Reasoning over RDF Knowledge Bases: where we are

Simona Colucci[1], Francesco M. Donini[2], and Eugenio Di Sciascio[1]

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

Abstract. This paper aims at investigating the state of realization of the Semantic Web initiative, through the analysis of some applications taking background knowledge from RDF datasets. In particular, it shows the design and the implementation of an extended experiment, which demonstrates that input datasets are often used only as data structures, without taking into account the logical formalization of properties involved in such RDF models.

1 Motivation

After more than 15 years since the launch of the Semantic Web (SW)[1] initiative, the way of conceiving knowledge modeling has radically changed, by going through heterogeneous phases. In the envisioned SW architecture, graphically synthesized in the well-known SW layer-cake, the data interchange layer is detached from the ontology level and different modeling languages enable them. In particular, Resource Description Framework (RDF) [7] is the language implementing the data interchange layer; as for the ontology level, languages of different expressiveness have been defined to cover different modeling needs. Such languages may provide a relatively low expressiveness, like the schema language associated to RDF, RDF Schema (RDFS) [3], or may present a modeling potential comparable to really expressive languages for knowledge representation, like the Web Ontology Language (OWL), today at its second version, OWL-2 [13].

Currently, the so-called Web of Data [12] offers a huge amount of data, modeled in RDF according to vocabularies defined in different ontology languages. Independently on the expressiveness of such languages, RDF data come with their own—although simple—semantics.

Nevertheless, most SW applications seem to completely disregard the implicit informative content embedded in RDF data sources, exploiting them just as rough *data structures*—in particular, directed graphs.

In order to support this claim, we here design an extended experiment aimed at testing the compliance to the SW initiative of applications using RDF data sources as background knowledge. The experiment can be used to discriminate SW applications on the basis of their ability to correctly interpret some predicates defined in RDFS-semantics [7] and to manage *blank nodes* as special RDF resources.

We devised the designed experiments in one representative SW application managing RDF datasets as data source. Results show that: i) in most cases, the application behaves differently when working with datasets which are logically equivalent—according to RDFS semantics—although expressed in syntactically different ways; ii) blank nodes are not given special handling. This experiment demonstrates that the application does not consider the semantics of properties involved in the input datasets and does not follow the intuition about blank nodes.

Such a result raises important questions about the state of realization of the SW initiative: was not RDF conceived for data interchange? Was not the *meaning* of data part of the envisioned interoperability? Is really a flat transformation from RDF models to feature sets what the SW creators expected from released applications?

In this paper we start from the experimental results to discuss issues still open in the development of RDF applications fully realizing the SW initiative.

The rest of the paper is organized as follows: in the next section, we explain the rationale of the experiments we devised. Then, we show the execution of the experiments in Section 3. In Section 4 results and main lesson learned from the experiments are synthesized, before concluding the paper with Section 5.

2 Rationale of the experiments

We explain here our strategy for devising experiments that can discriminate whether some application based on SW data exploits just the syntax of data, or instead its semantics.

2.1 RDFS semantics

Experiment 1.
The first experiment is based on the RDFS predicate `rdfs:subClassOf`. According to RDFS semantics, such a predicate should be interpreted as transitive, meeting the intuition conveyed by its bare name. Consider the following RDF data patterns:

x `rdfs:subClassOf` u . u `rdfs:subClassOf` z . x `rdfs:subClassOf` z .	x `rdfs:subClassOf` u . u `rdfs:subClassOf` z .

where x, u, z are generic IRIs.

Two data sources D_1, D_2, where D_1 contains the left-hand pattern, and D_2 the right-hand pattern, must be completely equivalent with respect to RDFS semantics (see Rule *rdfs1* in the definition of the semantics of RDFS [7]). So, for instance, if a SW application computes a similarity measure $s(x, y)$ between x and another IRI y, the similarity $s(x, y)$ computed using triples in D_1 should be the same as the one computed using triples in D_2. If this experiment yields

a "no" answer, the SW application is using just the *syntactic* form of triples—like a database—not their semantics. While this might be fine for the results of the application at hand, one should be clear about the *sensitivity* of such an application to apparently redundant triples. The intuition about how "semantic" is a SW application here may be misleading, since syntactic differences that may be judged irrelevant may be in fact not so.

Experiment 2.
The second experiment is based on the RDFS predicate `rdfs:domain`. Consider the following RDF data patterns:

x r u . r `rdfs:domain` z . x `rdf:type` z .	x r u . r `rdfs:domain` z .

where x, r, u, z stand for some IRIs. Consider, again, two data sources D_1, D_2, defined to include the left-hand and the right-hand pattern above, respectively. According to RDFS semantics, the third triple in D_1 is trivially redundant w.r.t. to the content in D_2 (see Rule *rdfs2* in the RDFS semantics [7]). Also in this case, a SW application computing, for instance, a similarity measure $s(x, y)$ between x and y should return the same value when either D_1 or D_2 are used as data sources. Any different behavior of the application would reveal that triples are only syntactically parsed.

Experiment 3.
The third experiment is based on the RDFS predicate `rdfs:range`. Consider the following RDF data patterns:

x r u . r `rdfs:range` z . u `rdf:type` z .	x r u . r `rdfs:range` z .

where x, r, u, z stand for generic IRIs. The same arguments as for Experiment 2 apply if two data sources D_1, D_2 are defined to include the left-hand and the right-hand pattern above, respectively. Again, D_1 and D_2 are logically equivalent according to RDFS semantics (see Rule *rdfs3* in the document defining RDFS semantics [7]), and any application claiming to be SW-oriented should return identical results when either D_1 or D_2 are used as data sources. Also in this experiment, the reader may think to an application computing the similarity between two IRIs, for the sake of example.

Observe that the same experiments could be conducted if instead of a similarity measure, the SW application performs a clusterization of IRIs. In this case, the test must ascertain whether the IRI x is put in the same cluster or in a different cluster, depending on which of the two patterns above x is involved in.

2.2 The status of blank nodes

The second type of experiment analyzes if the SW application follows the intuition about *blank nodes*. In the original semantics of RDF, blank nodes are existential variables, whose scope is the entire RDF file they occur in. As such, they may stand for any IRI or literal, even one not occurring already in the file. We stress that blank nodes appear in the syntax and semantics of simple RDF, and that RDFS in this case inherits them from the simpler language. Hence this experiment can be set up even if one does not adopt RDFS semantics.

Experiment 4. Consider the three RDF data patterns below:

x	r	`ex:a .`	x	r	`_:b1 .`	x	r	`ex:a .`
y	r	`ex:c .`	y	r	`_:b2 .`	y	r	`ex:a .`

where, following Hayes&Patel-Schneider [7], blank nodes are prefixed by an empty namespace "_" as in `_:b`, and x, y, r denote generic IRIs. In the first pattern, IRIs x and y are linked through r to different, known, IRIs `ex:a`, `ex:c`. In the second pattern, they are linked to two blank nodes, that might coincide in one interpretation, while they might not in another. In the third pattern, both x and y are linked to the same IRI `ex:a`. Now let D_1, D_2, D_3 be three data sources which are the same but for the fact that, for some IRI x, D_1 contains the first pattern, while D_2 contains the second one, and D_3 the third.

Suppose now that a SW application estimates the similarity of x and y. Clearly, *ceteris paribus*, in D_1 IRIs x and y are less similar than in D_3, in which they share the same predicate-object pair. The situation for D_2 is an intermediate one: in fact, *both* the first and the third pattern can be instantiations of this one—among may others. Denoting by $s_1(x, y)$ the similarity the SW application computes when D_1 is given as data source, and by $s_2(x, y)$ and $s_3(x, y)$ those computed for D_2 and D_3, respectively, $s_2(x, y)$ should be an intermediate value between $s_1(x, y)$ and $s_3(x, y)$—that is, one should obtain $s_1(x, y) \leq s_2(x, y) \leq s_3(x, y)$.

We observed instead that many SW applications treat each blank node as a new IRI, different from every other one. This is what is logically called *skolemization*. While skolemization yields a data source which—considered as a formula—is roughly equivalent to the original one for what regards *entailment* [7, Sect.6], the picture is much different here. In fact, SW applications compute values and do other operations on the data—such as clusterization—as if RDF triples specified a *model*, not a formula (which represents a *set* of models). Performing a skolemization and then use the result for computations amounts to treating the second pattern above as if it were the first—or, treating D_1 and D_2 to be in the same situation. Again, the so-called "RDF graph" is treated in this case as a *real* graph—a data structure—bypassing its logical meaning.

When a SW application treats blank nodes by skolemizing them, the result of the experiment shows that always $s_1(x, y) = s_2(x, y) < s_3(x, y)$. Again, while this may be acceptable for the SW application at hand, one should clarify that blank nodes appearing in the data source are always treated as new, distinguished, IRIs.

This treatment may be counterintuitive especially when the blank node stands for one in a short, limited list of possible values—say, when r =`foaf:gender`, its object is intuitively either `foaf:male` or `foaf:female`—while Skolem constants add more values to the initial list, with no real meaning, but for the fact that they are values different from any other one.

In data analysis research community, the problem of correctly interpreting unbounded information is acknowledged and a florid research field is specifically devoted to the prediction of (so-called) missing values ([11], [2]), which are the analogous of blank nodes in RDF data sources. In our opinion, solutions predicting missing values (the value of blank nodes, in our case) by estimating the expected value of the variable they quantify [10] are not adequate to SW settings. In fact, this could prematurely make discrete features that should be kept general for further inference purpose.

3 Deployment of the experiments

We here show how the experiments designed in Section 2 may serve as a test checking if an application working on RDF data sources may be considered compliant with the SW requirements.

As use case, we choose a very popular SW application: the Linked Open Data extension (LODExtension) [9] of the Machine Learning tool RapidMiner [8]. The RapidMiner LODExtension enables several learning tasks on example sets automatically extracted by mining data sources in RDF.

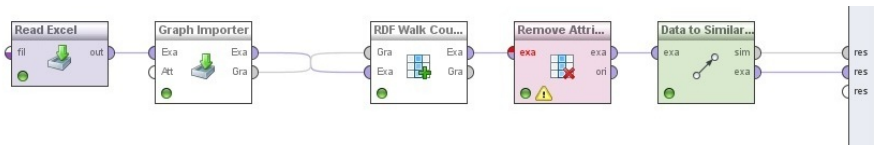

Fig. 1. Rapidminer workflow of performed experiments

All performed experiments follow the RapidMiner workflow shown in Figure 1, aimed at computing the similarity of RDF resources through the operations described below:

1. **Read Excel.** This operator loads data from Microsoft Excel spreadsheets; in this workflow, the spreadsheet includes a column with the IRIs of resources to compare.
2. **Graph Importer.** This operator generates an RDF graph for all input instances, either from SPARQL endpoint or local file, with user-specified graph depth. In our experiment, an RDF dataset is queried to import specific RDF graphs, whose root and depth are user-specified.
3. **RDF Walk Count Kernel.** This kernel method counts the different walks in the subgraphs (up to the provided graph depth) around the instance

nodes. The method implements the algorithm by de Vries *et al.* [5]. In this worflow, it is applied on the RDF graph imported in Step 2 for generating features describing the resources of interest according to the knowledge in the dataset; in particular, RDF Walk Count Kernel counts the different walks in the subgraphs (up to the provided graph depth) around the root. The operator returns a so-called "ExampleSet": a set of kernel-generated features describing the resources of interest. The generation process may be set to make use of inference on explicit knowledge.

4. **Remove attributes.** This is an auxiliary operator which cuts columns from example sets. In this workflow, it is used to remove the IRI of the root from the set generated at Step 4, because it is not object of comparison.
5. **Data to similarity.** This operator measures the similarity of each example of a given ExampleSet with every other example of the same ExampleSet. A similarity measure may be chosen by the user among available ones.

We now detail the execution of four experiments following the rationale in Section 2. For all of them, the "inference" option is set "on" in Step 4 and the similarity measure chosen at Step 5 is the one denoted as "Euclidean Distance" in RapidMiner[3] (based on the well-know measure of the same name [6]).

The first three experiments aim at testing the sensitivity of the workflow in Figure 1 to knowledge-irrelevant changes in the RDF input dataset. In particular, we show the similarity values returned by the selected SW application when two logically-equivalent RDF models are given as input. For each experiment, we describe the two input datasets—from now on D_1 and D_2—below:

Experiment 1. In this experiment, D_1 and D_2 describe resources Mandarin Orange (IRI `http://dbpedia.org/resource/Mandarin_orange`) and Tangerine (IRI `http://dbpedia.org/resource/Tangerine`). In particular, D_1 and D_2 syntactically differ in the assertions about Mandarin Orange (Resource 2), although being logically equivalent according to RDFS-semantics (see predicate `rdfs:subClassOf`). In fact, D_1 is depicted in Figure 2 and matches the left-hand pattern provided in the the description of Experiment 1 in Section 2.1, while D_2, depicted in Figure 3, matches the right-hand one.

Experiment 2. This experiment uses two datasets about Mandarin Orange (IRI `http://dbpedia.org/resource/Mandarin_orange`) and Coffea (IRI `http://dbpedia.org/resource/Coffea`). Coherently with the rationale provided in Section 2.1, D_1 and D_2 are syntactically different but logically equivalent according to RDF-S semantics (see predicate `rdfs:domain`). In Figure 4 (respectively Figure 5), it is shown D_1 (respectively D_2), that matches the left-hand (respectively, right-hand) pattern given in the description of Experiment 2 in Section 2.1.

Experiment 3. This experiment uses two other datasets about Mandarin Orange (IRI `http://dbpedia.org/resource/Mandarin_orange`) and Coffea (IRI

[3] Intuitively, distance is meant to be inversely proportional to similarity

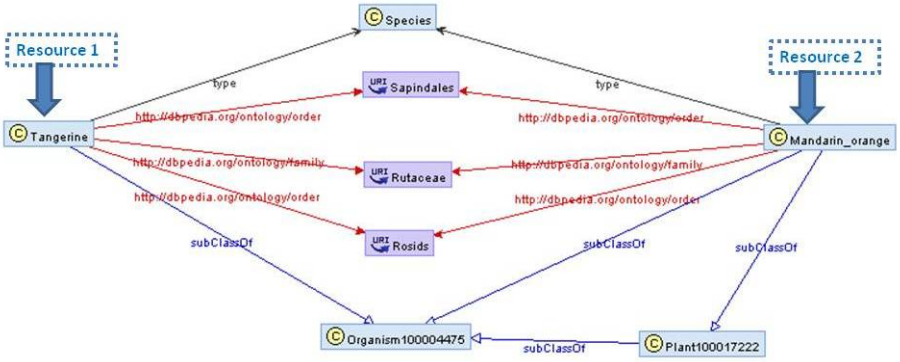

Fig. 2. D_1: an RDF dataset about Mandarin Orange and Tangerine. The graph matches the left-hand data pattern given in Experiment 1 in Section 2.1, by binding x to Resource 2

`http://dbpedia.org/resource/Coffea`) involving predicate `rdfs:range`. According to the rationale of Experiment 3 explained in Section 2.1, the two datasets are logically equivalent, even though syntactically D_1 (Figure 6) matches the left-hand graph pattern in the experiment description, while D_2 (Figure 7) matches the right-hand one.
The three experiments share the goal to test that:

$$dist_1(x, y) = dist_2(x, y) \tag{1}$$

where for $i = 1, 2$, $dist_i(x, y)$ denotes the Euclidean Distance between x and y, when they are defined in D_i.

Experiment 4 This experiment aims at testing if the workflow for computing similarity gives anonymous resources special handling or considers them exactly as any other resource. The experiment follows the rationale described in Section 2.2 and uses the dataset in Figure 8.

In particular, we show below how to retrieve in Figure 8 three graph patterns given in the description of Experiment 3 in Section 2.2:

- by binding x to Resource 1 and y to Resource 3, the graph in Figure 8 matches the first graph pattern (on the left of the table);
- by binding x to Resource 2 and y to Resource 5, the graph in Figure 8 matches the second graph pattern (on the center of the table);
- by binding x to Resource 1 and y to Resource 4, the graph in Figure 8 matches the third graph pattern (on the right of the table);

According to the rationale provided in Section 2.2, the experiment goal is verifying that:

$$dist(Resource1, Resource3) \geq dist(Resource1, Resource3) \geq dist(Resource1, Resource4) \tag{2}$$

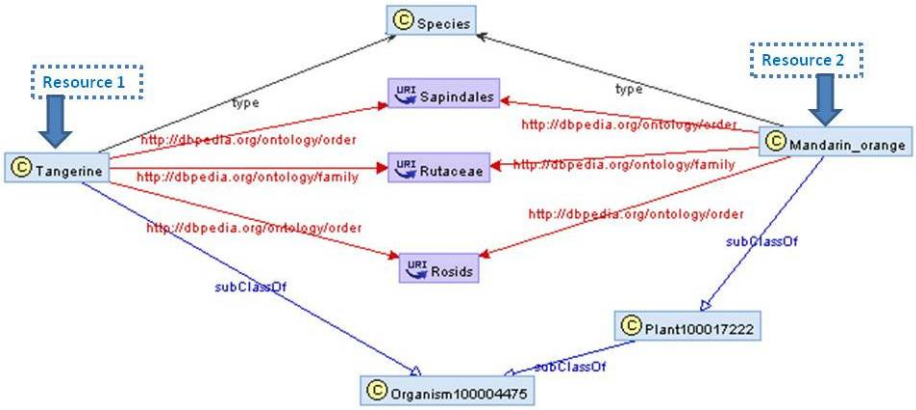

Fig. 3. D_2: a dataset logically equivalent to the one in Figure 2. The graph matches the right-hand data pattern given in Experiment 1 in Section 2.1, by binding x to Resource 2 .

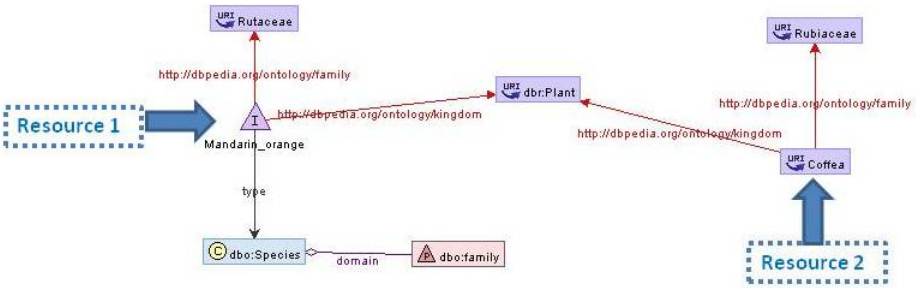

Fig. 4. D_1: an RDF dataset about Mandarin Orange and Coffea. The graph matches the left-hand data pattern given in Experiment 2 in Section 2.1, by binding x to Resource 1

where $dist(x, y)$ denotes the Euclidean Distance between x and y (recall Footnote 3).

4 Results and Discussion

We here report the results of experiments performance w.r.t. the settings detailed in Section 3. For each experiment, Table 1 shows the values of Euclidean Distance ($dist_i(x, y)$) between each pair of resources x and y (defined in D_i).

Experiment 1–3 We recall from Section 3 that Experiment 1–3 are focused on checking if the selected SW application satisfies the goal in Equation 1. This reverts to check if, for each experiment, the values of $dist_i(x, y)$ coincide in the rows related to data sources D_1 and D_2. The reader may verify that this happens only for Experiment 1, while the test fails for Experiment 2 and 3.

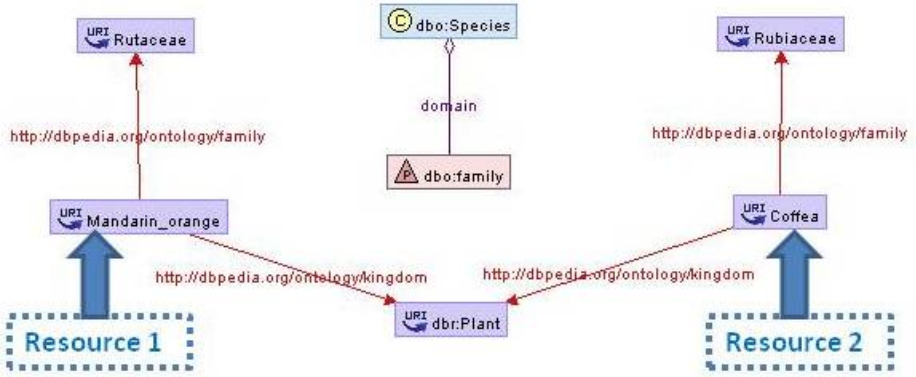

Fig. 5. D_2: a dataset logically equivalent to the one in Figure 4. The graph matches the right-hand data pattern given in Experiment 2 in Section 2.1, by binding x to Resource 1

In other words, the analyzed SW application takes into account the transitivity of predicate `rdfs:subClassOf` by applying inference rules on knowledge extracted from input data sources. On the contrary, it seems not to apply the entailment patterns about `rdfs:domain` and `rdfs:range` in the process of generating features from the input data sources. In fact, in both Experiment 2 and 3, the application does not notice that $D_1 \equiv D_2$ according to RDF-S-semantics.

The reason for such a dichotomy lays in the kernel method applied in Step 3 of the process workflow. The Walk Count Kernel operator generates features by visiting paths in the input RDF-graph rooted in the resource to describe. But this operator does not take into account one important peculiarity of RDF-grahs: the fact that the set of labels of nodes and arcs may overlap in RDF, causing some paths to be connected through the predicate and not through the object of an RDF statement (as an example, the right-hand patterns given for Experiment 2 and 3 in Section 2.1 are connected through r and not through u). A definition of such paths in terms of RDF-paths may be found in previous work [4].

We believe that a SW application should be able to manage such paths when mining knowledge from data sources in order to be compliant with RFDS-semantics. This would avoid the production of completely misleading results, like the ones of Experiment 2 and 3 given in Table 1. In fact, by comparing Row 3 and Row 4 (alternatively, Row 5 and 6), the reader may notice that Resource 1 and Resource 2 are considered more distant from each other if (in dataset D_1) Resource 1 is further described by the statement:
`http://dbpedia.org/resource/Mandarin_orange`
`http://www.w3.org/1999/02/22-rdf-syntax-ns#type`
`http://dbpedia.org/ontology/Species` .
which is completely redundant with respect to the content of D_2 according to RDFS-semantics.

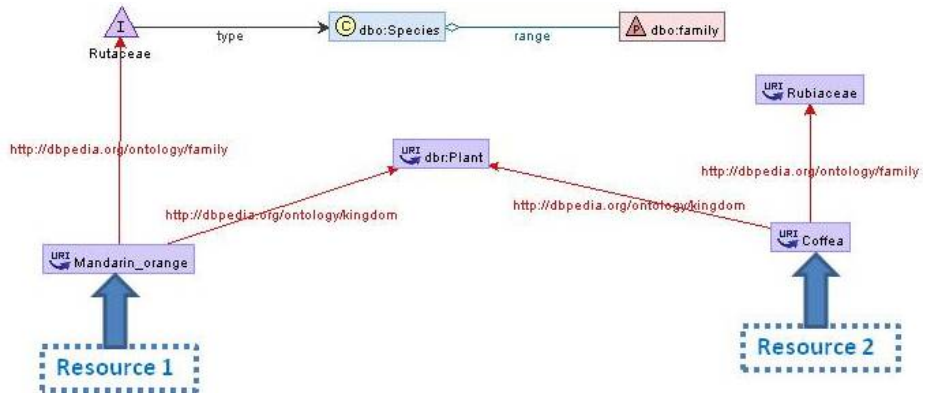

Fig. 6. D_1: an RDF dataset about Mandarin Orange and Coffea. The graph matches the left-hand data pattern given in Experiment 3 in Section 2.1, by binding x to Resource 1

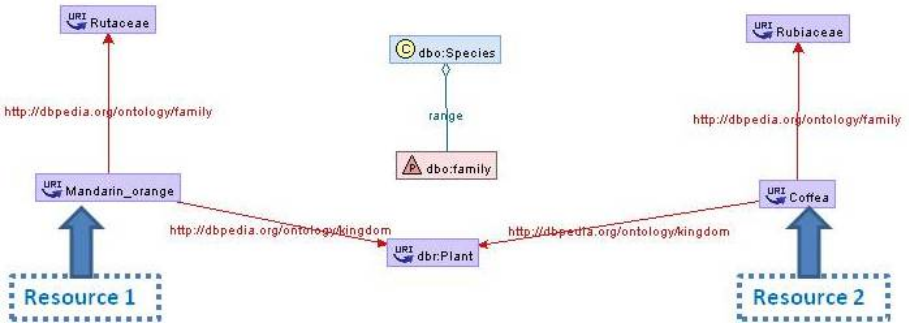

Fig. 7. D_2: A dataset logically equivalent to the one in Figure 6. The graph matches the right-hand data pattern given in Experiment 3 in Section 2.1, by binding x to Resource 1

Experiment 4 The results of Experiment 4 are formalized in the last three rows of Table 1. We recall that the experiment goal, formalized in Equation 2, is checking if there is any special handling for blank nodes w.r.t. IRIs in the computation of similarity. This reverts to check: i) if the application considers Resource 1 and 3 (which are linked to different *IRIs* through the predicate `http://dbpedia.org/ontology/address`) more distant from each other than Resource 2 and 5 (which, instead, are linked to different *blank nodes* through the predicate `http://dbpedia.org/ontology/address`); ii) if both the distances above are bigger than the distance between Resource 1 and 4 (which are linked to *the same IRI* through the predicate `http://dbpedia.org/ontology/address`).

The reader may notice that the application satisfies only the second part (item ii) of the goal: it computes a distance equal to 0 between Resource 1 and 4, as one may expect. On the contrary, it fails to address the first part of the

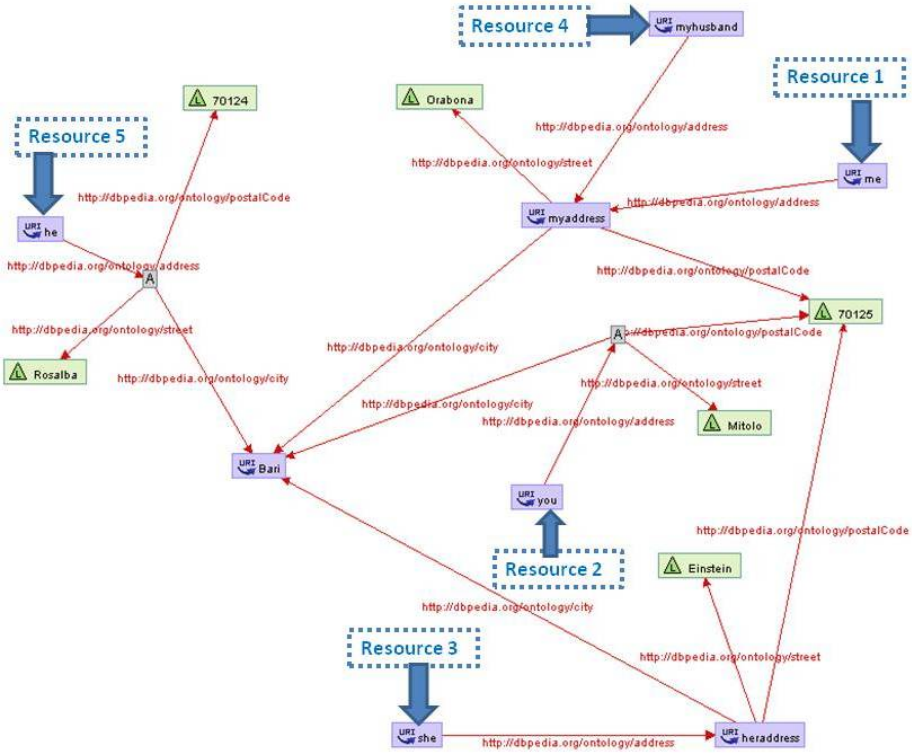

Fig. 8. Example of RDF dataset involving blank nodes (denoted by A).

goal: Resource 1 and 3 are considered as distant from each other as Resource 2 and 5, showing that blank nodes are treated the same as any other IRI.

We believe that a SW application (even only compliant with simple-entailemnt regime) should give the proper treatment to blank nodes, in order to avoid undesirable loss of information which are evident even in our simple example data source. By looking at Figure 8, the reader may notice that all resources are connected to an address, which is in turn connected to the resource `http://dbpedia.org/resource/Bari` through the property `http://dbpedia.org/ontology/city`.
The Walk Count Kernel operator generates one feature for each path rooted in the five resources. In particular, by mining the model of Resource 2 (analogously, 5) it generates (among the others) a feature stating that the resource is connected to *some address* in the city of Bari. In our opinion, such a feature should be matched by all the other resources, but the application considers it exclusive of Resource 2 (analogously, 5). This happens because the application gives to *some address* a symbolic name (a Skolem constant), which by definition is different by any other IRI (in thi case, address) it manages.

	Input Data Source:D_i	Binding		$dist_i(x, y)$
		x	**y**	
Experiment 1	D_1 (Figure 2)	Resource 2	Resource 1	4
	D_2 (Figure 3)	Resource 2	Resource 1	4
Experiment 2	D_1 (Figure 4)	Resource 1	Resource 2	4,472
	D_2 (Figure 5)	Resource 1	Resource 2	2,449
Experiment 3	D_1 (Figure 6)	Resource 1	Resource 2	3,162
	D_2 (Figure 7)	Resource 1	Resource 2	2,449
Experiment 4	Figure 8	Resource 1	Resource 3	4,243
		Resource 2	Resource 5	4,243
		Resource 1	Resource 4	0

Table 1. Euclidean Distance ($dist_i(x, y)$) between each pair of resources x and y (defined in D_i), to be compared in designed experiments.

5 Conclusion

We designed a set of experiments which can be used to discriminate applications involving RDF data sources on the basis of their compliance to the SW initiative. In particular, three experiments focus on the compliance to RDFS-semantics, and check if an application correctly interprets the predicates `rdfs:subClassOf`, `rdfs:domain` and `rdfs:range`. The fourth experiment is focused on the management of blank nodes and aims at discriminating applications on the basis of the handling they give to such–special–resources.

In order to show how the set of experiments may be performed, we chose as test application a RapidMiner process computing the Euclidean distance between two IRIs. The workflow embeds operators from the LODExtension of RapidMiner, which is meant to support classical data mining tasks with information extracted from RDF data sources (also by applying inference techniques).

Results show that the chosen application fails three of the tests induced by the four experiments: it is not able to manage predicates `rdfs:domain` and `rdfs:range` and blank nodes. Knowledge deriving by transitivity of predicate `rdfs:subClassOf` is instead correctly inferred. In other words, the application may not be considered fully compliant to the SW initiative.

Our future work will be devoted to run the designed experiments in a broader set of applications traditionally considered as SW-oriented. Results of such an extended experiment would reveal the current state of realization of the SW initiative.

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American 248(4) (2001), (34-43)
2. Bischof, S., Martin, C., Polleres, A., Schneider, P.: Collecting, Integrating, Enriching and Republishing Open City Data As Linked Data. In: Proceedings, Part II, of the 14th International Semantic Web Conference on The Semantic Web - ISWC 2015 - Volume 9367. pp. 57–75. Springer-Verlag New York, Inc., New York, NY, USA (2015), http://dx.doi.org/10.1007/978-3-319-25010-6_4
3. Brickley, D., Guha, R.: RDF Schema 1.1- W3C Recommendation (2014), https://www.w3.org/TR/rdf-schema/
4. 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 39, 62 – 80 (2016), http://dx.doi.org/10.1016/j.websem.2016.02.001
5. De Vries, G.K.D., De Rooij, S.: A Fast and Simple Graph Kernel for RDF. In: Proceedings of the 2013 International Conference on Data Mining on Linked Data - Volume 1082. pp. 23–34. DMoLD'13, CEUR-WS.org, Aachen, Germany, Germany (2013), http://dl.acm.org/citation.cfm?id=3053776.3053781
6. Deza, M.M., Deza, E.: Encyclopedia of distances (2009)
7. Hayes, P., Patel-Schneider, P.F.: RDF semantics, W3C recommendation (2014), http://www.w3.org/TR/2014/REC-rdf11-mt-20140225/
8. Hofmann, M., R.Klinkenberg (eds.): RapidMiner: Data mining use cases and business analytics applications. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series, CRC Press (2013)
9. Paulheim, H., Fümkranz, J.: Unsupervised generation of data mining features from Linked Open Data. In: Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics. p. 31. ACM (2012)
10. Perez-Rey, D., Anguita, A., Crespo, J.: OntoDataClean: Ontology-Based Integration and Preprocessing of Distributed Data, pp. 262–272. Springer Berlin Heidelberg, Berlin, Heidelberg (2006), http://dx.doi.org/10.1007/11946465_24
11. Qi, Z., Wang, H., Meng, F., Li, J., Gao, H.: Capture Missing Values with Inference on Knowledge Base, pp. 185–194. Springer International Publishing, Cham (2017), http://dx.doi.org/10.1007/978-3-319-55705-2_14
12. Shadbolt, N., Hall, W., Berners-Lee, T.: The Semantic Web Revisited. Intelligent Systems, IEEE 21(3), 96–101 (2006)
13. W3C OWL Working Group: OWL 2 Web Ontology Language Document Overview (Second Edition)–W3C Recommendation (2012), https://www.w3.org/TR/owl2-overview/