Evaluating Class Assignment Semantic Redundancy on Linked Datasets

Leandro Mendoza
CONICET, Argentina
LIFIA, Facultad de Informática,
UNLP, Argentina

Alicia Díaz
LIFIA, Facultad de Informática,
UNLP, Argentina

Abstract

In this work we address the concept of *semantic redundancy* in *linked datasets* considering *class assignment* assertions. We discuss how redundancy can be evaluated as well as the relationship between redundancy and three class hierarchy aspects: the number of instances a class has, number of class descendants and class depth. Finally, we performed an evaluation on the DBpedia dataset using SPARQL queries for data redundancy checks.

1 Introduction

The amount of interlinked knowledge bases built under *Semantic Web* technologies and following the *linked data* (Heath and Bizer, 2011) principles has increased significantly last years. These knowledge bases (also known as *linked datasets*) contain information that associates *Web* entities (called *resources*) with well-defined semantics that specifies how these entities should be interpreted. In most linked datasets a substantial amount of data corresponds to *class assignment* assertions, that is, information that specifies resources (or individuals) as instances of certain classes. In this sense, resources are typified using classes usually defined through ontologies and organized into class hierarchies. Several different ontologies can be combined to classify resources within the same dataset giving rise to huge and complex interlinked structures that can suffer from data quality problems (Hogan et al., 2010). Thus, the use of practical mechanisms to handle knowledge conciseness becomes increasingly important to improve the overall dataset quality and the study of redundancy on class assignments assertions aims to contribute in this way. From a data quality perspective, class assignment redundancy is related with the concept of *extensional conciseness* which has been defined in (Zaveri et al., 2015) as "the case when the data does not contain redundant objects at instance level". In this scenario, redundancy means that a resource is specified as member of a class when it is not necessary, either because the information is explicitly duplicated or because it can be derived from information that already exists. Current works that have dealt with semantic redundancy on linked datasets implement algorithms based on graph pattern discovering techniques. In contrast, our work proposes a simplified approach based on SPARQL queries and considering class assignment assertions. We discuss how redundancy can be evaluated and perform an evaluation over the DBpedia dataset (Lehmann et al., 2015) in order to understand the relationship between redundancy and three class hierarchy aspects: the number of instances a class has, the class depth and its number of descendants. This approach may be useful for linked data users who need to measure semantic data redundancy in a practical way, understand its origin and detect when it may be useful (e.g. to improve performance) or when it can affect negatively the knowledge base (e.g. misuse of classes when typifying resources). The following sections are organized as follows: sections 2 and 3 give some background definitions and related work, respectively. Section 4 introduces the redundancy definition adopted and discusses some of the alternatives to address it on linked datasets. Section 5 shows the evaluation results. Finally, some conclusions and further work are given in section 6.

2 Background

In the linked data context, datasets are knowledge bases described using the RDF[1] data model

[1] https://www.w3.org/RDF/

and published following the *linked data principles*[2]. These datasets are collection of assertions about *resources* specified following the "*subject predicate object*" pattern. Assertions are RDF triples and resources may be anything identifiable by an HTTP URI. Knowledge representation mechanisms like RDFS[3] and OWL[4] extend RDF and allow datasets to be augmented with more expressive semantics. For example, it is possible to describe ontologies by specifying classes and relationships between them (e.g. "*SoccerPlayer* `rdfs:subClassOf` *Atlhete*") and to specify resources as member of those classes (e.g. "*Lionel_Messi* `rdf:type` *SoccerPlayer*"). From an overall perspective, information contained in these datasets can be split into two levels: *schema level* and *instance level*. *Schema level* refers to *terminological knowledge* (known as TBox), for example, classes, properties and their relationships. On the other hand, *instance level* refers to *assertional knowledge* (known as ABox), that is, propositions about entities of a specific domain of interest. An important type of assertional knowledge corresponds to *class assignments*, that is, RDF statements of the form "*resource* `rdf:type` *class*" used to specify resources as members of certain classes. The most common way to retrieve this information from linked datasets is through a SPARQL[5] endpoint. These endpoints are web services that accept SPARQL queries and return information that match with a given pattern. In this work we will use this mechanism to detect redundant class assignments.

3 Related Work

In the linked data literature, redundancy is related with the data quality dimension of *conciseness* (Zaveri et al., 2015) and has been studied and categorized from syntactic to semantic and from schema to instance levels (Pan et al., 2014). From a syntactic perspective most of the existing compression techniques focus on RDF serialization. On the other hand, from a semantic perspective, just a few works addressed redundancy. In (Wu et al., 2014) authors propose a graph based analysis method to identify graph patterns that can be used to remove redundant triples and calculate the volume of semantic redundancy. In (Joshi et al., 2013) authors employ frequent itemset (*frequent pattern)* mining techniques to generate a set of logical rules to compress RDF datasets and then use these rules during decompression. Both works mention the idea of semantic compression by removing derivable knowledge. Regarding the use of SPARQL for quality assessment, (Fürber and Hepp, 2010) and (Kontokostas et al., 2014) use query templates to detect some quality problems but semantic redundancy is not included. Inspired on the ideas of these works, we use a SPARQL query oriented approach to evaluate redundant class assignments and make this information explicit to users.

4 Redundant class assignments

As we know, linked datasets are basically sets of RDF triples and knowledge is specified using mechanisms provided by RDFS and OWL, each one with its own well-defined semantics (Hitzler et al., 2009). In this way, *schema* and *instance* level assertions can be considered as *propositions* to formally describe the notion of derivable knowledge. For example, the notation $\{p_1, p_2\} \models \{p_3, p_4\}$ (where $\models$ is called *entailment relation*) states that propositions p_3 and p_4 (also p_1 and p_2) are logical consequences of propositions p_1 and p_2 obtained under a certain set of rules (logic). Considering this, the concept of *data redundancy* can be associated to what is known in mathematical logic literature as *independence*, that is, the ability to deduce a proposition from other propositions. Formally, given a logic L (semantics) and a set of propositions P, it is defined as *independent* if for all proposition $p_i \in P$ does not hold that $\{P - p_i\} \models p_i$. In this way, a *non-independent* set of propositions can be considered redundant since it contains extra information that may not be necessary because if it is removed from the initial set, it can be obtained from the remaining propositions applying an inference mechanism and keeping the same *logical consequences*. Similarly, an *independent set* of propositions can be considered *non-redundant*.

As we mentioned in section 2, *class assignments* are *instance level* assertions (or propositions) that specify resources as members of certain classes. Thus, given a resource r, its class as-

[2] `https://www.w3.org/DesignIssues/LinkedData.html`

[3] `https://www.w3.org/TR/rdf-schema/`

[4] `https://www.w3.org/standards/techs/owl\#w3c\_all`

[5] `https://www.w3.org/TR/rdf-sparql-query/`

signment set (CAS_r) contains all the propositions that specify the classes to which r belongs. The idea of the previous paragraph can be applied to *class assignments* to define semantic redundancy since the **non-redundant** class assignment set of a resource r ($NRCAS_r$) is the *independent* set of CAS_r. Then, the **redundant** class assignment set ($RCAS_r$) can be considered as the difference of those sets. In the following subsections, we discuss some techniques that can be used to compute $NRCAS_r$ on linked datasets. Then, we will use one of these techniques to perform our evaluation.

4.1 Using SPARQL queries

Using SPARQL, a simple query can be implemented to get the $NRCAS$. For example, query in listing 1 can be used to get the non-redundant set of classes of a given resource (specified by *resource_URI*).

```
SELECT DISTINCT ?c
WHERE {
 <resource_URI> rdf:type ?c
 FILTER regex(str(?c),"ont_URI","i")
 FILTER NOT EXISTS {
   <resource_URI> rdf:type ?sc .
   FILTER regex(str(?sc),"ont_URI","i")
   ?sc rdfs:subClassOf ?c }
}
```

Listing 1: SPARQL query example to get non-redundant class assignments

Note that the mentioned query example considers only one ontology (filtered by *ontology_URI*) and does not implement any inference mechanism at instance or schema level. This means that the query will work while all class assignments and relationships between the involved classes will be specified explicitly on the dataset. If this is not the case and a transitive closure of sub/super classes is needed, it is necessary to implement an algorithm that iterates recursively over these queries until it gets the required classes. Using SPARQL *property paths* (e.g. `rdfs:subClassOf*` or `rdfs:subClassOf+`) it is possible to check connectivity of two classes by an arbitrary length path (route through a graph between two graph nodes)[6]. For example, it can be used to get all the classes that are descendant (or subclasses) of a given class (as shows example of listing 2) or to get all the ancestors (or depth) of a given class (as shows example of listing 3).

[6]https://www.w3.org/TR/sparql11-property-paths/

```
SELECT DISTINCT ?c
WHERE {
 ?c rdf:subClassOf* <class_URI>
}
```

Listing 2: SPARQL query to get class descendants

```
SELECT DISTINCT ?c
WHERE {
 <class_URI> rdf:subClassOf* ?c
}
```

Listing 3: SPARQL query to get class ancestors

It is important to highlight that the performance of SPARQL queries depends on its implementation and the dataset size. Although complex SPARQL queries can become unacceptably slow when working with large amounts of data, it is currently the most practical mechanism to access linked datasets.

4.2 Using graph based algorithm and reasoners

Given a class hierarchy, a resource r and its CAS_r, a way to compute redundancy is by interpreting the class hierarchy as a directed acyclic graph "G" in which each node is a class and each edge is the relation `rdfs:subClassOf`. A node "A" of "G" can be considered a class if there exist a triple with the form "A `rdfs:subClassOf` x", "x `rdfs:subClassOf` A" or "x `rdf:type` A". Then, a class "B" is subclass of a class "A" if node "A" is reachable from node "B" in "G", that is, if exists a path between B and A in the graph. Considering this, given a proposition set Q that specifies the classes to which an instance i belongs, a naive algorithm can be implemented to compute a *non-redundant* proposition set R: first set R=Q, then for each element q in Q check if there is a path from some of the remaining propositions in Q to q, if so, q is deleted from R. Finally, the algorithm returns R which is then the *non redundant class assignments set* of r. Removing redundancies can be associated with the problem known as *transitive reduction* (Aho et al., 1972) which has an unfortunate complexity class if it is

implemented naively. If the given graph is a finite directed acyclic graph current approaches that solve this problem are close to the upper bound $O(n^{2.3729})$ but if the graph has cycles the problem belongs to NP-hard class.

Another alternative to compute redundancy is by using *Semantic Web* reasoners that have been implemented based on decidable fragments of RDFS and OWL semantics. These tools implement inference mechanisms that can be used to deduce if a resource belongs to a given class (instance checking). The main advantage of using reasoners is that the potential of the underling semantics can be exploited (e.g. several ontologies can be combined to get implicit knowledge). On the other hand, the disadvantage of using these tools in linked data scenario is its complexity: inference techniques work well for small examples with limited knowledge but they turn unacceptably slow for large-scale datasets. Besides, when multiples ontologies are combined, inconsistencies can arise affecting the inference process and hampering the detection of redundant propositions.

5 Evaluation

To perform our evaluation we selected the English version of DBpedia[7] and set up a local mirror using a Virtuoso[8] server (version 7.2) . The mechanisms implemented to compute redundant class assignment avoid the use of complex graph based algorithms or RDFS/OWL reasoners and use a SPARQL query oriented approach (see section 4.1). Although resources in DBpedia are classified using several classes of different schema we only considered the DBpedia[9] core and YAGO[10] ontologies because information about the involved class hierarchies (subclass relationships) can be obtained directly from queries through the dataset SPARQL endpoint. DBpedia ontology is a shallow cross-domain ontology that covers more than 600 classes and was created based on Wikipedia *infoboxes*. YAGO is a taxonomy used in the YAGO knowledge base that currently covers more than 350,000 classes. The evaluation is organized in the next subsections as follows: we first performed an overall redundancy evaluation considering DBpedia and YAGO ontologies and then a further analysis was done per class groups but only considering the DBpedia class hierarchy in order to keep the number of classes manageable. For each class group, we analyzed the relationship between redundant class assignments (RCA) and three class hierarchy characteristics: class depth, class descendants and number of class assignments per class.

5.1 Overall redundancy evaluation

The first overall evaluation was made by retrieving all resources that belong to some class of the DBpedia ontology (6,729,604 resources of 453 classes) and then we did the same with the YAGO ontology (2,886,306 resources of 369,144 classes). For each resource we compute its CAS, its $NRCAS$ and its $RCAS$ (see section 4) considering both ontologies separately. Information about resources and its CAS and $NRCAS$ were obtained through SPARQL queries (see section 4.1) and $RCAS$ was obtained by computing the difference $CAS - NRCAS$. Results can be viewed in table of figure 1. Each element of each CAS was counted as a different class assignment (*nbCA* column), each element of $NRCA$ was counted as a non-redundant class assignment (*nbNRCA* column) and each element of RCA was counted as a redundant class assignment (*nbRCA* column). As we can see in chart of figure 1, considering classes of the DBpedia ontology almost half class assignments are redundant. On the other hand, considering the YAGO ontology 80% of class assignments are redundant. In the latter case, the amount of class assignments is higher and the amount of concepts in the class hierarchy increases considerably. These results suggest a relationship between the number of classes, class assignments and redundancy: as the number of classes and class assignments increases, so the probability of redundancy.

5.2 Redundancy and class depth

To analyze the relationship between redundant class assignments and class depth we categorized classes into groups from 0 to 6 according to their depth in the DBpedia class hierarchy (the distance from the root to that class) and then we count how many class assignments refer to those classes. Classes with depth 0 are the most general and 6 is the max depth found in the class hierarchy. A class assignment refers to (or belongs to) a class C if it is a triple of the form ($resource$ `rdf:type`

[7] http://wiki.dbpedia.org/Downloads2015-04

[8] http://virtuoso.openlinksw.com/

[9] http://wiki.dbpedia.org/services-resources/ontology

[10] https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/

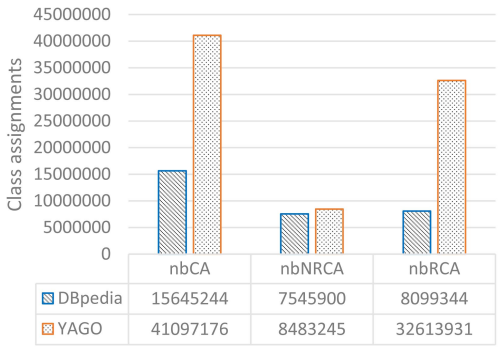

Figure 1: DBpedia and YAGO redundancy evaluations

C). To compute the depth of a class we used a SPARQL query to count the number of ancestors (see section 4.1 listing 3). Results can be viewed in table 1, chart of figure 2 shows the relationship between the class depth and the percentage of redundant class assignments (%RCA) and chart of figure 3 shows how these redundant class assignments are distributed.

Depth	nbClasses	nbCA	nbRCA
0	32	5,037,966	3,940,920
1	86	3,941,230	2,877,434
2	110	5,385,831	1,156,327
3	180	1,154,660	118,886
4	39	118,891	5,777
5	5	5,777	0
6	1	889	0

Table 1: Redundancy and class depth evaluation.

As we can see on chart of figure 2, as the class depth increases (more specific a class is), the number of redundant class assignments decreases.

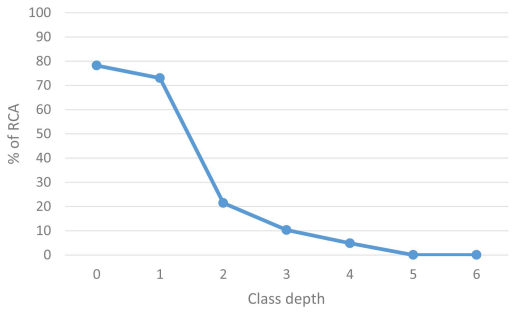

Figure 2: RCA vs class depth

Chart of figure 3 shows that more than 80% of redundant class assignments refer to more general classes (with less depth). For example, in table 1 we can see that there are 32 classes (*nbClasses* column) of depth 0 (*Depth* column), 5,037,966 class assignments (*nbCA* column) that refer to those classes and 3,940,920 of them are redundant (*nbRCA* column). Examples of classes that belong to that group are *Agent*, *Place*, *Work*, etc.

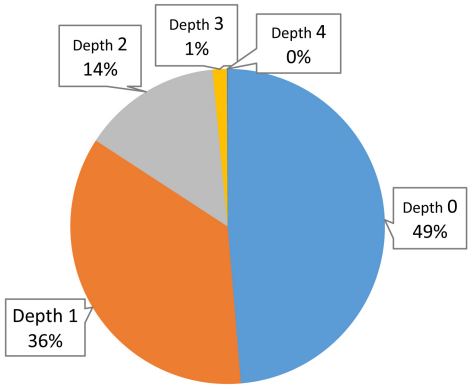

Figure 3: RCA distribution considering the class depth

5.3 Redundancy and class descendants

To analyze the relationship between redundant class assignments and class descendants we categorized classes into 10 groups according to the number of descendants that they have. To compute the descendants we used a SPARQL query to get the subclasses of a given class (see section 4.1 listing 2). Results are showed in table 2 and chart of figure 4 shows the relationship between the number of class descendants and the percentage of redundant class assignments (%RCA). Chart of figure 5 shows how these redundant class assignments are distributed. Classes that do not have descendants (330 classes) are the most specific and class assignments that belong to that group are not redundant. As we can see in chart of figure 4, when the number of descendant per class increases, the number of redundant class assignments also increases. For example, group named "1 to 5" refers to classes that have between 1 to 5 descendants (78 classes) and redundancy is relatively low. On the other hand, classes with several descendants (e.g. class *Agent*) has a high level of redundancy. As chart of figure 5 shows, only 3 classes have more than 100 descendants (*Agent*, *Person* and *Place*) and they concentrates the 65% of redundant class assignments.

nbDesc	nbClasses	nbCA	nbRCA
0	330	3,250,543	0
1 to 5	78	3,209,728	321,078
6 to 10	16	948,187	524,262
10 to 20	13	666,630	531,848
21 to 30	8	773,316	751,488
31 to 50	2	535,606	502,892
51 to 70	3	809,171	784,103
71 to 100	1	220,219	211,415
101 to 200	2	2,860,585	2,117,098
More than 200	1	5,231,844	4,472,258

Table 2: Redundancy and class descendants evaluation.

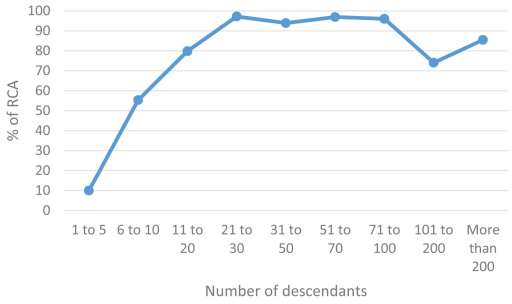

Figure 4: RCA vs class descendants

5.4 Redundancy and number of class assignments

To analyze the relationship between redundant class assignments and the number of class assignments per class we categorized classes into 10 groups according to the number of class assignments that refers to a class. Table 3 shows the evaluation results and chart of figure 6 shows the relationship between redundancy and the number of class assignments (or instances) per class. Columns *nbCA-acc* and *nbRCA-acc* show the total number of class assignments and redundant class assignments in each group.

nbCA	nbClasses	nbCA-acc	nbRCA-acc
0 to 10K	315	599,261	101,861
10K to 20K	35	484,931	120,512
20K to 30K	34	588,845	267,460
30K to 40K	25	384,601	220,270
40K to 30K	11	900,290	36,156
100K to 200K	7	906,730	365,085
200K to 300K	5	1,274,010	1,222,844
300K to 400K	1	396,046	395,804
400K to 500K	2	934,843	681,445
More than 500K	6	9,292,013	4,472,258

Table 3: Redundancy and number of class assignments evaluation.

As we can see, most of classes (315) have

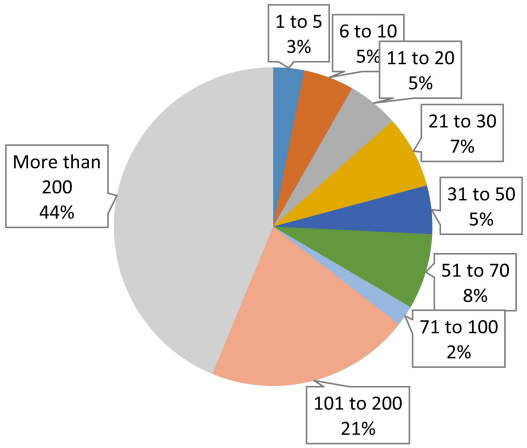

Figure 5: RCA distribution considering class descendants

less than 10K class assignments. Besides, as this number increases the number of classes involved decreases but the percentage of redundant class assignments increases. For example, classes that have more than 500K class assignments (e.g. *Agent*, *Place*, *Person*, etc.) concentrate most of them (9,292,013) and 48% are redundant. We also observe that the second group of most used classes (between 200K and 500K class assignments) consists of about 8 classes with high levels of redundancy (between 70% and 95%).

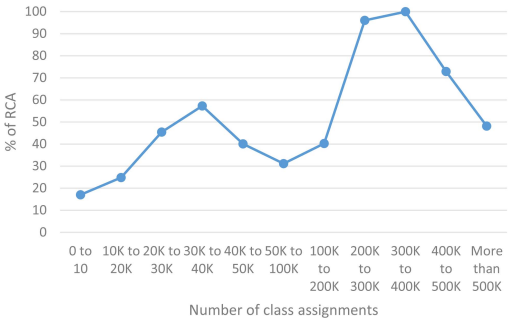

Figure 6: RCA vs number of class assignments

6 Conclusions and future work

In this work we addressed the concept of *semantic redundancy* considering *class assignments* assertions in *linked datasets*. Based on a formal definition we discussed how *redundant* (and *non-redundant*) class assignments sets can be detected. Inspired in previous related work, we conducted an evaluation over the English version of DBpedia based on SPARQL queries. We analyzed the relationship between redundancy and three class hi-

erarchy characteristics: the number of instances a class has, its depth and its number of descendants.

Regarding the evaluation results, they suggest that there is a relationship between redundancy and depth of a class in a hierarchy: as more general a class is (less depth), more redundant class assignments can be found that refer to that class. In a similar way, as the number of descendants per class increases so does the number of redundant class assignments related to that class. Particularly, we noted that this also occurs in most populated classes (with more class assignments). In this sense, datasets that use complex and large class hierarchies to typify their resources in uncontrolled environments (such as *crowsourced* generated content) may be more prone to *class assignments* redundancy. Considering this, we can make the following observations:

- SPARQL queries as a mechanism to evaluate redundancy offer practical ways to implement quality checks and get some statistics of linked datasets. However, SPARQL inference capabilities are limited: we can discover just some graph patterns using queries but other implicit knowledge will be *unreachable* since we can not exploit all the semantic capabilities or expressiveness of the used languages.

- Redundancy analysis can be used to detect class assignments patterns and data publishers behaviors. For example, in our evaluation we detected that when a resource is assigned to a very specific class, it is also assigned explicitly to the ancestors of that class. Discovering these kinds of patterns may be useful to improve the linked data generation process or even to understand how classes described in a given ontology are used on a specific dataset.

Future work will be focused on the development and assessment of semantic redundancy metrics that support our results on other linked datasets. Besides, we plan to study how redundancy can affect other data quality dimensions particularly those related with semantic accuracy.

References

A. V. Aho, M. R. Garey, and J. D. Ullman. 1972. The transitive reduction of a directed graph. *SIAM Journal on Computing*, 1(2):131.

Christian Fürber and Martin Hepp. 2010. Using semantic web resources for data quality management. In *Proceedings of the 17th International Conference on Knowledge Engineering and Management by the Masses*, EKAW'10, pages 211–225, Berlin, Heidelberg. Springer-Verlag.

Tom Heath and Christian Bizer. 2011. *Linked Data: Evolving the Web into a Global Data Space*, volume 1 of *Synthesis Lectures on the Semantic Web: Theory and Technology*. Morgan Claypool, 1st ed., html version edition, February.

Pascal Hitzler, Markus Krötzsch, and Sebastian Rudolph. 2009. *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC.

Aidan Hogan, Andreas Harth, Alexandre Passant, Stefan Decker, and Axel Polleres. 2010. Weaving the pedantic web. In *Linked Data on the Web Workshop (LDOW2010) at WWW'2010*, volume 628, pages 30–34. CEUR Workshop Proceedings.

Amit Krishna Joshi, Pascal Hitzler, and Guozhu Dong, 2013. *The Semantic Web: Semantics and Big Data: 10th International Conference, ESWC 2013, Montpellier, France, May 26-30, 2013. Proceedings*, chapter Logical Linked Data Compression, pages 170–184. Springer Berlin Heidelberg, Berlin, Heidelberg.

Dimitris Kontokostas, Patrick Westphal, Sören Auer, Sebastian Hellmann, Jens Lehmann, Roland Cornelissen, and Amrapali Zaveri. 2014. Test-driven evaluation of linked data quality. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 747–758, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2015. DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 6(2):167–195.

Jeff Z Pan, JM Gómez-Pérez, Yuan Ren, Honghan Wu, and Man Zhu. 2014. Ssp: Compressing rdf data by summarisation, serialisation and predictive encoding. Technical report, Technical report, 07 2014. Available as http://www. kdrive-project. eu/resources.

Honghan Wu, Boris Villazn-Terrazas, Jeff Z. Pan, and Jos Manul Gmez-Prez. 2014. How redundant is it? - an empirical analysis on linked datasets. In Olaf Hartig, Aidan Hogan, and Juan Sequeda, editors, *COLD*, volume 1264 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Amrapali Zaveri, Anisa Rula, Andrea Maurino, Ricardo Pietrobon, Jens Lehmann, and Sören Auer. 2015. Quality assessment for linked data: A survey. *Semantic Web Journal*.