Collaborative Project

GeoKnow - Making the Web an Exploratory for Geospatial Knowledge

Project Number: 318159 **Start Date of Project:** 2012/12/01 **Duration:** 36 months

Deliverable 3.5.2

Final report on spatial data quality assessment

Dissemination Level	Public
Due Date of Deliverable	Month 32, 30/07/2015
Actual Submission Date	Month 35, 12/10/2015
Work Package	WP3, Spatial knowledge aggregation, fusing and quality assessment
Task	T3.5
Type	Report
Approval Status	Final
Version	1.0
Number of Pages	19
Filename	D3.5.2_Final_Report_On_Spatial_Data_Quality _Assessment.pdf

Abstract: This deliverable reports on the newly gathered experiences in data quality aspects since the initial report D3.5.1. This includes new and adapted metrics. All of the metrics are implemented into *GeoKnow Quality Evaluator (GQE)*, a software tool to measure the quality of Linked Geospatial datasets. The quality of different datasets is assessed using GQE and results are compared.

Project funded by the European Commission within the Seventh Framework Programme (2007 - 2013)

History

Version	Date	Reason	Revised by
0.0	07/07/2015	First draft created	Muhammad Saleem
0.1	15/07/2015	Draft revised	Mohamed Ahmed Sherif
0.2	08/08/2015	Unister contributions	Matthias Wauer
0.2	09/08/2015	First version created	Muhammad Saleem
0.3	08/09/2015	Internal review	Amrapali Zaveri
0.3	11/10/2015	Internal review	Giorgos Giannopoulos
0.4	12/10/2015	Final version submitted	Muhammad Saleem

Author List

Organization	Name	Contact Information
INFAI	Muhammad Saleem	saleem@informatik.uni-leipzig.de
INFAI	Axel-Cyrille Ngonga Ngomo	ngonga@informatik.uni-leipzig.de
INFAI	Mohamed Ahmad Sherif	sherif@informatik.uni-leipzig.de
INFAI	Jens Lehmann	lehmann@informatik.uni-leipzig.de
Unister	Didier Cherix	didier.cherix@unister.de
Unister	Matthias Wauer	matthias.wauer@unister.de

Executive Summary

Large Geospatial data sets are specially prone to errors due to multiple autonomous data providers that use different assumptions about structure and semantics of data and no uniform quality standards. This deliverable reports on the newly gathered data quality metrics since the initial report D3.5.1. These include both new and adapted metrics pertaining to Geospatial data quality. We present *GeoKnow Quality Evaluator (GQE)*, an automatic software tool to measure the quality of Linked Geospatial datasets. Using GQE, we have measured the data quality of three well-known Linked Geospatial datasets – Linked Geo Data, NUTS, Geo Linked Data – and presented the results. Our results show that these datasets achieve different performance for the proposed data quality metrics implemented in GQE.

Contents

List of Figures

List of Tables

1 Introduction

The Linking Open Data (LOD) cloud hosts 9960[1] publicly available knowledge bases with many of them containing geospatial annotations. Producing a single high quality integrated Geospatial data set from different RDF representations is one of the main challenges in the GeoKnow project. In general, LOD knowledge bases comprise only few logical constraints or are not well modelled. Thus, merging geospatial features from different data sets is not straightforward. With GeoKnow aiming to address an extensive range of users including customers in an industrial environment, the quality of data sets resulting from fusing other data sets becomes a crucial factor for the acceptance and distribution of the project's results.

In recent years, the concern for Geospatial data quality has increased due to a number of factors including [3]: (1) increased data production by the private sector and non-government agencies, which are not governed by uniform quality standards (production of data by national agencies has long been required to conform to national accuracy standards), and (2) increased reliance on secondary data sources, due to the growth of the Internet, data translators, and data transfer standards, making poor quality data ever easier to get.

This deliverable extends the initial data quality report D3.5.1 by adding new and adapted metrics (defined in next section). All of the metrics are implemented into GQE, a fully automated java tool to measure the quality of Linked Geospatial datasets. It requires the SPARQL endpoint (hosting the data set) URL as input and produces an RDF DataCube[2] file (containing specified metric results) as output. The resulting file is then visualized via CubeViz[3], a visualizer for RDF DataCube files.

The rest of this deliverable is organized as follows: we first explain the metrics implemented into GQE. We then provide the implementation details. In particualr, we discuss the RDF DataCube generation and running the GQE from command line. We finally describe the experimental setup and present the results.

[1] http://stats.lod2.eu/
[2] http://www.w3.org/TR/vocab-data-cube/
[3] http://cubeviz.aksw.org/

2 Metrics

In this section, we describe the new and adapted metrics implemented into GQE to measure the quality of spatial datasets.

2.1 Properties Per Class

In this metric, we calculate how many distinct predicates exist for instances of a class. This metric just needs a dataset as input. The output is an integer per class representing the distinct predicates that are used in statements where the subject is an instance of the class.

2.2 Instances Per Class

This metric just calculates how many distinct instances exist for each class. This metric can be used to weigh the importance of a class in a dataset.

2.3 Average Surface Area Per Class

This metric calculates the average surface contained in polygons for each class. This metric is important to relativise the number of instances of some class. A class representing continents has only a few instances but the covered surface is much bigger than that of a class representing a city.

2.4 Number of Intersecting Classes per Instances

This metric calculates how many instances have more types that are only in this class. This is important to represent how specific the current class is. In a very specific class, outliers are more significant than in a general class.

2.5 Average Number of Points Per Class

This metric represents the average of points per class. For each instance of the current class, the metric computes how many points are linked from this instance. This metric is important to differentiate between classes representing multi-point objects and those representing one point.

2.6 Average number of Polygons Per Class

This metric reports the average number of polygons within a class of a geospatial dataset.

2.7 Average Distance Between Point Sets which Represent the Same Resource

In this metric we compute the average distance between polygons which represent the same resource in two linked datasets. This metric takes as input a source dataset S, a target dataset T and a set of point set distance functions D. We depend on the set of point set distance functions first introduced in a previous deliverable[4]. Geospatial resources are commonly described by means of vector geometry[5]. Each vector description can be

[4]Deliverable 3.4.1 "Metrics for Linked Geospatial Information"
[5]Most commonly encoded in the WKT format, see http://www.opengeospatial.org/standards/sfa

modelled as a set of points. We will write $R = (p_1, \ldots, p_n)$ to denote that the vector description of the resource R comprises the points $p_1, \ldots, p_n$. A point p_i on the surface of the planet is fully described by two values: its latitude $lat(p_i) = \varphi_i$ and its longitude $lon(p_i) = \lambda_i$. We will denote points p_i as pairs (φ_i, λ_i). Then, the distance between two points p_1 and p_2 can be computed by using the *orthodromic distance*

$$\delta(p_1, p_2) = \mathbf{R} \cos^{-1}\left(\sin(\varphi_1)\sin(\varphi_2) + \cos(\varphi_1)\cos(\varphi_2)\cos(\lambda_2 - \lambda_1)\right),$$

where $\mathbf{R} = 6371km$ is the planet's radius[6]. Computing the distance between sets of points is yet a more difficult endeavour. Over the last years, several measures have been developed to achieve this task. Most of these approaches regard vector descriptions as ordered sets of points. The input for the distances consists of two point sets $s = (s_1, \ldots, s_n)$ and $t = (t_1, \ldots, t_m)$, where n resp. m stands for the number of distinct points in the description of s resp. t. W.l.o.g, we assume $n \geq m$. Here, we used nine metrics:

1. Mean Metric
$$d_{mean}(s,t) = \delta\left(\frac{\sum_{s_i \in S} s_i}{n}, \frac{\sum_{t_j \in T} t_j}{m}\right). \tag{1}$$

2. Max Metric
$$d_{max}(s,t) = \max_{s_i \in s, t_j \in t} \delta(s_i, t_j). \tag{2}$$

3. Min Metric
$$d_{min}(s,t) = \min_{s_i \in s, t_j \in t} \delta(s_i, t_j). \tag{3}$$

4. Average Metric
$$d_{average}(s,t) = \frac{1}{nm} \sum_{s_i \in S, t_j \in t} \delta(s_i, t_j). \tag{4}$$

5. Sum of Minimums Metric (SOM)
$$d_{som}(s,t) = \frac{1}{2}\Big(\sum_{s_i \in s} \min_{t_j \in t} \delta(s_i, t_j) + \sum_{t_i \in t} \min_{s_j \in s} \delta(t_i, s_j)\Big). \tag{5}$$

6. Surjection Metric
$$d_{surjection}(s,t) = \min_{\eta} \sum_{(e1,e2) \in \eta} \delta(e_1, e_2), \tag{6}$$
where η is the surjection from the larger of the point sets S and T to the smaller.

7. Fair Surjection Metric (FS)
$$d_{fs}(s,t) = \min_{\eta'} \sum_{(e1,e2) \in \eta'} \delta(e_1, e_2), \tag{7}$$
where η' is the evenly mapped surjection from the larger of the sets s and t to the smaller.

8. Link Metric
$$d_{link}(s,t) = \min_{R} \sum_{(s_i,t_j) \in R} \delta(s_i, t_j), \tag{8}$$
where minimum is computed from all relations R, where R is a linking between s and t satisfying the previous two conditions.

[6] We assume the planet to be a perfect sphere.

9. Hausdorff Metric

$$d_{hausdorff}(s,t) = \max_{s_i \in S} \left\{ \min_{t_j \in T} \left\{ \delta(s_i, t_j) \right\} \right\}. \quad (9)$$

Given a source dataset S, we start the metric computation procedure by querying S for the set of all classes C within S. For each source class c in C, we select the set of source class instances s which (1) have a vector geometry and (2) linked with a target dataset instance t in T and (3) t also has a vector geometry representation. Afterwords, each of the aforementioned distance functions is applied against each pair of linked source and target instances, formally $d(s,t)$. Finally, for each d in D, we compute $averagePS(d)$ as the average all the computed point sets metrics for each class, formally

$$averagePS(d) = \sum_{\forall s \in c} \frac{d(s,t)}{|s|}. \quad (10)$$

The result of the metric is a table containing (1) *metric name* d, (2) *class name* c, (3) *time stamp* and (4) *the average distance* $averagePS(d)$.

2.8 Class Coverage

This metric was introduced in [2] and determines how well the instance data conform to rdf:class (class for short), i.e., how well a specific class is covered by the different instances of that class. The coverage of a class C demented by $Coverage(C)$ is defined as follow:

Definition 2.1 (Class Coverage)**.** For a dataset D, let $P(C)$ denote the set of distinct properties having class C and $I(C)$ denote the set of distinct instances having class C. Let $I(p,C)$ denote the number of distinct instances having predicate p and class C. Then, the coverage of the class $CV(C)$ is

$$CV(C) = \frac{\sum_{\forall p \in P(C)} I(p,C)}{|P(C)| \times |I(C)|}$$

2.9 Weighted Class Coverage

Definition 2.1 considers the structuredness of a dataset with respect to a single class. Obviously, a dataset D has instances from multiple classes, with each instance belonging to at least one of these classes (if multiple instantiation is supported). It is possible that dataset D might have a high structuredness for a class C, say CV(C) = 0.8, and a low structuredness for another class C', say CV(C') = 0.15. But then, what is the structuredness of the whole dataset with respect to our class system (set of all classes)? Duan et al. [2] proposed a mechanism to compute this, by considering the weighted sum of the coverage $CV(C)$ of individual classes. In particular, for each class C, the weighted coverage is defined below.

Definition 2.2 (Weighted Class Coverage)**.** By using Definition 2.1, the weighted coverage for a class C denoted by $WT(CV(C))$ is calculated using the following formula:

$$WT(CV(C)) = \frac{|P(C)|+|I(C)|}{\sum_{\forall C' \in D} |P(C')|+|I(C')|}$$

2.10 Dataset Structuredness

By using Definitions 2.1, 2.2, we are now ready to compute the structuredness, hereafter termed as coherence, of a whole dataset D.

Definition 2.3 (Dataset Structuredness)**.** The overall structuredness or coherence of a dataset D denoted by $CH(D)$ is define as

$$CH(D)) = \sum_{\forall C \in D} CV(C) \times WT(CV(C))$$

3 Implementation

In this section, we describe the the GQE DataCube generation process followed by the step-by-step instructions to run the GQE.

3.1 DataCubes Generation

Each of the aforementioned performance metric is implemented into GQE which produces an RDF Datacube file as output for each of the metrics. The resulting RDF Datacube file is then visualized via CubeViz. In this section, we explain a sample Datacube representation of the output for our last metric, i.e., the dataset structuredness. The format remains the same for all other metrics. The dataset structuredness DataCube has *Dataset, TimeStamp, Structuredness* as observations which show the structuredness values of the dataset along with the time stamp. A sample resulting DataCube is shown in Table 1. The corresponding query to print this table is shown in Listing 1.

Table 1: Dataset structuredness sample results.

Dataset	TimeStamp	Structuredness
Linked Geo Data	Tue May 12 01:19:51 CEST 2015	0.308755930062984
NUTS	Sun May 10 18:49:04 CEST 2015	0.802331367134043
GeoLinked Data	Mon May 11 16:04:59 CEST 2015	0.992417184558037

```
1 prefix gk-dim: <http://www.geoknow.eu/properties/>
2 prefix sdmx-measure: <http://purl.org/linked-data/sdmx/2009/measure#>
3 prefix qb: <http://purl.org/linked-data/cube#>
4 SELECT ?Dataset ?TimeStamp ?Structuredness
5 WHERE
6 {
7 ?obsrv qb:dataSet <http://www.geoknow.eu/dataset/ds1> ;
8     gk-dim:Dataset ?Dataset
9     gk-dim:TimeStamp ?TimeStamp ;
10    sdmx-measure:Structuredness ?Structuredness ;
11   a qb:Observation .
12 }
```

Listing 1: SPARQL query to print Structuredness of dataset

According to the RDF DataCube vocabulary[7], a well-formed representation of the statistical data comprise:

[7] http://www.w3.org/TR/vocab-data-cube/

Datasets Definition: Listing 2 shows the dataset definitions for the structuredness metric. Each dataset has a data structure definition and is explained next.

```
<http://www.geoknow.eu/dataset/ds3>  a qb:DataSet ;
          dcterms:publisher "AKSW, GeoKnow" ;
          rdfs:label "Dataset Structuredness" ;
          rdfs:comment "Dataset Structuredness" ;
          qb:structure <http://www.geoknow.eu/data-cube/dsd3> ;
          dcterms:date "Mon May 11 16:09:38 CEST 2015".
```

Listing 2: DataCube generation: Dataset definitions

Data Structure Definition: Listing 3 shows the data structure definitions of the structuredness metric. This Datacube contains of three components. Each dataset component can be a dimension, property or measure and is explained next.

```
<http://www.geoknow.eu/data-cube/dsd3> a qb:DataStructureDefinition ;
                        rdfs:label   "A Data Structure Definition"@en ;
                        rdfs:comment  "A Data Structure Definition for
                            DataCube3" ;
                        qb:component <http://www.geoknow.eu/data-cube/dsd3/c1>,
                        <http://www.geoknow.eu/data-cube/dsd3/c2>,
                        <http://www.geoknow.eu/data-cube/dsd3/c3> .
```

Listing 3: DataCube generation: Data structure definitions

Component Specification: Listing 4 and Listing 5 shows the specification of each of the dataset component defined in Listing 3. Our DataCube contains 2 dimensions, i.e., *Dataset, TimeStamp* and one measure, i.e., *Structuredness*.

```
<http://www.geoknow.eu/data-cube/dsd3/c1> a qb:ComponentSpecification ;
                      rdfs:label   "Component Specification of Dataset" ;
                      qb:dimension gk-dim:Dataset .
<http://www.geoknow.eu/data-cube/dsd3/c2> a qb:ComponentSpecification ;
                       rdfs:label   "Component Specification of Time Stamp" ;
                      qb:dimension gk-dim:TimeStamp .
<http://www.geoknow.eu/data-cube/dsd3/c3> a qb:ComponentSpecification ;
                      rdfs:label   "Component Specification of Structuredness"
                         ;
                      qb:measure sdmx-measure:Structuredness .
```

Listing 4: DataCube generation: Component specifications

```
gk-dim:TimeStamp a qb:DimensionProperty ;
                 rdfs:label   "Time Stamp"@en .
gk-dim:Dataset a qb:DimensionProperty ;
                 rdfs:label   "Dataset name"@en .
sdmx-measure:Structuredness a qb:DimensionProperty ;
                 rdfs:label   "Dataset Structuredness"@en .
```

Listing 5: DataCube generation: Dimensions Units and Measures

Writing Observations: The final step is to write the observations. Listing 6 shows a sample observation written for Geo Liked Data.

```
<http://www.geoknow.eu/data-cube/dsd3/obs1> qb:dataSet <http://www.geoknow.eu
    /dataset/ds2> ;
    gk-dim:Dataset "GeoLinkedData" ;
    gk-dim:TimeStamp "Mon May 11 16:04:59 CEST 2015" ;
    sdmx-measure:Structuredness 0.992417184558037 ;
    a qb:Observation .
```

Listing 6: DataCube generation: Writing observations

3.2 Running GQE

GQE is publicly available and can be checkout from project home page https://github.com/GeoKnow/GeoQuality. We also provide a command line jar utility at project home page with the usage information along with a sample input is given in Listing 7. Basically, the jar requires the data set SPARQL endpoint URL (argument -e), the required GQE quality metric number (argument -m), and the output file location (argument -o) for the resulting DataCube file. For average point set we additional need to specify the class (argument -c) and predicate (argument -p) representing the point.

```
usage: java -jar GQE.jar -e <endpoint_url> -m <metric_number> -o <file> [-c
    <class_name>] [-p <predicate>]
-e    endpoint <endpoint_url> SPARQL endpoint URL
-o --output <file>   Output file
-c --class  <class_name>   Class name required for metric number X
-p --predicate <predicate> Predicate for point, required for metrix Y
-m    metrics  <metric_number>   GQE metric number, where:
    1     Properties Per Class
    2     Instances Per Class
    3   Surface Area Per Class
    4     Number of Intersecting Classes Instances
    5     Average Number of Points Per Class
    6     Average number of Polygons Per Class
    7     Average Distance Between Point Sets which Represent the Same
        Resource
    8     Coverage, weigted Coverage and Structurdness

For Example: java -jar GQE.jar -e http://linkedgeodata.org/sparql -m 1 -
    avgPointsSet
```

Listing 7: Usage for GQE jar utility

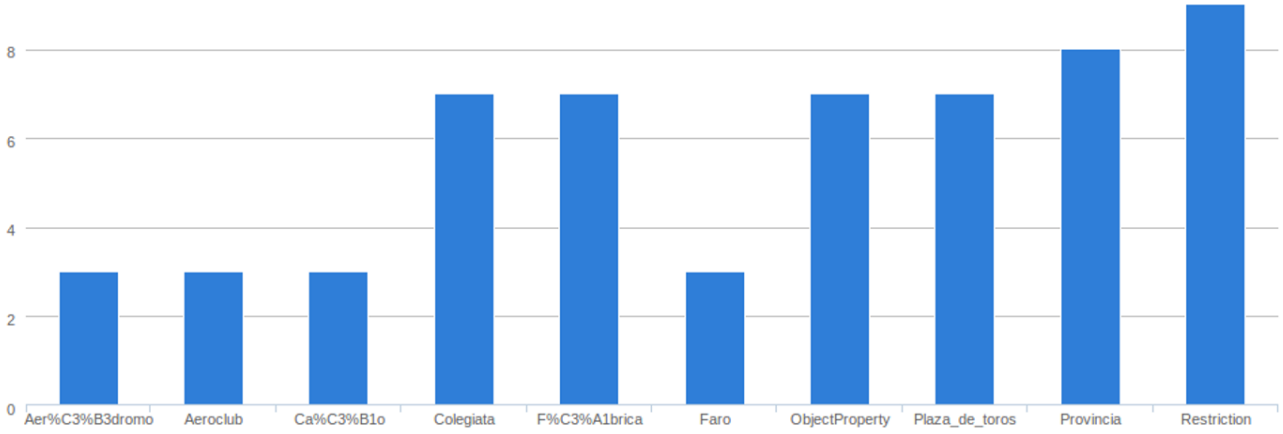

Figure 1: Metric 1 results: CubeViz Visualization of the number of properties per class of LGD dataset

4 Evaluation

In this section we first describe the experimental setup followed by the experimental results for each of the GQE data quality metric.

4.1 Experimental Setup

We selected three publicly available datasets of different sizes for our experiments. The first dataset, *Nuts*, contains a detailed description of 1,461 specific European regions.[8] The second dataset, *GeolinkedData*, available from the end point `http://geo.linkeddata.es/sparql`.[9] Finally, the third dataset, *LinkedGeoData*, available at `http://linkedgeodata.org/Datasets`.[10].

In addition to that, we evaluated the approach on three private datasets of the GeoKnow E-Commerce use case (WP6). A hotel reviews dataset extracted using Sparqlify contains geospatial coordinates of hotels. A regions dataset contains point geometries for the regions of such hotels, but these datasets are not entirely interlinked. Finally, a polygons dataset contains complex geometries of different types, e.g., countries, lakes and rivers, extracted using TripleGeo.

4.2 Results and Discussion

Figure 1 and Figure 2 show the number of properties and the number of instances, respectively, for the selected classes of GeoLinkedData. These information are simple to collect but important to understand the overall structure of a dataset. In both figures we can see a variation of values across the classes.

Figure 3 shows the average surface for the selected class. Some instances don't have a surface. The figure shows that only instances from the classes Polygon and MultiPolygon have a surface in the NUTS dataset. Some other classes like NUTSRegion have only an indirect surface: they are related to a polygon or multipolygon.

Some instances have more than one type. Figure 4 shows how many different classes from the selected class have a type relation to the instances of the selected class. The classes ObjectProperty and TransitiveProperty have a value of 1. An instance of ObjectProperty can only be a TransitiveProperty too. An instance

[8] More details about the *Nuts* dataset can be found at `http://datahub.io/dataset/linked-nuts`
[9] More details about the *GeolinkedData* dataset can be found at `http://datahub.io/dataset/geolinkeddata`
[10] More details about the *GeolinkedData* dataset can be found at `http://datahub.io/dataset/linkedgeodata`

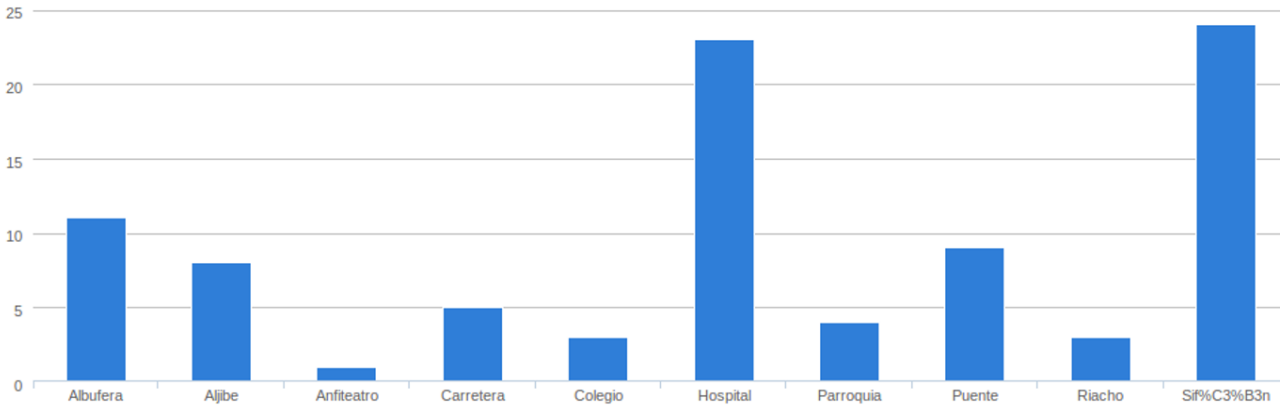

Figure 2: Metric 2 results: CubeViz Visualization of the number of instances per class of LGD dataset

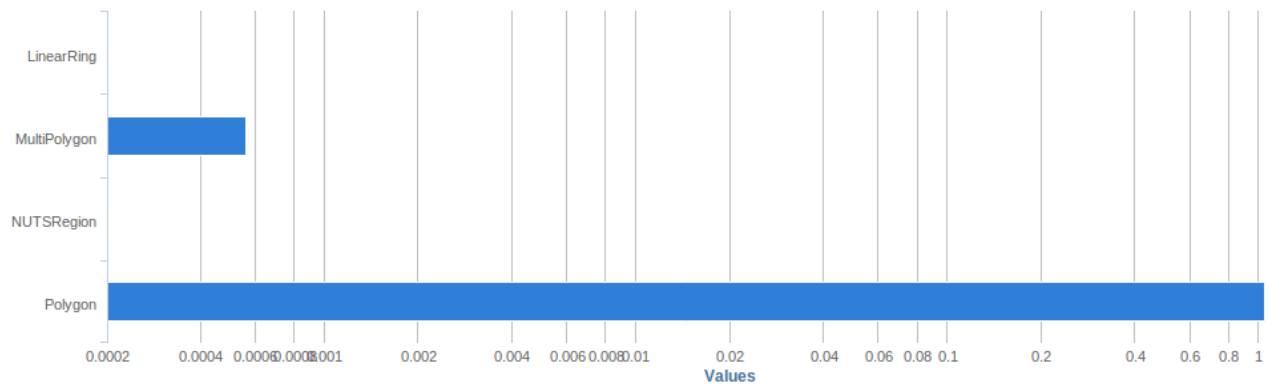

Figure 3: Metric 3 results: CubeViz Visualization of the average surface area per class of the NUTS dataset

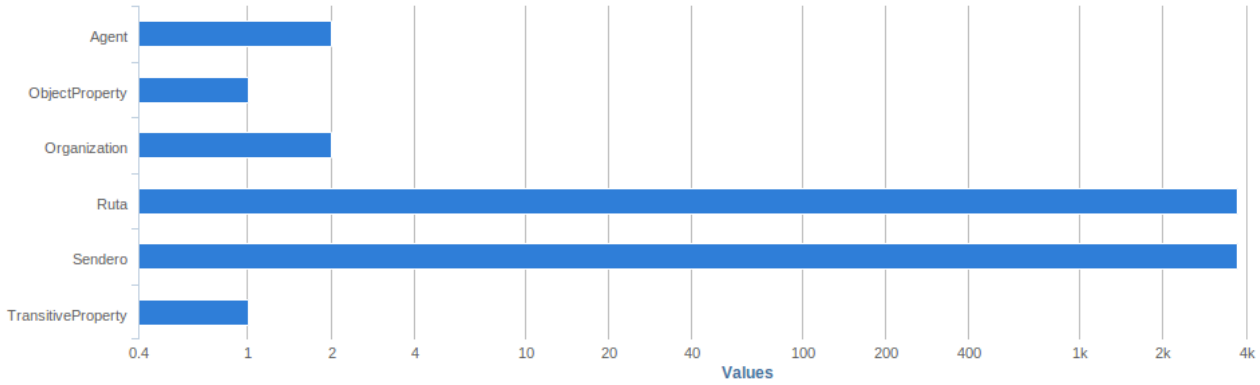

Figure 4: Metric 4 results: CubeViz Visualization of the number of intersecting classes Instances of the GLD dataset

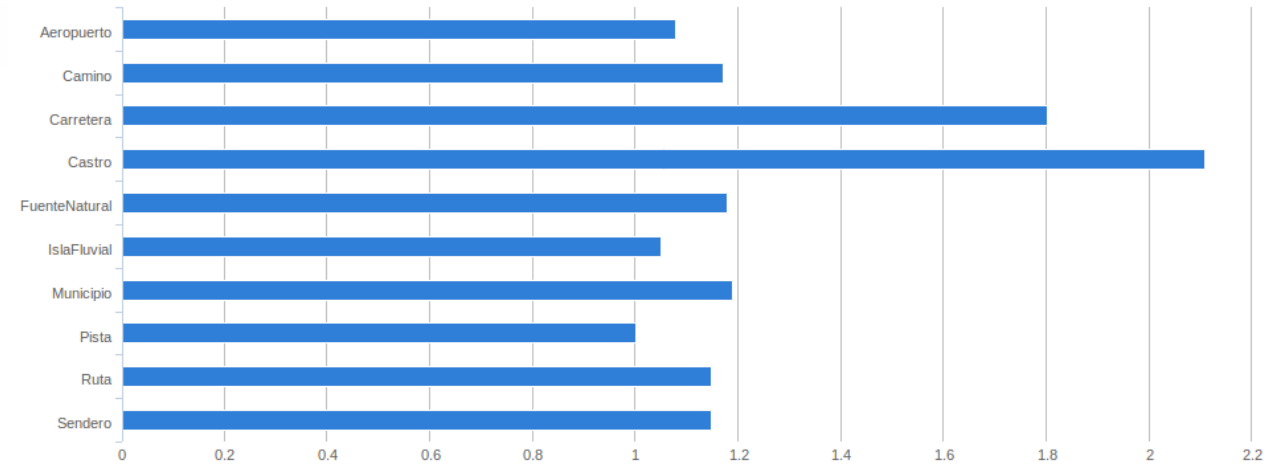

Figure 5: Metric 5 results: CubeViz Visualization of the average number of points per class of the GLD dataset

of TransitiveProperty can be an ObjectProperty or a DatatypeProperty. Some values of the intersecting classes metric can be used to verify if a dataset follows the OWL definitions.

Figure 5 and Figure 6 show the average number of points and polygons, respectively, for a selected class. The values of both figures have an important variation. The values show expected semantics in the dataset: an instance of polygon shouldn't have any polygon in it, au contraire a multipolygon should have more than one polygon related to itself.

Figure 7 shows the results of applying the aforementioned point set distances function in Section 2.7 for computing the average distances between resources. to generate this figure we used *GeoLinkedData* as the source dataset and *DBpedia* as the target dataset. The selected class from the *GeoLinkedData* was `http://geo.linkeddata.es/ontology/Provincia`. Average point set distances based on *surjection*, *fair surjection* and *link* metrics achieve higher average distances (about 1300°). While average point set distances based on *average*, *minimum*, *maximum*, *mean* and *hausdorf* metrics achieves less average distances (about 700°).

Figure 8 shows the CubeViz Visualization of the coverage metric for the selected classes of Linked Geo Data. We can clearly see a large variation in the coverage values of the the different classes. For example, Geometry class is well-covered (coverage value = 0.53) while Landuse class is not well-covered (coverage value = 0.0033).

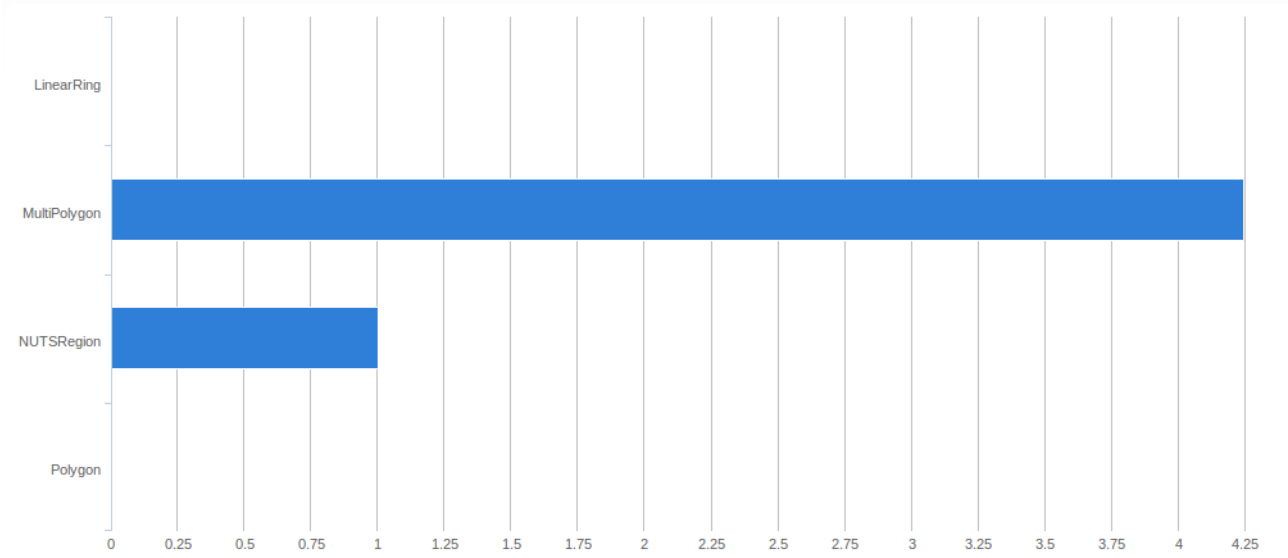

Figure 6: Metric 6 results: CubeViz Visualization of the average number of polygons per class of NUTS dataset

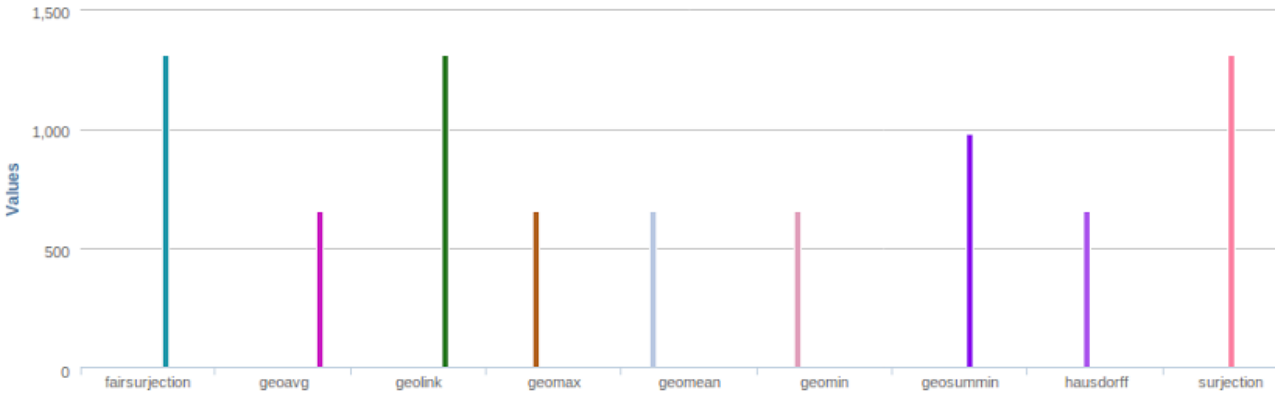

Figure 7: Metric 7 results: CubeViz Visualization of the average point set distances of the GLD dataset

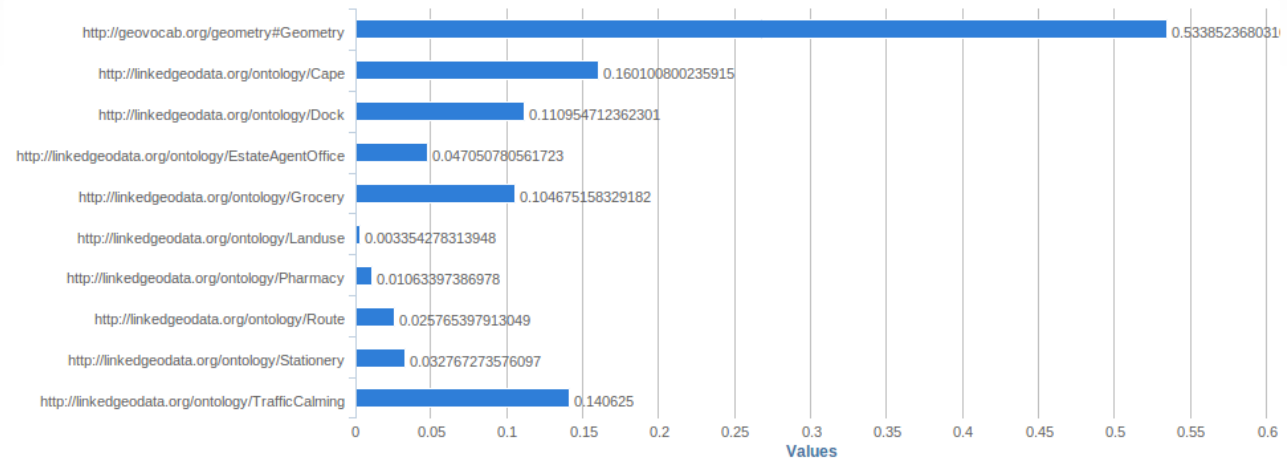

Figure 8: Metric 8 results: CubeViz Visualization of the coverage of Linked Geo Data selected classes

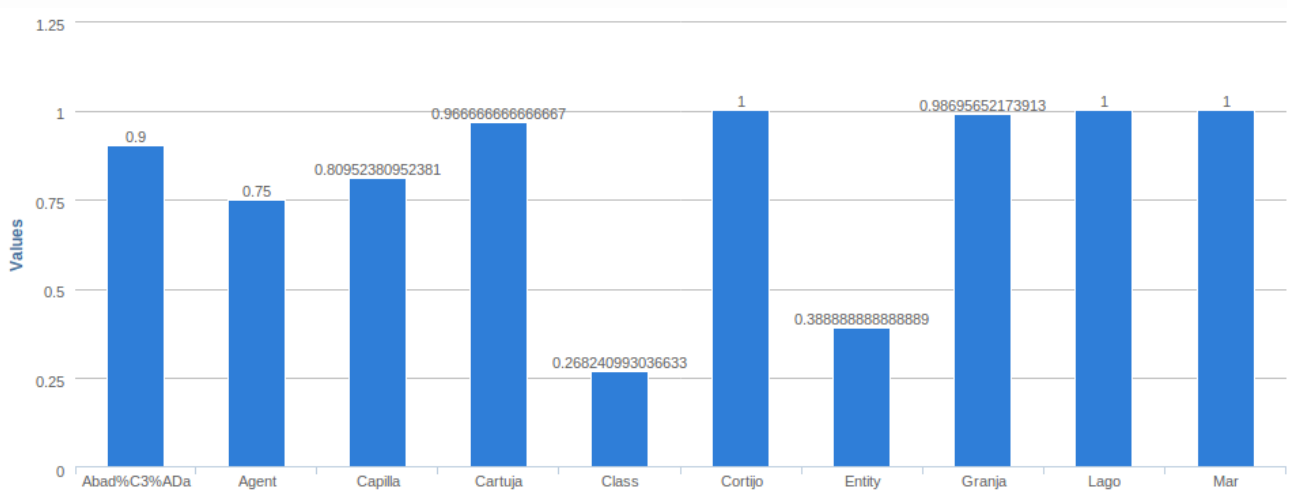

Figure 9: Metric 9 results: CubeViz Visualization of the weighted coverage of Geo Linked Data selected classes

Figure 9 shows the weighted coverage results for the Geo Linked Data sets selected classes. Same like coverage results, the weighted coverage values also varies ranging from a full weighted coverage (i.e., weighted coverage value =1) to low weighted coverage value, e.g., weighted coverage value of only 0.268. Note that both coverage and weighted coverage values directly affect the overall dataset coherence or structuredness, as explained in the next paragraph.

Figures 8 and 9 measure the coverage of the different classes within the same dataset. Figure 10 compares the overall structuredness or coherence of the selected datasets. Recall that structuredness of a dataset is calculated from the coverage and weighted coverage of the all the classes used in a dataset. Our results for this metric show that Geo Linked Data set is highly structured and Linked Geo Data is low structured, while NUTS is reasonably well structured. Note that the dataset structuredness value directly affects the overall query execution runtime over the specified dataset [2], [4], [1].

4.3 Use Case Specific Experiments

In addition to the metrics defined in Section 2, Unister tested the extensibility of the approach by implementing a few custom use-case-specific metrics. For example, we examined the correctness of materialized geospatial

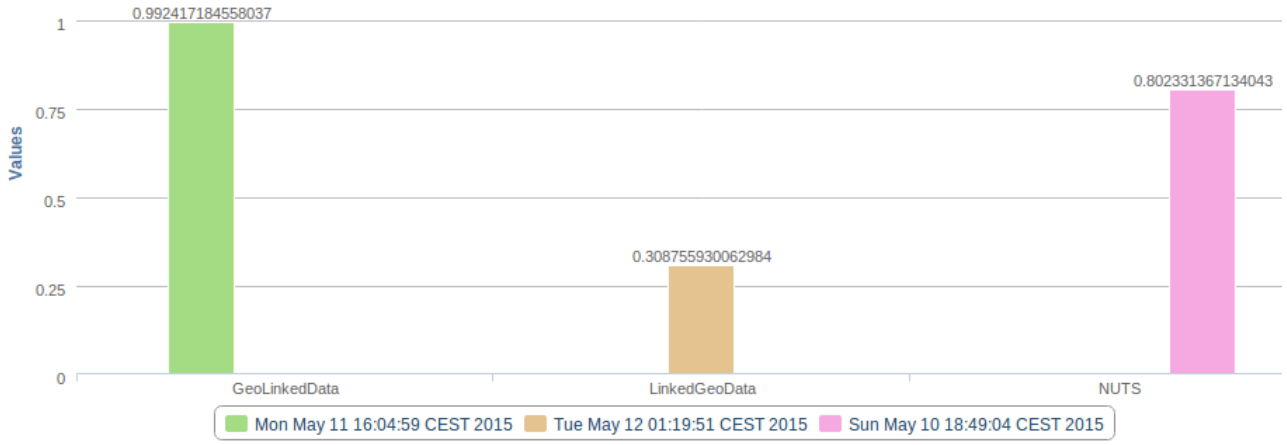

Figure 10: Metric 10 results: CubeViz Visualization of the structuredness of the selected datasets

properties, such as a "located in" relationship, with regards to different target classes. By visualising histograms of such metrics, we were able to identify outliers by selecting hotels located in, e.g., more than one country. In addition to that, with a previous interlinking of the three datasets we were able to identify potential data quality issues, e.g., if a hotel resource stated that it is located in a certain region, but the geocoordinate of the hotel is not within the region's interlinked geometry. In these cases, we encountered either interlinking issues or erroneous data of certain resources, such as invalid geocoordinates. The datasets are private, hence no visualisations or metrics details are included here. Still, we were able to validate that the approach can be extended easily in order to analyse additional quality metrics.

5 Conclusion and Future Work

In this deliverable we presented the final report on Geospatial Data Quality assessment. We presented a report on the newly gathered data quality metrics since the initial report D3.5.1. These included both new and adapted metrics pertaining to Geospatial data quality. We have implemented all of the presented Geospatial data quality metrics into GQE, an automatic software tool to measure the quality of Linked Geospatial datasets. Using GQE, we have measured the data quality of three well-known Linked Geospatial datasets – Linked Geo Data, NUTS, Geo Linked Data – and presented the results. Our results showed that these datasets achieve different performance for the proposed data quality metrics implemented in GQE. In the future, we will add other data quality metrics such as completeness, accuracy etc. into GQE.

References

[1] Güne Aluç, Olaf Hartig, M Tamer Özsu, and Khuzaima Daudjee. Diversified stress testing of rdf data management systems. In *The Semantic Web–ISWC 2014*, pages 197–212. Springer, 2014.

[2] Songyun Duan, Anastasios Kementsietsidis, Kavitha Srinivas, and Octavian Udrea. Apples and oranges: a comparison of rdf benchmarks and real rdf datasets. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 145–156. ACM, 2011.

[3] Edward. Spatial data quality and transportation applications. In *5th FIG Regional Conference*. 2006.

[4] Muhammad Saleem, Qaiser Mehmood, and Axel-Cyrille Ngonga Ngomo. Feasible: A featured-based sparql benchmark generation framework. In *International Semantic Web Conference (ISWC)*. LCNS, 2015.