Lessons Learned — The Case of CROCUS: Cluster-Based Ontology Data Cleansing

Didier Cherix[2], Ricardo Usbeck[1,2(✉)], Andreas Both[2], and Jens Lehmann[1]

[1] University of Leipzig, Leipzig, Germany
{usbeck,lehmann}@informatik.uni-leipzig.de
[2] R & D, Unister GmbH, Leipzig, Germany
{andreas.both,didier.cherix}@unister.de

Abstract. Over the past years, a vast number of datasets have been published based on Semantic Web standards, which provides an opportunity for creating novel industrial applications. However, industrial requirements on data quality are high while the time to market as well as the required costs for data preparation have to be kept low. Unfortunately, many Linked Data sources are error-prone which prevents their direct use in productive systems. Hence, (semi-)automatic quality assurance processes are needed as manual ontology repair procedures by domain experts are expensive and time consuming. In this article, we present CROCUS – a pipeline for cluster-based ontology data cleansing. Our system provides a semi-automatic approach for instance-level error detection in ontologies which is agnostic of the underlying Linked Data knowledge base and works at very low costs. CROCUS has been evaluated on two datasets. The experiments show that we are able to detect errors with high recall. Furthermore, we provide an exhaustive related work as well as a number of lessons learned.

1 Introduction

The Semantic Web movement including the Linked Open Data (LOD) cloud[1] represents a combustion point for commercial and free-to-use applications. The Linked Open Data cloud hosts over 300 publicly available knowledge bases with an extensive range of topics and DBpedia [1] as central and most important dataset. While providing a short time-to-market of large and structured datasets, Linked Data has yet not reached industrial requirements in terms of provenance, interlinking and especially data quality. In general, LOD knowledge bases comprise only few logical constraints or are not well modelled.

Industrial environments need to provide high quality data in a short amount of time. A solution might be a significant number of domain experts that are checking a given dataset and defining constraints, ensuring the demanded data quality. However, depending on the size of the given dataset the manual evaluation process by domain experts will be time consuming and expensive. Commonly, a dataset is integrated in iteration cycles repeatedly which leads to a

[1] http://lod-cloud.net/

V. Presutti et al. (Eds.): ESWC Satellite Events 2014, LNCS 8798, pp. 14–24, 2014.
DOI: 10.1007/978-3-319-11955-7_2

generally good data quality. However, new or updated instances might be error-prone. Hence, the data quality of the dataset might be contaminated after a re-import.

From this scenario, we derive the requirements for our data quality evaluation process. (1) Our aim is to find singular faults, i.e., unique instance errors, conflicting with large business relevant areas of a knowledge base. (2) The data evaluation process has to be efficient. Due to the size of LOD datasets, reasoning is infeasible due to performance constraints, but graph-based statistics and clustering methods can work efficiently. (3) This process has to be agnostic of the underlying knowledge base, i.e., it should be independent of the evaluated dataset.

Often, mature ontologies, grown over years, edited by a large amount of processes and people, created by a third party provide the basis for industrial applications (e.g., DBpedia). Aiming at short time-to-market, industry needs scalable algorithms to detect errors. Furthermore, the lack of costly domain experts requires non-experts or even layman to validate the data before influencing a productive system. Resulting knowledge bases may still contain errors, however, they offer a fair trade-off in an iterative production cycle.

In this article, we present CROCUS, a cluster-based ontology data cleansing framework. CROCUS can be configured to find several types of errors in a semi-automatic way, which are afterwards validated by non-expert users called quality raters. By applying CROCUS' methodology iteratively, resulting ontology data can be safely used in industrial environments.

On top of our previous work [2] our contributions are as follows: we present (1) an exhaustive related work and classify our approach according to three well-known surveys, (2) a pipeline for semi-automatic instance-level error detection that is (3) capable of evaluating large datasets. Moreover, it is (4) an approach agnostic to the analysed class of the instance as well as the Linked Data knowledge base. (5) we provide an evaluation on a synthetic and a real-world dataset. Finally, (6) we present a number of lessons learned according to error detection in real-world datasets.

2 Related Work

The research field of ontology data cleansing, especially instance data can be regarded threefold: (1) development of statistical metrics to discover anomalies, (2) manual, semi-automatic and full-automatic evaluation of data quality and (3) rule- or logic-based approaches to prevent outliers in application data.

In 2013, Zaveri et al. [10] evaluate the data quality of DBpedia. This manual approach introduces a taxonomy of quality dimensions: (i) accuracy, which concerns wrong triples, data type problems and implicit relations between attributes, (ii) relevance, indicating significance of extracted information, (iii) representational consistency, measuring numerical stability and (iv) interlinking, which looks for links to external resources. Moreover, the authors present a *manual*

Table 1. Table of founded papers for each in [8] defined Dimension on the basis of [9, Tables 8–9]. The blue dimensions are considered in this work.

Dimension		Procedure			
		Automatic	Semi - Automatic	Manual	Not Specified
Intrisic DQ	Believability			[3]	
	Objectivity			[3]	
	Reputation			[3]	
	Correctness		[4,5]	[3]	
Contextual DQ	Completeness	[6]	[5]	[3]	
	Added Value	[6]		[3]	
	Relevancy			[3]	
	Timeliness			[3]	
	Amount of data			[3]	
Representation	Interpretability			[3]	[7]
	Understandability			[3]	
	Consistency			[3]	[7]
	Conciseness			[3]	[7]
Accessibility	Availability			[3]	[7]
	Response time			[3]	
	Security			[3]	

error detection tool called *TripleCheckMate*[2] and a *semi-automatic* approach supported by the description logic learner (DL-Learner) [11,12], which generates a schema extension for preventing already identified errors. Those methods measured an error rate of 11.93 % in DBpedia which will be a starting point for our evaluation.

A *rule-based* framework is presented by Furber et al. [13] where the authors define 9 rules of data quality. Following, the authors define an error by the number of instances not following a specific rule normalized by the overall number of relevant instances. Afterwards, the framework is able to generate statistics on which rules have been applied to the data. Several *semi-automatic* processes, e.g., [4,5], have been developed to detect errors in instance data of ontologies. Bohm et al. [4] profiled LOD knowledge bases, i.e., *statistical* metadata is generated to discover outliers. Therefore, the authors clustered the ontology to ensure partitions contain only semantically correlated data and are able to detect outliers.

[2] http://github.com/AKSW/TripleCheckMate

Hogan et al. [5] only identified errors in RDF data without evaluating the data properties itself.

In 2013, Kontokostas et al. [14] present an *automatic* methodology to assess data quality via a SPARQL-endpoint[3]. The authors define 14 basic graph patterns (BGP) to detect diverse error types. Each pattern leads to the construction of several cases with meta variables bound to specific instances of resources and literals, e.g., constructing a SPARQL query testing that a person is born before the person dies. This approach is not able to work iteratively to refine its result and is thus not usable in circular development processes.

Bizer et al. [3] present a *manual* framework as well as a browser to filter Linked Data. The framework enables users to define rules which will be used to clean the RDF data. Those rules have to be created manually in a SPARQL-like syntax. In turn, the browser shows the processed data along with an explanation of the filtering.

Network measures like degree and centrality are used by Guer et al. [6] to quantify the quality of data. Furthermore, they present an *automatic* framework to evaluate the influence of each measure on the data quality. The authors proof that the presented measures are only capable of discovering a few quality-lacking triples.

Hogan et al. [7] compare the quality of several Linked Data datasets. Therefore, the authors extracted 14 rules from best practices and publications. Those rules are applied to each dataset and compared against the Page Rank of each data supplier. Thereafter, the Page Rank of a certain data supplier is correlated with the datasets quality. The authors suggest new guidelines to align the Linked Data quality with the users need for certain dataset properties.

A first classification of quality dimensions is presented by Wang et al. [8] with respect to their importance to the user. This study reveals a classification of data quality metrics in four categories, cf. Table 1. Recently, Zaveri et al. [9] present a systematic literature review on different methodologies for data quality assessment. The authors chose 21 articles, extracted 26 quality dimensions and categorized them according to [8]. The results shows which error types exist and whether they are repairable manually, semi-automatic or fully automatic. The presented measures were used to classify CROCUS.

To the best of our knowledge, our tool is the first tool tackling error accuracy (intrinsic data quality), completeness (contextual data quality) and consistency (data modelling) at once in a semi-automatic manner reaching high f1-measure on real-world data.

3 Method

First, we need a standardized extraction of target data to be agnostic of the underlying knowledge base. SPARQL [15] is a W3C standard to query instance data from Linked Data knowledge bases. The `DESCRIBE` query command is a way to retrieve descriptive data of certain instances. However, this query command

[3] http://www.w3.org/TR/rdf-sparql-query/

depends on the knowledge base vendor and its configuration. To circumvent knowledge base dependence, we use *Concise Bounded Descriptions* (CBD) [16]. Given a resource r and a certain description depth d the CBD works as follows: (1) extract all triples with r as subject and (2) resolve all blank nodes retrieved so far, i.e., for each blank node add every triple containing a blank node with the same identifier as a subject to the description. Finally, CBD repeats these steps d times. CBD configured with $d = 1$ retrieves only triples with r as subject although triples with r as object could contain useful information. Therefore, a rule is added to CBD, i.e., (3) extract all triples with r as object, which is called *Symmetric Concise Bounded Description* (SCDB) [16].

Second, CROCUS needs to calculate a numeric representation of an instance to facilitate further clustering steps. Metrics are split into three categories:

(1) The simplest metric counts each property (*count*). For example, this metric can be used if a person is expected to have only one telephone number.

(2) For each instance, the range of the resource at a certain property is counted (*range count*). In general, an undergraduate student should take undergraduate courses. If there is an undergraduate student taking courses with another type (e.g., graduate courses), this metric is able to detect it.

(3) The most general metric transforms each instance into a numeric vector and normalizes it (*numeric*). Since instances created by the SCDB consist of properties with multiple ranges, CROCUS defines the following metrics: (a) numeric properties are taken as is, (b) properties based on strings are converted to a metric by using string length although more sophisticated measures could be used (e.g., n-gram similarities) and (c) object properties are discarded for this metric.

As a third step, we apply the *density-based spatial clustering of applications with noise* (DBSCAN) algorithm [17] since it is an efficient algorithm and the order of instances has no influence on the clustering result. DBSCAN clusters instances based on the size of a cluster and the distance between those instances. Thus, DBSCAN has two parameters: ϵ, the distance between two instances, here calculated by the metrics above and $MinPts$, the minimum number of instances needed to form a cluster. If a cluster has less than $MinPts$ instances, they are regarded as outliers. We report the quality of CROCUS for different values of $MinPts$ in Sect. 4 (Fig. 1).

Finally, identified outliers are extracted and given to human quality judges. Based on the revised set of outliers, the algorithm can be adjusted and constraints can be added to the Linked Data knowledge base to prevent repeating discovered errors.

4 Evaluation

LUBM benchmark. First, we used the LUBM benchmark [18] to create a perfectly modelled dataset. This benchmark allows to generate arbitrary knowledge

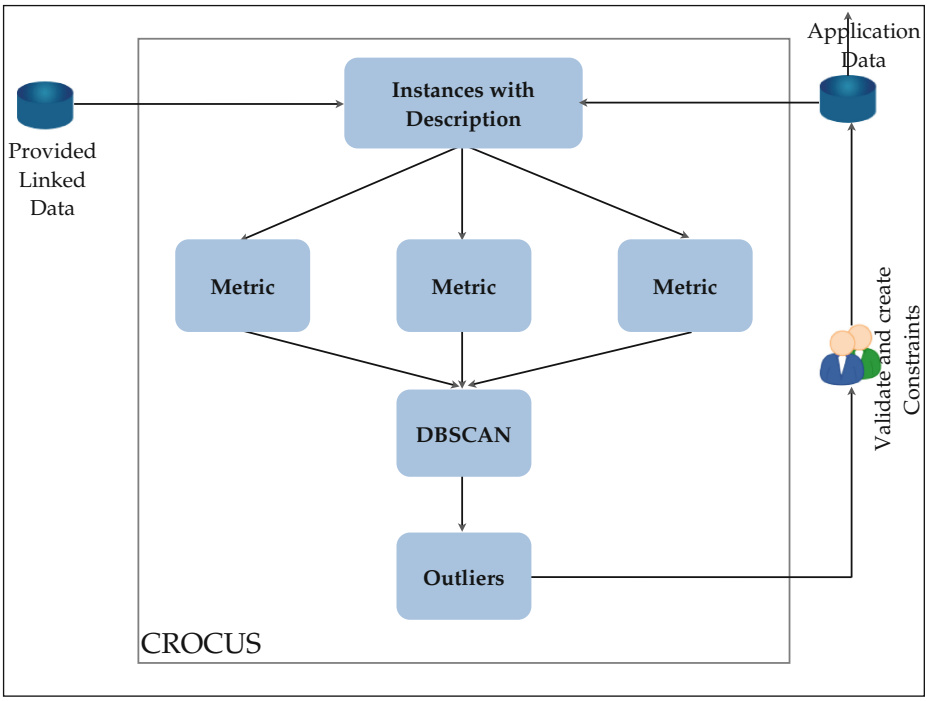

Fig. 1. Architecture of CROCUS.

bases themed as university ontology. Our dataset consists of exactly one university and can be downloaded from our project homepage[4].

The LUBM benchmark generates random but error free data. Thus, we add different errors and error types manually for evaluation purposes:

- *completeness of properties (count)* has been tested with CROCUS by adding a second phone number to 20 of 1874 graduate students in the dataset. The edited instances are denoted as I_{count}.
- *semantic correctness of properties (range count)* has been evaluated by adding for non-graduate students (`Course`) to 20 graduate students ($I_{rangecount}$).
- *numeric correctness of properties (numeric)* was injected by defining that a graduate student has to be younger than a certain age. To test this, 20 graduate students ($I_{numeric}$) age was replaced with a value bigger than the arbitrary maximum age of any other graduate.

For each set of instances holds: $|I_{count}| = |I_{rangecount}| = |I_{numeric}| = 20$ and additionally $|I_{count} \cap I_{rangecount} \cap I_{numeric}| = 3$. The second equation overcomes a biased evaluation and introduces some realistic noise into the dataset. One of those 3 instances is shown in the listing below:

[4] https://github.com/AKSW/CROCUS

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix ns2: <http://example.org/#> .
@prefix ns3: <http://www.Department6.University0.edu/> .

ns3:GraduateStudent75 a ns2:GraduateStudent ;
  ns2:name "GraduateStudent75" ;
  ns2:undergraduateDegreeFrom <http://www.University467.edu> ;
  ns2:emailAddress "GraduateStudent75@Department6.University0.edu" ;
  ns2:telephone "yyyy-yyyy-yyyy" , "xxx-xxx-xxxx" ;
  ns2:memberOf <http://www.Department6.University0.edu> ;
  ns2:age "63" ;
  ns2:takesCourse ns3:GraduateCourse21 , ns3:Course39 , ns3:
      GraduateCourse26 ;
  ns2:advisor ns3:AssociateProfessor8 .
```

Listing 1.1. Example of an instance with manually added errors (*in red*).

DBpedia - German universities benchmark. Second, we used a subset of the English DBpedia 3.8 to extract all German universities. The following SPARQL query (Listing 1.2) presents already the difficulty to find a complete list of universities using DBpedia.

```
SELECT DISTINCT ?instance
    WHERE {
        {   ?instance a dbo:University.
            ?instance dbo:country dbpedia:Germany.
            ?instance foaf:homepage ?h.
        } UNION {
            ?instance a dbo:University.
            ?instance dbp::country dbpedia:Germany.
            ?instance foaf:homepage ?h.
        } UNION {
            ?instance a dbo:University.
            ?instance dbp::country "Germany"@en.
            ?instance foaf:homepage ?h.
        }}
```

Listing 1.2. SPARQL query to extract all German universities.

After applying CROCUS to the 208 universities and validating detected instances manually, we found 39 incorrect instances. This list of incorrect instances, i.e., CBD of URIs, as well as the overall dataset can be found on our project homepage. For our evaluation, we used only properties existing in at least 50 % of the instances to reduce the exponential parameter space. Apart from an increased performance of CROCUS we did not find any effective drawbacks on our results.

Results. To evaluate the performance of CROCUS, we used each error type individually on the adjusted LUBM benchmark datasets as well as a combination of all error types on LUBM[5] and the real-world DBpedia subset.

Table 2 shows the f1-measure (F1), precision (P) and recall (R) for each error type. For some values of $MinPts$ it is infeasible to calculate cluster since DBSCAN generates only clusters but is unable to detect outlier. CROCUS is able to detect the outliers with a 1.00 f1-measure as soon as the correct size of $MinPts$ is found.

[5] The datasets can also be found on our project homepage.

Table 2. Results of the LUBM benchmark for all three error types.

	LUBM								
	count			*range count*			*numeric*		
MinPts	F1	P	R	F1	P	R	F1	P	R
2	—	—	—	—	—	—	—	—	—
4	—	—	—	0.49	1.00	0.33	—	—	—
8	—	—	—	0.67	1.00	0.5	—	—	—
10	0.52	1.00	0.35	1.00	1.00	1.00	—	—	—
20	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
30	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
50	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
100	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table 3 presents the results for the combination of all error types for the LUBM benchmark as well as for the German universities DBpedia subset. Combining different error types yielding a more realistic scenario influences the recall which results in a lower f1-measure than on each individual error type. Finding the optimal $MinPts$ can efficiently be done by iterating between $[2, \ldots, |I|]$. However, CROCUS achieves a high recall on the real-world data from DBpedia. Reaching a f1-measure of 0.84 for LUBM and 0.91 for DBpedia highlights CROCUS detection abilities.

Table 3. Evaluation of CROCUS against a synthetic and a real-world dataset using all metrics combined.

	LUBM			DBpedia		
MinPts	F1	P	R	F1	P	R
2	0.12	1.00	0.09	0.04	0.25	0.02
4	0.58	1.00	0.41	0.04	0.25	0.02
8	0.84	1.00	0.72	0.04	0.25	0.02
10	0.84	1.00	0.72	0.01	0.25	0.01
20	0.84	1.00	0.72	0.17	0.44	0.10
30	0.84	1.00	0.72	0.91	0.86	0.97
50	0.84	1.00	0.72	0.85	0.80	0.97
100	0.84	1.00	0.72	0.82	0.72	0.97

Table 4. Different error types discovered by quality raters using the German universities DBpedia subset.

Property	Errors
`dbp:staff`, `dbp:established`, `dbp:internationalStudents`	Values are typed as `xsd:string` although they contain numeric types like integer or double.
`dbo:country`, `dbp:country`	`dbp:country` "Germany"@en collides with `dbo:Germany`

In general, CROCUS generated many candidates which were then manually validated by human quality raters, who discovered a variety of errors. Table 4 lists the identified reasons of errors from the German universities DBpedia subset detected as outlier. As mentioned before, some universities do not have a

`dbo:country` property. However, we found a new type of error. Some literals are of type `xsd:string` although they represent a numeric value. Lists of wrong instances can also be found on our project homepage.

Overall, CROCUS has been shown to be able to detect outliers in synthetic and real-world data and is able to work with different knowledge bases.

5 Lessons Learned

By applying CROCUS on a real-world ontology a set of erroneous candidates is provided to the quality raters. Based on those candidates quality raters and domain experts are able to define constraints to avoid a specific type of failure.

Obviously, there are some failures which are too complex for a single constraint. For instance, an object property with more than one authorized class as range needs another rule for each class, e.g., a property `locatedIn` with the possible classes `Continent, Country, AdminDivision`. Any object having this property should only have one instance of `Country` linked to it. The same holds for an instance of type `Continent`. However, it is possible to have more than one `AdminDivision` since each district or state is an `AdminDivision`.

One possible solution is to create distinct classes for `State` and `District`. An even better way is to introduce new subproperties, i.e., `locatedInDistrict`. Thus, it is possible to define a rule that an object can only have one district. This does not exclude objects with more than one `locatedIn` associated to an instance of `AdminDivision` as its range.

6 Conclusion

We presented CROCUS, a novel architecture for cluster-based, iterative ontology data cleansing, agnostic of the underlying knowledge base. With this approach we aim at the iterative integration of data into a productive environment which is a typical task of industrial software life cycles.

The experiments showed the applicability of our approach on a synthetic and, more importantly, a real-world Linked Data set. Finally, CROCUS has already been successfully used on a travel domain-specific productive environment comprising more than 630.000 instances (the dataset cannot be published due to its license).

In the future, we aim at a more extensive evaluation on domain specific knowledge bases. Furthermore, CROCUS will be extended towards a pipeline comprising a change management, an open API and semantic versioning of the underlying data. Additionally, a guided constraint derivation for laymen will be added.

Acknowledgments. This work has been partly supported by the ESF and the Free State of Saxony and by grants from the European Union's 7th Framework Programme provided for the project GeoKnow (GA no. 318159). Sincere thanks to Christiane Lemke

References

1. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. Seman. Web J. (2014)
2. Cherix, D., Usbeck, R., Both, A., Lehmann, J.: Crocus: Cluster-based ontology data cleansing. In: Proceedings of the 2nd International Workshop on Semantic Web Enterprise Adoption and Best Practice (2014)
3. Bizer, C., Cyganiak, R.: Quality-driven information filtering using the wiqa policy framework. Web Semant. Sci. Serv. Agents World Wide Web **7**(1), 1–10 (2009)
4. Böhm, C., Naumann, F., Abedjan, Z., Fenz, D., Grutze, T., Hefenbrock, D., Pohl, M., Sonnabend, D.: Profiling linked open data with ProLOD. In: IEEE 26th International Conference on Data Engineering Workshops ICDEW 2010, pp. 175–178 (2010)
5. Hogan, A., Harth, A., Passant, A., Decker, S., Polleres, A.: Weaving the pedantic web. In: Bizer, C., Heath, T., Berners-Lee, T., Hausenblas, M. (eds.) LDOW. CEUR Workshop Proceedings, vol. 628. CEUR-WS.org (2010)
6. Guéret, C., Groth, P., Stadler, C., Lehmann, J.: Assessing linked data mappings using network measures. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 87–102. Springer, Heidelberg (2012)
7. Hogan, A., Umbrich, J., Harth, A., Cyganiak, R., Polleres, A., Decker, S.: An empirical survey of linked data conformance. Web Semant. Sci. Serv. Agents World Wide Web **14**, 14 (2012)
8. Wang, R.Y., Strong, D.M.: Beyond accuracy. what data quality means to data consumers. J. Manage. Inf. Syst. **12**(4), 5–33 (1996)
9. Zaveri, A., Rula, A., Maurino, A., Pietrobon, R., Lehmann, J., Auer, S., Hitzler, P.: Quality assessment methodologies for linked open data. Seman. Web J. (2013) (Submitted)
10. Zaveri, A., Kontokostas, D., Sherif, M.A., Bühmann, L., Morsey, M., Auer, S., Lehmann, J.: User-driven quality evaluation of dbpedia. In: Sabou, M., Blomqvist, E., Noia, T.D., Sack, H., Pellegrini, T. (eds.) I-SEMANTICS, pp. 97–104. ACM (2013)
11. Lehmann, J.: DL-learner: learning concepts in description logics. J. Mach. Learn. Res. **10**, 2639–2642 (2009)
12. Bühmann, L., Lehmann, J.: Pattern based knowledge base enrichment. In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) ISWC 2013, Part I. LNCS, vol. 8218, pp. 33–48. Springer, Heidelberg (2013)
13. Fürber, C., Hepp, M.: Swiqa - a semantic web information quality assessment framework. In: Tuunainen, V.K., Rossi, M., Nandhakumar, J. (eds.) ECIS (2011)
14. Kontokostas, D., Westphal, P., Auer, S., Hellmann, S., Lehmann, J., Cornelissen, R., Zaveri, A.J.: Test-driven evaluation of linked data quality. In: Proceedings of the 23rd International Conference on World Wide Web (2014, to appear)
15. Quilitz, B., Leser, U.: Querying distributed RDF data sources with SPARQL. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 524–538. Springer, Heidelberg (2008)
16. Stickler, P.: Cbd-concise bounded description. W3C Member Submission 3 (2005)

17. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD, vol. 96, pp. 226–231 (1996)
18. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. Web Semant. Sci. Serv. Agents World Wide Web **3**(2–3), 158–182 (2005)