Using Semantic Web Resources for Data Quality Management

Christian Fürber and Martin Hepp

E-Business & Web Science Research Group, Werner-Heisenberg-Weg 39,
85577 Neubiberg, Germany
christian@fuerber.com, mhepp@computer.org

Abstract. The quality of data is a critical factor for all kinds of decision-making and transaction processing. While there has been a lot of research on data quality in the past two decades, the topic has not yet received sufficient attention from the Semantic Web community. In this paper, we discuss (1) the data quality issues related to the growing amount of data available on the Semantic Web, (2) how data quality problems can be handled within the Semantic Web technology framework, namely using SPARQL on RDF representations, and (3) how Semantic Web reference data, e.g. from DBPedia, can be used to spot incorrect literal values and functional dependency violations. We show how this approach can be used for data quality management of public Semantic Web data and data stored in relational databases in closed settings alike. As part of our work, we developed generic SPARQL queries to identify (1) missing datatype properties or literal values, (2) illegal values, and (3) functional dependency violations. We argue that using Semantic Web datasets reduces the effort for data quality management substantially. As a use-case, we employ Geonames, a publicly available Semantic Web resource for geographical data, as a trusted reference for managing the quality of other data sources.

Keywords: Semantic Web, Ontologies, Data Quality Management, Ontology-Based Data Quality Management, Metadata Management, SPARQL, Linked Data, Geonames, Trust

1 Introduction

Data is the source for almost every business transaction or decision and also has become increasingly important for our social activities. With the current evolution of the internet from a "Web of Documents" to a "Web of Data", huge amounts of business-relevant data is being published on the Web. That data may be used to increase automation of business operations and supporting processes of our social activities. Hence, it becomes critical to manage the correctness and reliability, i.e., the quality of data on the Web. It is a well-known fact that when using data for business cases, data quality problems can influence the satisfaction of customers and employees, produce unnecessary costs, and cause missed revenues [1]. Eventually, performing business processes based on poor data can be very expensive. Yet in 1998,

the average total costs of poor data quality have been estimated to be 8 – 12 % of a company's revenues [2]. In the meanwhile, the automation of business processes and likewise the production and consumption of data have increased. Thus, the impact of poor data quality will likely be even higher today. Despite its importance to the overall success for the adaption of the Semantic Web, data quality is currently missing in the well-known Semantic Web layer cake diagram published by the World Wide Web Consortium (W3C) [3]. At present, there are only a few research approaches addressing data quality management with the use of Semantic Web technologies. Even less approaches provide means for quality management of data published on the Semantic Web.

In this paper, we (1) define typical problems that may occur on the data instance level in Semantic Web resources, (2) describe how to identify data quality problems of literal values in the Web of Data, and (3) show how we can use Semantic Web technology and datasets for data quality management of knowledge bases or local relational sources. As part of our proposal, we present SPARQL queries for data quality checks that can be executed completely within the Semantic Web technology stack.

2 Data Quality Management on the Semantic Web

In data quality research, high data quality is often described as data that is "fit for use" [4]. This definition relies on the subjective judgment of data quality by data consumers. Usually data consumers consider data to be of high quality and, therefore, "fit for use", when data meets their requirements. *Wang, et al.* have analyzed this perspective in more detail and identified 15 essential dimensions of data quality for data consumers, such as accuracy, completeness, accessibility, and relevance [4]. The perspective of a data consumer on data quality is of high practical relvance, in particular during data presentation.

From the technical perspective, high quality data is data that is "free of defects" [5]. Several typologies have been developed to classify respective defects in data [6-10]. These categorizations of data quality problems are especially suitable for the development of algorithms for the identification and improvement, as they provide insight into their technical characteristics. Because we aim at automated tools for data quality management and because the context of data consumption is often not immediately available on the Web of Data, we adopt the technical perspective on data quality to develop algorithms for the identification of data quality problems in knowledge bases. For a summary of instance-related data quality problems found in single-source scenarios, we refer to our previous work published in [11]. In the following, we summarize the most important types of defects for Semantic Web data.

2.1 Missing Literal Values

Missing values are values of certain properties that are missing, even though they are needed, either in general or within a certain context. For instance, the literal value for

the property `country` might always be mandatory in address data, while the literal values for the property `state` might only be required for instances with the literal value "USA" in the datatype property `country`. In other words there are at least two types of missing values: (1) values that must not be missing at all, and (2) values that must not be missing in certain contexts. In Semantic Web data, missing literal values can be either in the form of (1) an empty literal attached to a datatype property, or by (2) the absence of a particular datatype property for a certain instance. Solution algorithms for the identification of missing values in Semantic Web scenarios have to consider both patterns. The first case will often be found when RDF data is derived automatically from relational sources and no sanity checks for empty attributes are included in the transformation process. The popular Semantic Web repository at http://loc.openlinksw.com/sparql, for instance, includes more than 3 Million triples from more than 44,000 RDF graphs with an empty literal as the object of at least one triple.

It is important to note that (1) standard database-style cardinality constraints cannot be modeled in neither RDFS nor OWL, and that (2) the constraints in real-world settings are often more subtle than simple validity intervals for the frequency of a particular property.

2.2 False Literal Values

False literal values are either (1) imaginary values that do not have a corresponding state in reality, (2) values that may exist in reality, but do not represent the correct state of an object, (3) values that are supposed to represent the current state of an object, but use the wrong syntax or are mistyped, and (4) values that represent an outdated real-world state of an object. In cases where values are part of functional dependencies, false values may be discovered by the use of queries for the identification of functional dependency violations (see below). Other cases require a separate set of query elements that can identify illegal states solely by the definition of legal or illegal states for an object. In many cases, the actual set of valid values is a small subset of the lexical space defined by the standard datatype (e.g. xsd:string or xsd:int).

Checks for false literal values can be performed with different levels of accuracy by (1) the definition of legal values or value ranges, (2) the definition of illegal values or value ranges, (3) the definition of syntax patterns for legal or illegal values, e.g. by the use of regular expressions, or (4) the definition of valid time ranges for sufficiently current values (only applicable in cases where time data about the actuality of values is available). In this paper, we focus on the use of legal value lists for the identification of illegal values. The identification of legal but outdated values is part of our future work, because it requires meta-data about the temporal properties of the datasets.

2.3 Functional Dependency Violations

Functional dependencies can be defined as dependencies between the values of two or more different properties [12], e.g. the value for the datatype property `prop:city` may depend on the value for datatype property `prop:zipcode`. More formally we can express a functional dependency between the literal x for datatype property A and the dependent literal y for the datatype property B as shown in (1).

$$A\ (x) \rightarrow B\ (y) \quad (1)$$

$$FDV: \{y \mid y \notin V_c\} \quad (2)$$

A functional dependency violation (FDV) occurs when the dependent literal y obtains a value outside of the correct value set V_c. Thereby, V_c can consist of exactly one correct value or a set of correct values. In a lot of real world cases, literal values are not unique, even when they aim at uniquely representing only one real-world entity. This circumstance is mainly caused by the existence of homonyms and other forms of lexical variety (e.g. British vs. American English). For example, the same city name may have different legal zip codes if the city name is a homonym, which is used for multiple different cities around the world. The city name "Neustadt", for instance, which is used for at least 48 different cities according to Geonames data[1]. Accordingly, it may have way more than 48 different valid zip codes. The theoretical problem underlying such cases is that literal values in RDF data cannot be mapped easily to a single conceptual entity, because this disambiguation is regularly prohibitively expensive in real-world settings.

In short, homonymous literal values may have a set of legal dependent values even if the individual entity can only obtain a single correct value. Likewise, different countries may assign identical zip codes for different cities since there is no global authority that enforces unique zip codes worldwide. Hence, zip codes may also be homonyms on a global scale, unless we use a strict prefix system.

Thus, the identification of illegal combinations of literal values has to be tolerant to homonyms. The major drawback of this tolerance to homonyms is that we will be unable to spot values that are within the set of valid lexical representations, but incorrect in that particular case.

In section 3.5, we show how functional dependency violations can be identified using trusted knowledge bases, e.g. Semantic Web datasets, as references. We will provide queries that minimize undiscovered incorrect value combinations and are tolerant to homonymous literal values.

[1] http://www.geonames.org/

3 Identification of Data Quality Rule Violations in the Web of Data

In the Semantic Web technology stack, there are multiple options to identify data quality problems. For instance, it is possible to use the formal semantics of an underlying ontology [cf. 14], such as disjointness axioms, to identify data quality problems in knowledge bases referring to that ontology. On the other hand, data quality problems can be identified by the definition of queries in SPARQL[2], the query language for Semantic Web data. Additionally, we could define conceptual elements in the ontology for the annotation of data to enable data quality problem identification algorithms through SPARQL. In this paper, we focus on the second option. The other two options are subject to our future work.

3.1 Methodology for Improving Data Quality

Data and information quality management has been addressed in database-oriented research for nearly two decades. Total Data Quality Management (TDQM) as proposed by *Wang* [13] is one of the most prominent methodologies for managing data quality. Based on the principle that the quality of data needs to be managed similar to the management of product quality, it describes an adjusted lifecycle of quality management suitable for data. The TDQM methodology encompasses four phases, namely (1) the definition phase, (2) the measurement phase, (3) the analysis phase, and (4) the improvement phase. In the definition phase, the requirements that constitute high quality data are defined regarding the different perspectives of data stakeholders, i.e. consumers, manufacturers, suppliers, or managers. Based on these requirements, metrics are developed and executed in the measurement phase. The analysis phase covers the examination of potential data quality problems identified through the metrics in the measurement phase. Possible alternative solutions have to be identified in order to resolve the data quality problems at its origin. Besides the simple correction of data values, the resolution of problems can also require the correction of business processes. Eventually, the best solution is executed during the improvement phase. Since business processes and thereby data manufacturing underlies changes, the TDQM lifecycle needs to be repeated over time.

In this paper, we focus on the definition and measurement phase of TDQM when applying DQM techniques to the Semantic Web. We define a set of data quality requirements through SPARQL queries that can be executed during the measurement phase and are useful for the subsequent analysis of the data quality problems and their causes. At the moment, we do not define key performance indicators (KPI) for the judgment of data and information quality, which can also be part of TDQM.

[2] http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/

3.2 Architecture

As a major goal of our approach, we aim at defining generalized data quality rules, i.e. metrics for the identification of data quality problems, solely with the use of technologies provided by the Semantic Web technology stack. Therefore, we use the SPARQL query language extended by elements of Jena's ARQ language[3] due to its high expressivity. Hence, our data quality rules are transparent to the user and applicable to knowledge bases throughout the Web of Data. Besides official W3C specifications, we use D2RQ[4] for wrapping relational data sources into the Resource Description Framework (RDF[5]). Moreover, with the use of Jena ARQ's SERVICE keyword[6], we are able to execute federated SPARQL queries over the Semantic Web and remotely retrieve linked data available from public SPARQL endpoints in real time.

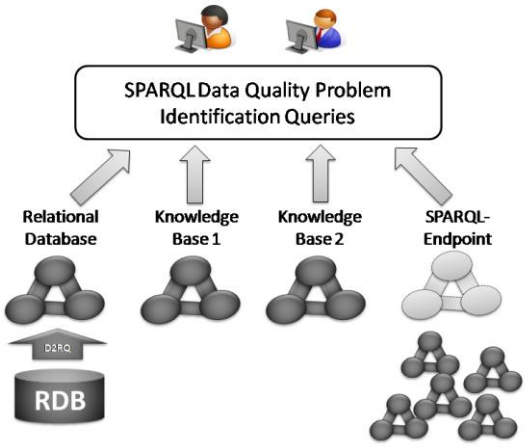

Fig. 1. Using SPARQL for data quality problem identification in the Web of Data

In the following, we present our generic SPARQL data quality problem identification queries for the identification of missing literals, illegal literals, and functional dependency violations in Semantic Web knowledge bases. Parameters that need to be substituted by real ontology elements before the execution of the query are put into angle brackets.

[3] http://jena.sourceforge.net/ARQ/
[4] http://www4.wiwiss.fu-berlin.de/bizer/d2rq/
[5] http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/
[6] http://jena.sourceforge.net/ARQ/service.html

3.3 Identification of Missing Literals

In section 2.1, we explained the different types of missing values which can occur in Semantic Web scenarios. In [11], we defined a query that identifies datatype properties that have missing literal values for a single, known datatype property. In this paper, we (1) extend this query by enabling the additional detection of missing (but required) datatype properties attached to instances of particular classes, and (2) define a query for the identification of literals that are missing on the basis of a functional dependency, e.g. datatype properties that are mandatory within certain literal value combinations.

Table 1. SPARQL queries for the identification of missing literals

Data Quality Problem	Generalized SPARQL Query

Missing literal or datatype property	`SELECT ?s`
	`WHERE{{`
	`  ?s a <class1> .`
	`  ?s <prop1> "" .}`
	`UNION{`
	`  ?s a <class1> .`
	`  NOT EXISTS {`
	`  ?s <prop1> ?value}}}`

Functionally dependent missing literal or datatype property	`SELECT ?s`
	`WHERE{{`
	`  ?s a <class1> .`
	`  ?s <prop1> <value1> .`
	`  NOT EXISTS{`
	`  ?s <prop2> ?value2 .`
	`  }`
	`  }UNION{`
	`  ?s <prop1> <value1> .`
	`  ?s <prop2> "" .`
	`  }}`

In case one, we select all instances `?s` of class `<class1>` that have an empty literal value for the datatype property`?prop1` and/or that do not have the datatype property `?prop1`. The second query returns all instances `?s` of class `<class1>` that have a datatype property `<prop1>` with the literal value `<value1>` and do not have the datatype property `<prop2>` at all or simply no literal value for it. Hence, the latter query only checks for missing literal values or datatype properties in instances with the value `<value1>` for datatype property `<prop1>`. Thus, it can be used to check literals that are only mandatory when another datatype property of the same instance obtains a certain literal value. E.g. the property `prop:state` may only be mandatory in cases where the property `prop:country` has the literal value `"USA"`. Note that the queries shown in here still require the substitution of the parameters in the angle brackets by real ontology classes, properties, and literal values.

3.4 Identification of Illegal Literal Values

In this section, we focus on the identification of incorrect values of a single datatype property without the use of any relationships from the knowledge base at hand. Without any semantic relationships to other literal values of an instance, the concise identification of false values is a difficult task that requires a precise definition of the allowed values for a datatype property. This can be done *approximately* by defining (1) the syntactical pattern for valid values or (2) a range of legal values. *Accurately*, illegal values can be identified by the definition of (3) all allowed values for a datatype property. The latter option can be very costly to implement since it may require manual effort to define or obtain an exhaustive list of legal values. In cases where there is a trusted reference, e.g. a knowledge base on the Semantic Web or a relational data source with a corresponding attribute that contains a nearly complete list of legal values, the manual effort can be minimized. Moreover, it can be promising to (4) define all illegal values for a certain datatype property. Depending on the ratio of legal vs. illegal literals, one option may be more efficient than the other. Often, it is unfeasible to define all illegal values without constraining the ability of the system to deal with innovation and dynamics in the domain; thus, option (4) should only be applied on the subset that is already retrieved through option (3) in order to identify strictly forbidden literal values and, therefore, reduce the amount of manual checks. Table 2 below shows the queries for options (3) and (4) that utilize literal values of a trusted reference indicating legal or illegal values respectively.

Note that the queries shown in here still require the substitution of the parameters in the angle brackets by real datatype properties. The OPTIONAL clause in here is meant to be used against the trusted knowledge base.

Table 2. Identification of illegal values with the use of a trusted knowledge base

Data Quality Problem	Generalized SPARQL Query

Listed values of <prop2> are legal	`SELECT ?s`
	`WHERE {`
	`?s <prop1> ?value .`
	`OPTIONAL {`
	`?s2 <prop2> ?value .`
	`} .`
	`FILTER (!bound(?s2)) .`
	`}`

Listed values of <prop2> are illegal	`SELECT ?s`
	`WHERE {`
	`?s <prop1> ?value .`
	`OPTIONAL {`
	`?s2 <prop2> ?value .`
	`} .`
	`FILTER bound(?s2) .`
	`}`

3.5 Identification of Functional Dependency Violations

In [11], we proposed an algorithm for the identification of functional dependency violations, which required the manual definition of legal combinations of literal values of two dependent datatype properties. Similar to false values it is also possible to check functional dependent literal values against a trusted knowledge base that already contains information on these dependencies. This latter option may save a lot of manual work, but requires the availability of trusted data sources that contain the required literal combinations. In Table 3, we provide two different algorithms for the identification of functional dependency violations. Each of the algorithms requires a trusted reference source. The first algorithm presented in Table 3 returns all instances `?s` that do not have an identical value combination in the trusted knowledge base for the literal value combinations of `<prop1> ?value1` and `<prop2> ?value2`. The semantics of datatype property `<prop1>` should thereby be equivalent to `<prop3>`, and likewise `<prop2>` to `<prop4>` in order to obtain the required literal value sets.

Table 3. SPARQL queries for the identification of functional dependency violations

Data Quality Problem	Generalized SPARQL Query

Functional dependency check tolerant to homonyms, n-ary literal value combinations, and missing datatype properties	`SELECT ?s`
	`WHERE {`
	`  ?s a <class1> .`
	`  ?s <prop1> ?value1 .`
	`  ?s <prop2> ?value2 .`
	`NOT EXISTS {`
	`  ?s2 a <class2> .`
	`  ?s2 <prop3> ?value1 .`
	`  ?s2 <prop4> ?value2 .`
	`}}`

Functional dependency check that enforces the existence of datatype properties	`SELECT ?s`
	`WHERE {{`
	`  ?s a <class1> .`
	`NOT EXISTS{`
	`  ?s <prop1> ?value1 .`
	`  ?s <prop2> ?value2 .}}`
	`UNION{`
	`  ?s a <class1> .`
	`  ?s <prop1> ?value1 .`
	`  ?s <prop2> ?value2 .`
	`NOT EXISTS{`
	`  ?s2 a <class2> .`
	`  ?s2 <prop3> ?value1 .`
	`  ?s2 <prop4> ?value2 .}}}`

Although the first query returns functional dependency violations that are not defined in the trusted reference, it does not return functional dependency violations in which the whole datatype property of the tested knowledge base is missing. Therefore, we have extended the first query by additionally defining the triples for the

datatype properties <prop1> and <prop2> to be mandatory. Thus, the second query returns all functional dependency violations that are not listed in the trusted knowledge base and all instances of the tested knowledge base that have missing datatype properties involved in the functional dependency (<prop1> and <prop>).

Both queries tolerate multiple assignments of the same literal value to more than one literal value of the dependent datatype property as the reference holds these combinations. As already stated in section 2.3, this must be tolerated due to the existence of homonymous values and n-ary relationships between literal values of different datatype properties. The drawback of this approach is that an incorrect instance will not be identified by the queries if the instance contains a legal combination that is still incorrect in the further context. For example, the combination of the datatype property `prop:city "Neustadt"` and the datatype property `prop:country "DE"` could be identified as correct, although the instance actually intends to represent the Austrian city `"Neustadt"`. Such detection errors can be minimized by integrating more than two datatype properties into the functional dependency check, thus enhancing the probability of the identification of illegal value combinations, e.g. the zip code of our knowledge base may clearly indicate that the country has to be "AT". Accordantly, the generalized SPARQL queries as proposed in Table 3 may be extended by additional dependent datatype properties and their literal values if the trusted reference provides those properties and literals. Note that the queries shown in Table 3 still require the substitution of the parameters in the angle brackets by real ontology classes and properties.

3.6 Options for Integrating Trusted Knowledge Bases into Quality Checks

Table 4. Checking the existence of city names of a local knowledge base in DBPedia

Federated SPARQL Query

```
PREFIX dbo:<http://dbpedia.org/ontology/>
SELECT *
WHERE {
?s1 stockdb:location_CITY ?city .
OPTIONAL{
SERVICE <http://dbpedia.org/sparql>{
?s2 a dbo:City .
?s2 rdfs:label ?city .
FILTER (lang(?city) = "en") .
}}
FILTER(!bound(?s2))
}
```

Assuming that our knowledge base is stored locally, we have at least two basic options how to integrate Semantic Web resources as trusted data sources in our data quality identification metrics, namely (1) by replicating the data into local data sources, or (2) by querying linked data remotely, e.g. by including SPARQL endpoints remotely by means of the SERVICE function for query federation from

Jena's ARQ language[7]. Moreover, with wrapping technologies, such as D2RQ[8], it is also possible to use non-Semantic-Web data as a reference for quality management purposes in the Semantic Web technology stack.

4 Evaluation

In the following, we evaluate our proposal. First, we analyze the data quality problem identification techniques presented in the previous section. The techniques for the identification of illegal values and functional dependency violations are evaluated by comparison of a local knowledge base against the publicly available data dumps from Geonames[9], a Semantic Web knowledge base for geographical data. The techniques for the identification of missing literal values and datatype properties have been sufficiently evaluated on a local knowledge base and are not considered in here. In a second phase, we evaluate the quality of Geonames itself.

4.1 Evaluation of Data Quality Techniques

We tested our queries against a local knowledge base that contains manually created address data. We thereby used a locally installed replication of Geonames as the trusted knowledge base for values and value combinations on geographical data. We checked whether the city names also occur in Geonames and are, therefore, considered legal, and whether all instances of our knowledge base have correct combinations of city and country names. We thereby used the city-country-combinations in Geonames as a trusted reference.

Table 5. Evaluation of functional dependency algorithms

No.	City	Country Property	Country	1st Algorithm	2nd Algorithm
1	Nantes2	Yes		X	X
2	Stavern	Yes	Norway		
3	Neubiberg	No			X
4	Neubiberg	Yes	USA	X	X
5	San Rafael	Yes	US	X	X
6	Melbourne	Yes	Australia		
7	Las Vegas	Yes	France	X	X

It must be noted that solely the existence of a certain city-country-combination was tested by our algorithm. It was not tested whether the combination is correct in further context of the data, although the used query is flexible enough to consider a third literal value or even more, if appropriate.

[7] http://jena.sourceforge.net/ARQ/service.html
[8] http://www4.wiwiss.fu-berlin.de/bizer/d2rq/
[9] http://download.geonames.org/export/dump/

Since Geonames only supplies the ISO-3166 2-letter country code to indicate the country, we had to adjust the queries slightly to convert the country codes into country names by using matches to literals of other datatype properties of Geonames that are connected to a full country name. Table 5 shows the results of the two queries for the identification of functional dependency violations from Table 3. The "X" indicates that the literal combination was detected as illegal by the algorithm. The data set for No. 3 had a missing datatype property for the country name. Thus, it was only detected by the second algorithm of table 3.

The test data contained the seven instances shown in Table 5. The queries were executed on an AMD QuadCore CPU with 4 GB RAM. Due to the small size of the test data, the execution time of the queries was not part of our evaluation. We also evaluated the first query from Table 2 using Geonames' full city labels ("`asciiname`") as a legal reference for evaluating the correctness of our city names. The algorithm identified that "Nantes2" is not known by Geonames. To enable the detection of incorrect literals, such as country names used as values for the property for city of the tested knowledge base, the trusted reference (Geonames) should be filtered to names of category "P" (city, village) when querying for illegal values.

4.2 Identification of Quality Problems in Geonames

To evaluate whether we can trust in the quality of a knowledge base, it is appropriate to apply quality problem identification metrics on the trusted knowledge base itself. Therefore, we applied algorithms for identifying a functional dependency violation and missing literals to the Geonames dataset itself. To evaluate the quality of the property "population", we defined an illegal combination between instances classified as populated places, such as country, state, region (fclass "A"), or city, village (fclass "P") that have a population of "0". Surprisingly, we observed that 93.3 % of all populated places in Geonames indicate a population of zero. If the value "0" means that the information about the accurate population is not available, then the value might be correct, but is still misleading to anyone who is not aware of this meaning. The other quality checks have shown that the properties `fclass`, `fcode`, `asciiname`, `country`, and `timezone` have only a few missing literals relative to the whole data set. Hence, for our quality checks it seems to be suitable to use the `asciinames` property from Geonames as a reference for legal location names.

Table 6. Quality metrics applied to Geonames (in literals)

Data Quality Problem	# of Occurences	Total #	Defect Ratio in Percent
Populated places (fclass P or A) without population (0)	2,626,026	2,814,701	93.30 %
Missing classification (fclass)	134,155	7,069,329	1.90 %
Missing classification (fcode)	135,253	7,069,329	1.91 %
Missing asciiname	604	7,069,329	0.01 %
Missing country	10,579	7,069,329	0.15 %
Missing timezone	29,312	7,069,329	0.41 %

4.3 Limitations

Although the algorithms of the latter two sections may identify false values and functional dependency violations very accurately without the investment of much manual effort, the use of Semantic Web resources as trusted references has one major weakness, which is that the reference data must be (1) complete and (2) reliable. If the reference dataset is incomplete, correct values in the data will be marked as incorrect. If the reference dataset contains illegal values, corresponding defects in the data to be analyzed will not be found. It is likely that there is at least a partial overlap between defects found in Semantic Web resources and relevant local datasets. For example, we have to expect the same common typos in DBPedia, derived from Wikipedia content, and local databases. One possible solution approach to this problem is the utilization of data quality problem identification techniques, e.g. as presented in [11], on the trusted source itself before starting the use of Semantic Web resources as a trusted reference for quality checks on local knowledge bases.

5 Related Work

Despite its importance, data quality has not yet received a lot of attention by Semantic Web researchers. A frequent misconception is that trust and data quality were the same. However, it is obvious that many of the data quality problems that we discuss in this paper are not directly related to the identity of the publisher of the data, nor to the lack of access control or authentication. For instance, there will be a lot of public datasets from the governments suffering from quality issues, despite the fact that the origin of the data and the integrity of the transformation and transportation is not in doubt. In the following, we summarize relevant related work.

Hartig and Zhao proposed a framework to assess the information quality of web data sources based on provenance information [15]. In addition, *Hartig* proposed an extension for the definition of trust values within Semantic Web data [16]. *Bizer and Cyganiak* described a framework to filter poor information in Web-based information systems according to user defined quality requirements [17]. Although these approaches are very promising, they do not provide much help to cure data quality problems in Semantic Web data sources or local data. Additionally, they are focused on the subjective assessment of data quality by users which may be occasionally not accurate enough or even wrong.

Lei, et. al. proposed an approach to identify data quality problems in semantic annotations. This approach utilizes Semantic Web data to identify incorrect classifications [18]. The proposed approach rather focuses on the quality of semantic annotations during its creation, but not on the quality of knowledge bases at instance level.

Other approaches use Semantic Web technology to identify and correct data quality problems in information systems. *Brüggemann and Grüning* have used ontologies to annotate incorrect data, e.g. redundant instances or incorrect attribute value combinations, to train detection algorithms for automated identification of data quality problems in cancer registries and data sources from the energy industry.

Furthermore, they proposed to use domain ontologies to populate commonly accepted data quality rules within the domain [19]. However, they focus on a small set of data quality problems of information systems and neither use the potential of data already published on the Semantic Web, nor attempt to identify quality problems within the Semantic Web. Moreover, none of the above approaches solely utilizes the expressivity and functionality of SPARQL as a widely established pillar of the Semantic Web technology stack.

The database and data quality research community has provided several proposals to identify, avoid, and cleanse data quality problems primarily for relational data sources [20]. However, those cannot be applied directly to data on a Web of Data, nor do they utilize Semantic Web datasets as references.

6 Conclusion and Outlook

The usefulness of knowledge representation strongly depends on the underlying data quality. Likewise, the success of the Semantic Web will depend on the quality of the published data. It is clear that the Semantic Web itself will never be a complete nor consistent knowledge representation, and that every consumer will have to apply filtering and cleansing techniques prior to using Semantic Web data. However, the ratio of noise and errors on one hand and the technical effort for filtering and cleansing the data for a given purpose will highly affect the value of Semantic Web data. Thus, it is very important to address data quality issues on a Web scale as part of the core Semantic Web technology stack. Unfortunately, there is currently a very limited amount of research on data quality on and for the Semantic Web.

In this paper, we have presented an approach to evaluate the quality of knowledge bases solely by using SPARQL queries. We provided generic queries for the identification of (1) missing literal values or datatype properties, (2) illegal literal values, and (3) functional dependency violations. Queries for the latter two data quality problems were built to make use of already available knowledge bases as a trusted reference. Including access of knowledge published in the Semantic Web in the data quality management process seems very promising for reducing the manual effort for data quality management. The major drawback of our approach is the uncertainty about the quality of the used knowledge bases available in the Semantic Web. Thus, we started to evaluate the quality of Geonames and have identified several data quality problems.

Our future work will address the extension of the evaluation of Geonames and other Semantic Web resources, such as DBPedia. Moreover, we plan to evaluate the quality of geographical data in DBPedia by using Geonames as a trusted knowledge base, and vice versa. We also plan to apply our approach for the quality assurance of master data of a local information system. To gain insight into the practical usefulness of Semantic Web resources for data quality management, we also plan to develop information quality scoring approaches built on top of our existing queries.

References

1. Redman, T. C.: Data quality for the information age. Artech House, Boston (1996)
2. Redman, T. C.: The impact of poor data quality on the typical enterprise. Communications of the ACM, 41, 79--82 (1998)
3. Brett, S.: World Wide Web Consortium (W3C), http://www.w3.org/2007/Talks/0130-sb-W3CTechSemWeb/layerCake-4.png, retrieved on Mar 08th (2010)
4. Wang, R. Y., Strong, D. M.: Beyond accuracy: what data quality means to data consumers. Journal of Management Information Systems, 12(4), 5--33 (1996)
5. Redman, T. C.: Data quality: the field guide. Digital Press, Boston (2001)
6. Rahm, E., Do, H.-H.: Data Cleaning: Problems and Current Approaches. IEEE Data Engineering Bulletin 23(4), 3--13 (2000)
7. Oliveira, P., Rodrigues, F., Henriques, P.R., and Galhardas, H.: A Taxonomy of Data Quality Problems, In: Proc. 2nd Int. Workshop on Data and Information Quality (in conjunction with CAiSE'05), Porto, Portugal (2005)
8. Oliveira, P., Rodrigues, F., Henriques, P. R.: A Formal Definition of Data Quality Problems. In: International Conference on Information Quality (2005)
9. Leser, U., and Naumann, F.: Informationsintegration: Architekturen und Methoden zur Integration verteilter und heterogener Datenquellen, dpunkt-Verlag, Heidelberg (2007)
10. Kashyap, V., and Sheth, A.P.: Semantic and Schematic Similarities Between Database Objects: A Context-Based Approach, Very Large Data Base Journal (5), 276--304 (1996)
11. Fürber, C., Hepp, M.: Using SPARQL and SPIN for Data Quality Management on the Semantic Web. 13th International Conference on Business Information Systems (BIS2010). Springer LNBIP (forthcoming), Berlin, Germany (2010)
12. Olson, J.: Data quality: the accuracy dimension. Morgan Kaufmann; Elsevier Science, Oxford (2003)
13. Wang, R.Y.: A product perspective on total data quality management. Commun. ACM 41 (1998) 58-65
14. Gruber, T.R.: A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition 5 (1993) 199-220
15. Hartig, O., Zhao, J.: Using Web Data Provenance for Quality Assessment. First International Workshop on the role of Semantic Web in Provenance Management (Co-located with the 8th International Semantic Web Conference, ISWC-2009), Washington D.C., USA. (2009)
16. Hartig, O.: Querying Trust in RDF Data with tSPARQL. In: Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Simperl, E. (eds.): 6th Annual European Semantic Web Conference (ESWC2009), Vol. 5554. Springer, Heidelberg (2009) 5-20
17. Bizer, C., Cyganiak, R.: Quality-driven information filtering using the WIQA policy framework. Web Semant. 7 (2009) 1-10
18. Lei, Y., Nikolov, A.: Detecting Quality Problems in Semantic Metadata without the Presence of a Gold Standard. EON, Vol. 329. CEUR-WS.org (2007) 51-60
19. Brüggemann, S., Grüning, F.: Using Ontologies Providing Domain Knowledge for Data Quality Management In: Pellegrini, T., Auer, S., Tochtermann, K., Schaffert, S. (eds.): Networked Knowledge - Networked Media Springer Berlin / Heidelberg (2009) 187-203
20. Batini, C., Scannapieco, M.: Data quality : concepts, methodologies and techniques. Springer, Berlin (2006)