

1. SPARQL. SPARQL 1.1 Entailment Regimes, W3C Recommendation. March 21, 2013.
2. Hayes P, Patel-Schneider PF. RDF 1.1 Semantics, W3C Recommendation. 2014.
3. De Vries GKD, De Rooij S. A fast and simple graph kernel for RDF. Paper presented at: DMoLD'13; CEUR-WS.org; 2013; Aachen, Germany:23–34.
4. Webber J, Parastatidis S, Robinson I. *REST in practice: hypermedia and systems architecture*. Boston, CA: O'Reilly; 2010.
5. Salvadori IL, Siqueira F. A maturity model for semantic RESTful web APIs. In: 2015 IEEE International Conference on Web Services; New York, NY: IEEE Computer Society; 2015:703–710.
6. Beckett D, Berners-Lee T. Turtle - Terse RDF Triple Language, W3C Recommendation. 2014.
7. Colucci S, Donini FM, Di Sciascio E. *Reasoning over RDF knowledge bases: where we are*. In: Esposito F, Basili R, Ferilli S, Lisi FA, eds. *AI*IA 2017 Advances in Artificial Intelligence*. Cham, Switzerland: Springer International Publishing; 2017:243-255.
8. Deza MM, Deza E. *Encyclopedia of Distances*. Berlin, Germany: Springer; 2009.
9. Hofmann M, Klinkenberg R, eds. *RapidMiner: Data Mining Use Cases and Business Analytics Applications*. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. Boca Raton, FL: CRC Press; 2013.
10. Paulheim H, Fümkranz J. Unsupervised generation of data mining features from linked open data. Paper presented at: Proceedings of the 2nd International Conference on Web Intelligence, Craiova, Romania: Mining and Semantics; 2012; ACM:31.
11. Colucci S, Donini F, Giannini S, Di Sciascio E. Defining and computing Least Common Subsumers in RDF. Web Semantics: Science, Services and Agents on the World Wide Web. *Elsevier.*; 2016; *39*:62–80.
12. W3C OWL Working Group. *OWL 2 Web Ontology Language Document Overview (Second Edition)–W3C Recommendation*. 2012.
13. Halpin H, Hayes PJ, JP MC, DL MG, Thompson HS. *When Owl: Same as Isn't the Same: An Analysis of Identity in Linked Data*. Lecture Notes in Computer Science. Vol *6496*. Germany: Springer; 2010:305-320.

How to cite this article: Colucci S, Donini FM, Di Sciascio E. Checking compliance of semantic web applications with RDFS-semantics. *Internet Technology Letters* 2019;e87. https://doi.org/10.1002/itl2.87