On the detection of inconsistencies in RDF data sets and their correction at ontological level

Youen Péron, Frédéric Raimbault, Gildas Ménier, Pierre-François Marteau

On the detection of inconsistencies in RDF data sets and their correction at ontological level

Youen Péron, Frédéric Raimbault, Gildas Ménier, and Pierre-François Marteau

Valoria, Université de Bretagne Sud, Université Européenne de Bretagne, Vannes, France
E-mail: {firstname.lastname}@univ-ubs.fr

Abstract. The rapid development of Linked Data leads to a proliferation of errors in published data, primarily related to inconsistencies between data instances and their related ontologies. This problem alters the reliability of Semantic Web applications when they involve the analysis or the exploitation of heterogeneous RDF data sets. We focus in this article on a way to correct inconsistencies caused by the domain and the range of a property. We present an algorithm to identify the source of these inconsistencies in the ontology, and to provides guidelines to correct or improve the ontology. The localization of the inconsistencies is based on a quantitative comparison between the classes of domains and ranges defined in the ontology and the ones built by the exhaustive analysis of instances used as subject or object in the properties. We show the usefulness of this method on a case study involving DBpedia: we use our approach to diagnose and correct common inconsistencies. Another experiment conducted on a large data set generated by SP2BENCH validates the scalability of the proposed algorithm.

1 Introduction

Launched in 2006 by Tim Berners-Lee, the *Linked Data*[1] movement and its recommendations is playing a leading role for publishing, sharing, and interconnecting data on the Semantic Web [4]. Many companies (eg. the BBC and The New York Times), organizations (eg. Wikipedia, Geonames, FreeBase, UN) and governments (eg. U.S. and U.K.) have adopted the principles of *Linked Data* to publish their data using RDF[2] standard and their ontologies[3] using RDFS[4] and OWL[5]: the size and variety of available data sets keep growing. April 2011, the *Linked Data* giant global graph data has been estimated over 200 large data sets (more than 1000 triplets each), and totaling 29 billion RDF triples and 400 million links[6]. Unfortunately, this uncontrolled growth leads to a proliferation of errors in published data, primarily related to inconsistencies between data instances and their related ontologies. This problem alters the reliability of Semantic Web applications when they involve the analysis or the exploitation of heterogeneous RDF data sets. Therefore, it is important to improve the quality of the published data to promote the development of the Semantic Web. Our method quantitatively evaluates domain and property misuses and helps fix these problems.

Most of the research –dealing with data quality in Semantic Web– focus either on ontology (class and properties definition) or raw data (class and literal instances).

Ontology validation has been extensively studied in [5, 3, 11, 9]. These works verify the consistency and completeness of an ontology: The tools are targeted towards ontology designers, irrespectively of the use related to the data publication.

Surprisingly, validation of raw data has been much less studied, even if it represents the main part of the published data. Some tools have been designed for syntactic validation, such as VRP [15] a tool from ICS-FORTHRFDFSUITE[7]. This kind of tools checks the data compliance with the RDF

[1] http://www.w3.org/DesignIssues/LinkedData.html
[2] http://www.w3.org/TR/rdf-concepts/
[3] http://www.w3.org/TR/webont-req/#onto-def
[4] http://www.w3.org/TR/rdf-schema/
[5] http://www.w3.org/TR/owl-semantics/rdf-concepts/
[6] http://www4.wiwiss.fu-berlin.de/lodcloud/
[7] http://139.91.183.30:9090/rdf

syntax and semantic constraints (also called consistency logic) in ontologies - under the assumption that the used ontologies are errors free. The Pedantic Web's project[8] proposes diagnosis of ontology errors, and especially recommendations to avoid them. In [7] the initiators of this project perform an analysis of errors found in a sample set obtained by crawling: they identify four symptoms of recurring errors and propose recommendations to manage them. The authors also introduce an on-line tool, RDF:ALERTS[9] designed to detect these errors. Our contribution takes part to the Pedantic Web main works: we propose to enhance the semantic analysis by adding range and domain verification for properties on non literal data (unlike [7] where the analysis is strictly restricted to range checking on literal data). Our analysis is performed on class instances, being subject or object of property. We also propose a way to measure the importance of inconsistencies detected so that the priority of the correction can be evaluated: if a correction has to be done, we propose a diagnosis and an adapted fix. In [14], the authors also address the problem of data compliance to ontologies. They propose a generic evaluation method of data, based on systematic search of some inconsistency patterns: this search is performed by SPARQL queries applied to the deductive closure of the graph of all inference rules. Their formulation is complex because it requires negations (which is non trivial for SPARQL). Since we want our method to be scalable such to be tested on very large set of RDF data, we propose a one-rule inference (hierarchy) process to speed up the detection of possible errors. We performed our test using the distributed system Hadoop and the request language PIG.

The rest of the paper is organized as follows: In section 2, we recall some basic notions of the semantic web and we define the notations used throughout this article. Section 3 contains the major part of our contribution: we first formalize the concepts of domain and range inconsistencies; then we describe an algorithm that detects and evaluates the number of inconsistencies. Section 4 gives the hints and solutions to enhance the ontology. A qualitative evaluation on the DBpedia ontology and a scalability testing are presented in the section 5. In the last section, we resume our work and give some possible generalizations.

2 Background and notations

We recall main concepts of the RDF data model, RDFS and OWL vocabularies and we give the notations we use in this paper.

2.1 RDF model

According to the RDF data model, all information is expressed as a triplet $(subject, predicate, object)$. The subject and predicate of a triple are resources identified by URIs[10]. In this article, we use the prefixed URI form to simplify the examples. A triplet object can be either a resource or a literal. Let $\mathcal{U}$ be the set of resources and $\mathcal{L}$ the set or literals. A triplet t is defined as follows :

$$t = (s, p, o) \in \mathcal{U} \times \mathcal{U} \times (\mathcal{U} \cup \mathcal{L})$$

Since a same resource may play either the role of a subject or the role of an object into different triplets, a set of triplets can be represented as a graph. We address the ABox/TBox separation problem (see [2]) by categorizing the triplets into the terminology data (TBox) and the assertion data (ABox).

2.2 Meta Data Vocabulary

Terminology information is described using specific RDF, RDFS or OWL vocabulary as defined by the W3C. We introduce here in after the terminology we use in this paper :

[8] site : `http://pedantic-web.org`
[9] `http://swse.deri.org/RDFAlerts/`
[10] URI : Uniform Resource Identifier

```
PREFIX dbpedia: <http://dbpedia.org/ontology/>
SELECT DISTINCT ?resource WHERE { ?resource a dbpedia:Person}
```

Fig. 1. SPARQL query that returns resources belonging the class `dbpedia:Person`

`rdf:type` is a resource that provides an elementary system of belonging to a class. A triplet $(r, rdf : type, c)$ translates as " r belongs to the class c " and is denoted by $r \in c$. Let $resources(c)$ be the set of resources belonging to the class c.
The SPARQL query in the figure 1 returns the set of resources belonging the class `dbpedia:Person`.
The number of resources in a class c is $|c| = |resources(c)|$. A same resource may belong to several classes. Let $\mathcal{C}$ be the set of classes. All resources belong to the class `owl:Thing`.

`rdf:Property` is a class whose instances are used as predicate in a triplet. These resources are called properties. Let the set of properties be $\mathcal{P} = resources($`rdf:Property`$)$. A property's domain and its range are constrained respectively in the TBox with the predicate `rdfs:range` and `rdfs:domain`. Domain and range classes of the property p are denoted respectively by $D(p)$ and $R(p)$. The default domain and the default range of a property p are `owl:Thing` .

`owl:ObjectProperty` is the class of properties which range cannot be a literal, but only a class instance (also called property-object in the text below).

`rdfs:subClassOf` is a property describing a subsumption relationship between a class c and a parent class f ($c \prec f$). This is a transitive relationship :

$$\forall c_1, c_2, c_3 \in \mathcal{C}\ (c_1 \prec c_2) \wedge (c_2 \prec c_3) \implies (c_1 \prec c_3)$$

Each class can subsume itself :

$$\forall c \in \mathcal{C},\ c \prec c$$

Let $\downarrow (c)$ be the set of classes that subsume a class c :

$$\downarrow (c) = \{j \in \mathcal{C}, j \prec c\}$$

The generated hierarchy is defined as $\mathcal{H}$. We assume that, for graphs considered in this paper, the closure of the subsumption relationship has already been performed (as preprocessing) on the graph so that the following formula is true :

$$(1)\ (\forall u \in \mathcal{U}), (\forall c, \forall f \in \mathcal{C}),\ (u \in c) \wedge (c \prec f) \implies (u \in f)$$

where $\mathcal{U}$ is the set of resources.
We used the reasoning method described in articles [16] and [8], which enables the implementation of inference rules on large (distributed) RDF data sets.

3 Inconsistencies detection

In this section, we introduce the notions of the actual domain and the actual range of an object-property.We compare them with the definition domain and the definition range of an object property. We propose a method which exploit this difference to reveal domain and range inconsistencies. We test this method on to the DBpedia data set.

3.1 Actual domain and actual range

Let $domain_a(p)$ be the set of resource appearing as subject of a property p. Let $range_a(p)$ be the set of resource appearing as objects of a object-property p. The SPARQL query given in figure 2 returns the size of the actual domain for a given property (e.g. `dbpedia:hometown`) by selecting and counting distinct instances used as subject of an example object-property `dbpedia:hometown`. Similarly, it computes the size of the actual range.

```
PREFIX dbpedia: <http://dbpedia.org/ontology/>
SELECT COUNT(DISTINCT ?subject) as ?eff_d, COUNT(DISTINCT ?object) as ?eff_r WHERE
{ {?subject dbpedia:hometown []} UNION {[] dbpedia:hometown ?object} }
```

Fig. 2. SPARQL query that gives the sizes of the actual domain and actual range of the given object-property ($domain_a$(`dbpedia:hometown`))

3.2 Definition domain and definition range

Let $domain(p)$ be the definition domain and $range(p)$ be the definition range of the object-property p. $domain(p)$ and $range(p)$ are the subset of respectively the actual domain and the actual range that belong respectively to the domain class and the range class given in the ontology, formely defined by :

$$domain(p) = domain_a(p) \cap resources(D(p))$$
$$range(p) = range_a(p) \cap resources(R(p))$$

The SPARQL query in figure 3 returns the sizes of the definition domain and the definition range of an example object-property `dbpedia:hometown`. This query restricts the query in figure 2 with two additional statements that filter resources that belonging to the domain or the range class of the given object-property.

3.3 Inconsistency definition

We define a domain inconsistency as an occurrence of a subject resource of an object-property p that does not belong to the definition domain of p. Similarly, we define a range inconsistency as an occurrence of an object resource of an object-property p that does not belong to the definition range of p. Let $\varepsilon_d(p)$ the number of domain inconsistencies for the object-property p and $\varepsilon_r(p)$ the number of range inconsistencies for the object-property p defined by the following relations :

$$\varepsilon_d(p) = |\{r \in \mathcal{U}, r \in domain_a(p) \wedge r \notin domain(p)\}|$$
$$\varepsilon_r(p) = |\{r \in \mathcal{U}, r \in range_a(p) \wedge r \notin range(p)\}|$$

By definition $domain(p)$ is a subset of $domain_a(p)$ and $range(p)$ is a subset of $range_a(p)$:

$$domain(p) \subset domain_a(p)$$
$$range(p) \subset range_a(p)$$

therefore the number of inconsistencies is:

$$\varepsilon_d(p) = |domain_a(p)| - |domain(p)|$$
$$\varepsilon_r(p) = |range_a(p)| - |range(p)|$$

```
PREFIX dbpedia: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT COUNT(DISTINCT ?subject) as ?def_d, COUNT(DISTINCT ?object) as ?def_r
WHERE {
{?subject dbpedia:hometown [].
 dbpedia:hometown rdfs:domain ?domain.  ?subject a ?domain }
UNION
{ [] dbpedia:hometown ?object .
  dbpedia:hometown rdfs:range ?range.  ?object a ?range }}
```

Fig. 3. SPARQL query that gives the size of the definition domain and the size of the definition range of an example property ($domain$(`dbpedia:hometown`))

3.4 Evaluation on DBpedia

The DBpedia project [1] provides an ontology to describe 3.5 million of resources based on 843,000 articles from Wikipedia. This large data set plays a central role in the Linked Data overall graph. The ontology of DBpedia is designed by contributors using a wiki. Each type of info-box included in Wikipedia is mapped to a class of the DBpedia's ontology. Subsumption relationships between classes are manually defined. Any article of Wikipedia has an equivalent resource in the data set of DBpedia. Thereby if an article includes an info-box then the equivalent resource belongs to the mapped class. Also keywords of an info-box are mapped to properties and provide relations between resources (object-properties) or informations about the resource (data-properties). In the DBpedia's wiki contributors define the global class hierarchy of DBpedia and the domain and the range for each properties. If different keywords have an identical semantic then the contributors can map them all onto a single property.

The DBpedia's data set is built upon an ontology[11] of 272 classes, 629 object-properties and 706 data-properties. Range and domain of each property are defined by a class or a set of classes. The data are extracted from 843,000 articles of Wikipedia (English version) modeled as a 3.5 million resources graph that complies to the ontology.

Since the DBpedia instances respect the hierarchical inference defined in (1), the SPARQL queries described in this article could be directly tested with the SPARQL endpoint provided by DBpedia.[12] We have evaluated the domain and range inconsistencies defined in section 3.3 on DBpedia for each of object-property.

The DBpedia inconsistencies are depicted in figure 4. The x-axis represents the object-properties sorted by descending number of inconsistencies and the y-axis represents the number of cumulative inconsistencies per object-property. X-axis is in a log scale.

The shape of the curve reveals that the inconsistency are concentrated on a small number of object-properties - located on the left on the vertical dotted line. The fixing of a small number of object-properties can enhance greatly the general consistency of the overall set of data. The method described above lacks of indication of which instances has caused the inconsistency, prevents a detailed tracking of the problem and therefore the debugging of the ontology. In the following parts of this paper, we describe a diagnostic tool that overcome these problems.

[11] `http://dbpedia.org/Downloads36`

[12] `http://dbpedia.org/snorql`

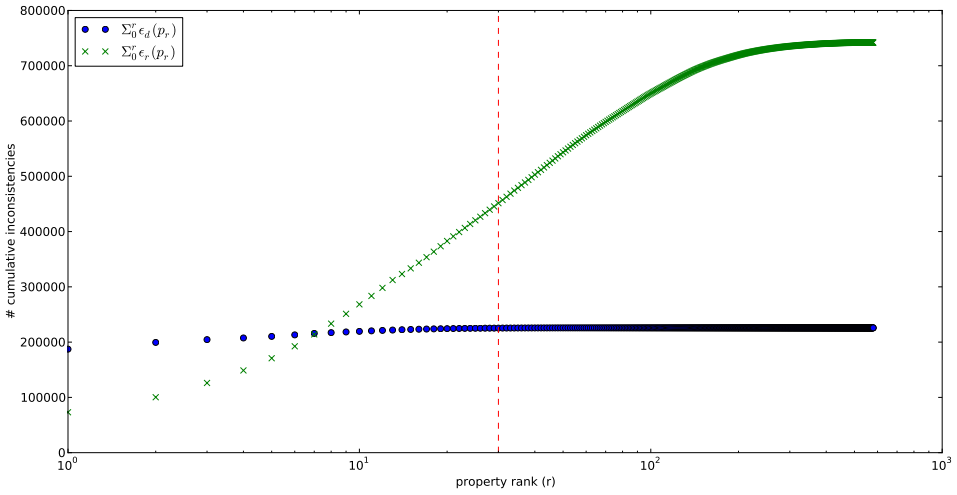

Fig. 4. Number of cumulative inconsistencies per object-property p as a function of the rank of the object property p.

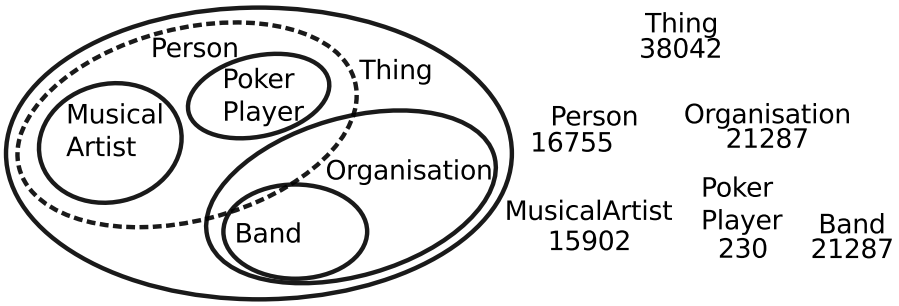

Fig. 5. Class `dbpedia:Person` is the domain of definition of `dbpedia:hometown`. The actual domain, hatched shaded, is divided between the classes: `dbpedia:Person` and `dbpedia:Band`. The values appearing in the class hierarchy are those of the histogram of classes for object-property.

4 Diagnostic tool

In this section we present a tool to diagnose the source of the inconsistencies and how it is possible to find and fix errors. Our tool is based on the histogram distribution of the actual domain and the actual range defined in the above paragraph. Using this histogram, we find the most specific class that include the actual domain and actual range of each property. Following a given schema of questions, an expert could fix each error.

We illustrate this tool on the DBpedia's object-property `dbpedia:hometown`. Its definition is `dbpedia:Person`, but in reality it is used with 21287 resources that do not belong to this class. Figure 5 shows the result of our tool. In the left part, the domain of `dbpedia:hometown` appears in dotted line, its actual domain in shaded (note that the range is not depicted here). In the right part, we have represented the class hierarchy and the number of resources belonging to each class and its herited classes.

4.1 Effective histogram distribution

In the first stage, we calculate the histogram distribution of the resources in the actual $domain_a(p)$ and the actual range $range_a(p)$ of an object-property p in the set of classes $\mathcal{C}$. Let us define two histograms of size $|\mathcal{C}|$: p_d for the resources of the actual domain of p and p_r for the resources involved in the actual range of p :

$$\forall c \in \mathcal{C}, \boldsymbol{p}_d[c] = |ressources(c) \cap domain_a(p)|$$
$$\forall c \in \mathcal{C}, \boldsymbol{p}_r[c] = |ressources(c) \cap range_a(p)|$$

The SPARQL query in the figure 6 generate this histogram for the actual domain of the object-property `dbpedia:hometown`. The query groups instances include in the actual domain by class and count them.

In the case of the object-property `dbpedia:hometown`, the values of the histogram are indicated on the tree of figure 5 for the relevant classes and are 0 for the other classes. The designer of the ontology can visualizes that the object-property `dbpedia:hometown` is involved in 38042 resources, distributed between people and organizations.

```
PREFIX dbpedia: <http://dbpedia.org/ontology/>
SELECT COUNT( DISTINCT ?subject) as ?instances, ?class WHERE
{ ?subject dbpedia:hometown [] . ?subject a ?class .}
```

Fig. 6. SPARQL query gives the actual domain of `dbpedia:hometown` histogram distribution.

4.2 Searching for the most specific class

We use the histogram distribution for an object-property p to suggest a consistent domain class. We select the most specific class, denoted by $\texttt{class}_d(p)$, that includes the actual domain of p. It is obtained by searching the lowest class in the hierarchy $\mathcal{H}$ among the classes containing the actual domain of the object-property. By design, $\texttt{class}_d(p)$ features both the following relations :

$$(i)\ \ \boldsymbol{p}_d[\texttt{class}_d(p)] = \boldsymbol{p}_d[\texttt{owl:Thing}]$$
$$(ii)\ (\forall c \in \mathcal{C})(c \neq \texttt{class}_d(p))\ (\boldsymbol{p}_d[c] < \boldsymbol{p}_d[\texttt{owl:Thing}]) \vee (\texttt{class}_d(p) \prec c)$$

(i) ensures that resources($\texttt{class}_d(p)$) includes $domain_a(p)$ and (ii) ensures that $\texttt{class}_d(p)$ is the most specific class containing $domain_a(p)$.

In the above example (see figure 5), `owl:Thing` is the most specific class for `dbpedia:hometown` that includes the actual domain.

In the same way, we define $\texttt{class}_r(p)$ as the most specific class that includes the actual range of p :

$$(i)\ \ \boldsymbol{p}_r[\texttt{class}_r(p)] = \boldsymbol{p}_r[\texttt{owl:Thing}]$$
$$(ii)\ (\forall c \in \mathcal{C})(c \neq \texttt{class}_r(p)\ (\boldsymbol{p}_r[c] < \boldsymbol{p}_r[\texttt{owl:Thing}]) \vee (\texttt{class}_r(p) \prec c)$$

4.3 Diagnostics and error correction

After having detect potential problems, we will now describe how it is possible to find and fix errors.

In the example of figure 5, there is a problem with `dbpedia:hometown`. The study of histogram class for `dbpedia:hometown` shows that most of the errors come from the instances of `dbpedia:Band`. These instances appear in the object-property `dbpedia:hometown` but do not belong to the domain class `dbpedia:Person`.

A first idea involves the application of the following assumption: each resource used as a subject of a property belongs to the domain of the class of this property.

$$(2)\ r \in domain_a(p) \implies r \in domain_o(p)$$

In the case of `dbpedia:hometown`, this would imply that each music group (instance of `dbpedia:Band`) belongs also to people (instance of `dbpedia:Person`). For example, the resource `dbpedia:The_Beach_Boys` uses the property `dbpedia:hometown` but it is not an instance of `dbpedia:Person`. But adding an inheritance relationship between `dbpedia:The_Beach_Boys` and `dbpedia:Person` may introduce errors on search related to `dbpedia:Person`.

A second solution would rely on the use of two different properties for the music group and for people. This is not acceptable since it would duplicate properties that have the same meaning - some ontology level problem may then arise.

A third idea is to relax the domain of `dbpedia:hometown`. In the DBpedia hierarchy, `owl:Thing` is the only less specific class than `dbpedia:Person`. The problem is that `dbpedia:hometown` is not compatible with every DBpedia's classes.

At least, we think that the best solution in this case, is to create a new class `dbpedia:Agent` as the domain of `dbpedia:hometown`. Both classes `dbpedia:Person` and `dbpedia:Organisation` subsume this new class in such a way that the actual domain coincides with the definition domain.

In general, we propose a diagnostic on erroneous definition domain for property using the following scheme :

IF all instances of actual domain of the property could belong to the domain class THEN the inference (2) applies
ELSE IF the property may has different meaning for different resources THEN create a distinct property for each semantics
ELSE IF it is possible to find a class that includes a part of the actual domain and that complies with the semantics of the property and if the number of errors is acceptable THEN a less specific class is selected as a domain replacement
ELSE the domain of the property is replaced by a new class created such as all classes with non-null value in the histogram subsume the new domain.

5 Evaluation

In this section, we analyze the quality of the results obtained by our approach applied on DBpedia (version 3.6), then we validate the scalability of our method using the benchmark SP2BENCH [13].

5.1 DBpedia analysis

We discuss below the sources of common errors encountered in DBpedia using the diagnostic process presented in paragraph 4.3.

`dbpedia:architect` is dedicated to the instance of buildings (`dbpedia:Buildings` class) but is in fact used in conjunction with `dbpedia:Place` - which is, into the hierarchy, parent to `dbpedia:buildings`. It seems acceptable to assume that every place having an architect is a building. For instance, the Mongomery place in New York is an instance of `dbpedia:Place`. Even if this place has an architect, it does not belong to `dbpedia:Building`. We propose to add the Montgomery place to `dbpedia:Building`.

`dbpedia:class` is the property causing the greatest number of errors. The notion of class is used in two different contexts. According to the ontology, this property should be used with transportation items. The histogram show that this property is mainly used by sub classes of `dbpedia:Species` - which has no relationship with transportation. Species are in no way means of transportation (with exceptions such as horses). In this case, the inference rule (2) see in section 4.3 should not apply.
It seems that DBpedia can have different semantics related to `dbpedia:Species` or `dbpedia:MeansOfTransportation`. We propose to fix this ambiguous property by creating - for each context - new properties in the TBox (and use the correct property in the ABox)

`dbpedia:city` has the range `dbpedia:City`. Resources from its actual range are mainly location instances (dbpedia:Place is a parent class of dbpedia:City). For instance, the triplet (`Cincinnati_bengals`, `dbpedia:city`, `paul_brown_stadium`) means that the American football team Cincinnati Bengals is located in the Paul Brown Stadium. It is therefore delicate to assume that all the places used as objects for `dbpedia:city` are cities. The property seems to have one and only one semantic for all resources in the actual range so it doesn't seem necessary to duplicate it. The most specific class that includes the actual range is `owl:thing`. This class is much too general to be used as range. The `dbpedia:place` is the best trade-off between the number of errors and the semantic consistency (only 37 resources among 17707 of the actual range of `dbpedia:city` are not instances of the `dbpedia:place` class).

5.2 Scalability

In this section we report preliminary results of an experiment that we conduct to evaluate the scalability of our approach. The SPARQL queries presented in this paper could be run on DBpedia's sparql endpoint but sometimes a timeout exception is raised due to the complexity of the query. We have reformulated our queries using the distributed system Hadoop and the request language PIG [10]. In our experimentations, the hierarchy inference, the actual domain size, the definition domain size and the actual histogram domain repartition are computed.

Since we want to study the processing time based on the graph size, the inputs of our tests are generated by SP2BENCH [13] with increasing sizes of data sets. SP2BENCH is mainly designed to evaluate SPARQL engines but it is also useful for scalable data sets generation.

The SP2BENCH generated data sets respect perfectly the associated TBox, we had to introduces inconsistencies by changing a properties's domain in the reference ontology (TBox). One of the five object-properties defined by the reference ontology was corrupted. Since the SP2BENCH data sets don't respect the hierarchy relation (1), the first stage of our experimentation was to compute the closure graph of that inference rule.

Experiments was carried on a 16 nodes Hadoop cluster which can process up to 250 map-reduce tasks in parallel. The cluster is composed of five 2-core Xeon 5060, 8 Go RAM, 2×250 Go hard

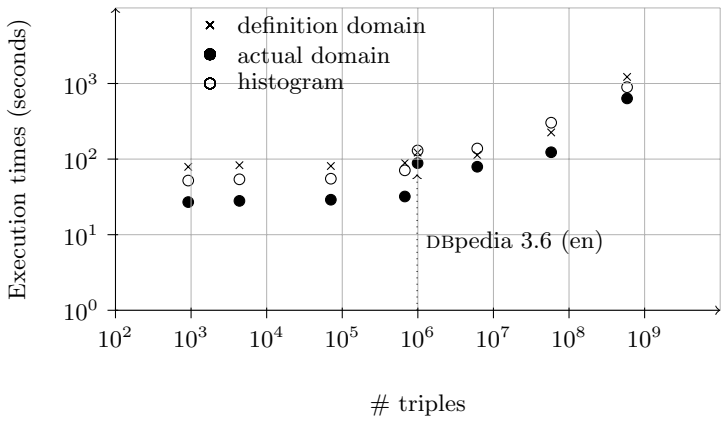

Fig. 7. Execution times required for the computation of actual domain size, definition domain size and actual histogram domain as a function of the number of triplets generated by SP2BENCH.

disk dedicated to Hadoop and twelve dual processors 6-core Xeon L5640, 48 Go RAM, 5×500 Go hard disk dedicated to Hadoop.

Figure 7 shows the results of the mean times of five runs for each data set. We used a log scale for the axis. We observe that the execution time is multiplied by 10^2 when the size of data increases by a factor of 10^7. As comparison with real-world data set, the hierarchical infered DBpedia data set contains $992,558$ triples and the histogram computation is lasts 130 seconds. This results demonstrates that our method scales well and efficiency on very large RDF data sets.

The fixed number of object-properties and classes is the main lack of this experimentation. We are looking for a benchmark that could grow up both the ABox size and the TBox size.

6 Conclusion

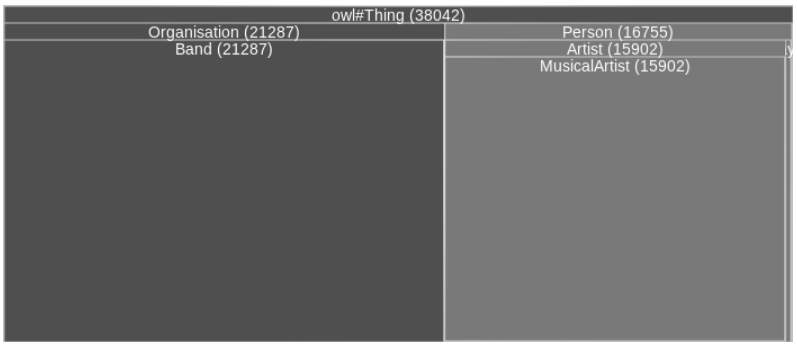

Fig. 8. Tree-map representation of the resources that belongs to the actual domain of the object-property `dbpedia:hometown`. Dark grey square represent inconsistencies resources and light grey one the definition domain of the object-property. Class `dbpedia:Person` is the domain of definition of `dbpedia:hometown`. The actual domain is distributed between two classes: `dbpedia:Person` and `dbpedia:Band`. The numbers near the names of the classes are the matching histogram values.

We have described and evaluated a method designed to analyzes a RDF graph and computes the domain/range inconsistencies number for a given property. We explained how to perform an error diagnosis and proposed ways to fix the ontology accordingly. As an example, we applied this process to DBpedia and suggest dedicated fixing procedures.

We have developed and published an online site that present the results of our system on the DBpedia data set.[13] This prototype is designed to rebuild automatically figure 5 for each object-property of DBpedia. The screen-shot of our application in the figure 8 shows how to use a tree-map to display the repartition of the actual domain of a property.

The experiments carried on large data sets demonstrate the scalability and effectiveness of our method.

Our approach has nevertheless some limitations: we assumed that the class hierarchy is error free so that it is possible to infer new relations. In some cases, the hierarchy relations should be checked or perhaps rebuilt - some systems [12] can compute a new hierarchy based on hierarchy clustering.

We are currently expanding our method to the processing of other OWL constraints such as range restriction (`owl:allValueFrom`), disjunctions (`owl:disjoint`), cardinality (`owl:cardinality`) etc.. The diagnosis could also be enhanced using a semi-supervised process involving the selection of the most relevant example - for instance using page rank like algorithm [6].

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: Dbpedia: A nucleus for a web of open data. The Semantic Web pp. 722–735 (2008)
2. Baader, F.: The description logic handbook: theory, implementation, and applications. Cambridge Univ Pr (2003)
3. Baclawski, K., Matheus, C., Kokar, M., Letkowski, J., Kogut, P.: Towards a symptom ontology for semantic web applications. In: ISWC. p. 650–667 (2004)
4. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. Int. J. Semantic Web Inf. Syst. 5(3), 1–22 (2009)
5. Gomez-Perez, A.: Evaluation of ontologies. International Journal of Intelligent Systems 16(3), 391–409 (2001)
6. Hogan, A., Harth, A., Decker, S.: Reconrank: A scalable ranking method for semantic web data with context. In: 2nd Workshop on Scalable Semantic Web Knowledge Base Systems. Citeseer (2006)
7. Hogan, A., Harth, A., Passant, A., Decker, S., Polleres, A.: Weaving the pedantic web. In: 3rd International Workshop on Linked Data on the Web (LDOW2010), in conjunction with 19th International World Wide Web Conference, CEUR (2010)
8. Hogan, A., Harth, A., Polleres, A.: Scalable authoritative OWL reasoning for the web. Information Systems 5(2), 49–90 (2009)
9. Kalyanpur, A., Parsia, B., Sirin, E., Hendler, J.A.: Debugging unsatisfiable classes in owl ontologies. Journal of Web Semantics 3(4), 268–293 (2005)
10. Olston, C., Reed, B., Srivastava, U., Kumar, R., Tomkins, A.: Pig latin: a not-so-foreign language for data processing. In: Proceedings of the 2008 ACM SIGMOD international conference on Management of data. pp. 1099–1110. ACM (2008)
11. Parsia, B., Sirin, E., Kalyanpur, A.: Debugging owl ontologies. In: International World Wide Web Conference. pp. 633–640 (2005)
12. Quan, T., Hui, S., Cao, T.: A fuzzy FCA-based approach to conceptual clustering for automatic generation of concept hierarchy on uncertainty data. In: Proc. of the 2004 Concept Lattices and Their Applications Workshop. pp. 1–12. Citeseer (2004)
13. Schmidt, M., Hornung, T., Lausen, G., Pinkel, C.: SP^ 2Bench: A SPARQL Performance Benchmark. In: Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on. pp. 222–233. IEEE (2009)
14. Tao, J., Ding, L., McGuinness, D.L.: Instance data evaluation for semantic web-based knowledge management systems. In: Proceedings of the 42th Hawaii International Conference on System Sciences (HICSS-42). pp. 5–8. Big Island, Hawaii (2009)
15. Tolle, K.: Validating RDF Parser: A Tool for Parsing and Validating RDF Metadata and Schemas. Master's thesis, University of Hannover (2000)
16. Urbani, J., Kotoulas, S., Oren, E., van Harmelen, F.: Scalable distributed reasoning using mapreduce. The Semantic Web-ISWC 2009 pp. 634–649 (2009)

[13] `http://cluster-valoria.univ-ubs.fr/swoct/swoct/`