Validating statistical index data represented in RDF using SPARQL queries

Jose Emilio Labra Gayo
WESO Research Group
University of Oviedo
Spain labra@uniovi.es

Jose M. Álvarez Rodríguez
South East European Research Center
Greece
jmalvarez@seerc.org

Abstract

In this position paper we describe an approach to validate statistical data representing index computations in RDF using SPARQL queries.

1 Introduction

Publishing statistical data is a promising domain where linked data approaches con offer a number of advantages.

TODO: a paragraph about linked data and RDF

The SPARQL [5] query language has been a successful technology to increase the adoption RDF. The current SPARQL 1.1 [4] has added new expressivity levels.

TODO: Talk about index data in general

The Web Index project (`http://thewebindex.org`) of 2012 offered a data portal[1] whose data was obtained by transforming the raw data and the computation values from Excel sheets to RDF[3]. In the new version of that data portal, we are planning to automate the validation and even the computation of the index data from the raw data.

TODO: Talk a little bit about the web index project, link to our publication

TODO: Add an overview of our approach…

We are currently implementing the new version of the Web Index data portal using this approach. At this moment, we have a running example and a validator implemented in Scala. The source code and the example are available at `https://github.com/weso/computex`.

Along the paper we will use Turtle and SPARQL notation and assume that the namespaces have been declared using the most common prefixes found in `http://prefix.cc`.

[1] `http://data.webfoundation.org`

2 Example data and Index computation process

TODO: A small screen capture

3 Data Model and Computex Ontology

TODO: Talk about the Computex ontology of statistical computations

3.1 Computex Ontology

The *Computex*[2] vocabulary defines terms related to the computation of statistical index data. It is an specialization of the RDF Data Cube vocabulary [2] and is compatible with it.

List of terms:

- **Indicator**
- **Concept**
- **Observation**. An observation. It contains at least values for the properties: cex:value, cex:indicator and cex:concept
- **Computation** we have declared the main computation types that we have needed for the WebIndex project, which have been summarized in table 1. That list of computation types is not exhaustive and can be further extended in the future.
- **WeightSchema** a weight schema for a list of indicators.

4 Validation approach

The validation approach was inspired by the integrity constraint specification proposed by the RDF Data Cube vocabulary which employs a set of SPARQL ASK queries to check the integrity of RDF Data Cube data. Although SPARQL ASK queries provide a good means to check the integrity, in practice their boolean nature does not offer too much help when a file does not validate. To solve that issue, we defined CONSTRUCT queries which, in case of error, construct an error message and a list of error parameters that can help to spot the problematic data.

We have also transformed the ASK queries defined in the RDF Data Cube vocabulary to CONSTRUCT queries. For example, SPARQL construct query to validate the RDF Data Cube integrity constraint 4 (IC-4) is:

```
CONSTRUCT {
 [ a cex:Error ; cex:errorParam [cex:name ”dim”; cex:value ?dim ] ;
   cex:msg ”Every Dimension must have a declared range” . ]
```

[2] http://purl.org/weso/computex

Table 1: Some types of statistical computations

Computation	Description	Properties
Mean	Mean of a set of observations	cex:observation
Increment	Increment an observation by a given amount	cex:observation amount
Raw	No computation. Raw value obtained from external source.	
Copy	A copy of another observation	cex:observation
Z-score	A normalization of an observation using the values from a Slice.	cex:observation cex:slice
Ranking	Position in the ranking of a slice of observations.	cex:observation cex:slice
AverageGrowth(N)	Expected average growth of N observations	cex:observations a collection of observations
WeightedMean	Weighted mean of an observation	cex:observation cex:slice cex:weightSchema

```
} WHERE {
  ?dim a qb:DimensionProperty .
  FILTER NOT EXISTS { ?dim rdfs:range [] }
}
```

In order to make our error messages compatible with EARL [1], we have defined cex:Error as a subclass of earl:TestResult and declared it to have the value earl:failed for the property earl:outcome.

We have also created our own set of SPARQL Construct queries to validate the *Computex* vocabulary terms, specially the computation of index data.

For example, the following query validates that every observation has at most one value.

```
CONSTRUCT {
 [ a cex:Error ; cex:errorParam # ... omitted
     cex:msg ”Observation has two different values” . ]
} WHERE {
 ?obs a qb:Observation .
 ?obs cex:value ?value1 .
 ?obs cex:value ?value2 .
 FILTER ( ?value1 != ?value2 )
}
```

Using this approach, we have been able to develop more expressive validations.

For example, we were able to validate that an observation was obtained as the mean of other observations.

```
CONSTRUCT {
 [ a cex:Error ; cex:errorParam # ...omitted
   cex:msg ”Mean value does not match” ] .
} WHERE {
  ?obs a qb:Observation ;
        cex:computation ?comp ;
        cex:value ?val .
  ?comp a cex:Mean .
  { SELECT (AVG(?value) as ?mean) ?comp WHERE {
      ?comp cex:observation ?obs1 .
          ?obs1 cex:value ?value ;
  } GROUP BY ?comp }
 FILTER( abs(?mean − ?val) > 0.0001)
}
```

5 Expressivity limits of SPARQL queries

Validating statistical computations using SPARQL queries offered a good exercise to check SPARQL expressivity. Although we were able to express most of the computation types. Some of them had to employ functions that are not part of the SPARQL 1.1 standard or had to be defined in a limited way. In this section we review the main difficulties.

- The Z-score of a value x_i is defined as $\frac{x-\bar{x}}{\sigma}$ where $\bar{x}$ is the mean and $\sigma = \sqrt{\frac{\sum_{i=1}^{N}(\bar{x}-x_i)^2}{N-1}}$ is the standard deviation. To validate that computation using SPARQL queries, it is necessary to employ the sqrt function. This function is not available in SPARQL 1.1 although some implementations like Jena ARQ[3] provide it.

- In order to validate the ranking of an observation (in which position it appears in a Slice), we used the GROUP_CONCAT SPARQL 1.1 function to group the ordered list of observations. However, SPARQL does not offer a function to calculate the position of a substring in a string[4], so in order to search the concept to rank, we had to divide the length of the substring before the concept's name by the length of the concept's name. This solution only works when all the concept's name have the same length.

- Given a list of values $x_1, x_2 \ldots x_n$ the expected value x_{n+1} can be extrapolated using the forward average growth formula: $x_n \times \frac{\frac{x_n}{x_{n-1}}+\ldots+\frac{x_2}{x_1}}{n-1}$.

[3] `http://jena.apache.org/documentation/query/library-function.html`

[4] This function is called strpos in PHP or indexOf in Java

If we represent the list of observations as an RDF collection, we were unable to define a SPARQL query that can express that formula for any length n. Instead, we were able to express that formula for a fixed length using SPARQL 1.1 property path expressions.

```
CONSTRUCT {
 [ a cex:Error ; cex:errorParam # ...omitted
   cex:msg ”Average growth value does not match” ] .
} WHERE {
  ?obs cex:computation [ a cex:AverageGrowth3 ;
                                   cex:observations ?ls ;] ;
   cex:value ?val .
 ?ls rdf:first ?x1 .
 ?ls rdf:rest/rdf:first ?x2 .
 ?ls rdf:rest/rdf:rest/rdf:first ?x3 .
 ?x1 cex:value ?v1 . ?x2 cex:value ?v2 . ?x3 cex:value ?v3 .
 BIND ((?v1 ∗ ((?v1 / ?v2) + (?v2 / ?v3)) / 2) as ?avgGrowth)
 FILTER (abs(?avgGrowth − ?val) > 0.001)
}
```

6 Conclusions and Future Work

Using SPARQL queries to validate and compute index data seems a promising use case for linked data applications. Although we have succesfully employed this approach to validate most of the statistical computations, we have found some limitations in current SPARQL 1.1 expressivity.

Our future plans are to automate the declarative computation of index data from the raw observations and to check the performance with real data like the Web Index data. We are also considering the feasibility of this approach for online calculation of index scores.

References

[1] Evaluation and report language EARL 1.0 schema. http://www.w3.org/TR/EARL10-Schema/, 2011. W3C Working Draft.

[2] The RDF data cube vocabulary. http://www.w3.org/TR/vocab-data-cube/, 2013. W3c Candidate Recommendation.

[3] J. M. Alvarez Rodríguez, J. Clement, J. E. Labra Gayo, H. Farhan, and P. Ordoñez. *Cases on Open-Linked Data and Semantic Web Applications*, chapter Publishing Statistical Data following the Linked Open Data Principles: The Web Index Project., pages 199–226. IGI Global, 2013. doi:10.4018/978-1-4666-2827-4.ch011.

[4] S. Harris and A. Seaborne. SPARQL 1.1 query language. http://www.w3.org/TR/sparql11-query/, 2013.

[5] E. Prud'hommeaux and A. Seaborne. SPARQL query language for RDF. http://www.w3.org/TR/rdf-sparql-query/, 2008.