Open Data - Knowledge Graph Lab

Prat Sicart, Joan
UPC Barcelona Tech
joan.prat.sicart@est.fib.upc.edu

Perez Gregori, Vicent
UPC Barcelona Tech
vicent.perez@est.fib.upc.edu

Wednesday 3rd April, 2019

This document represents the deliverable of the second assignment of the lab practices of the subject of Open Data corresponding to the Spring semester of the MIRI master degree during the course 2018/19.

A Exploring Dbpedia

```
rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs:http://www.w3.org/2000/01/rdf-schema#
owl: http://www.w3.org/2002/07/owl#
```

1. Find the class representing an Actor in the dataset (using filters).

```
SELECT ?x
WHERE { ?x rdf:type owl:Class.
        FILTER regex(?x, "Actor")
}
```

2. Find the super class for the class Actor.

```
SELECT ?x
WHERE { <http://dbpedia.org/ontology/Actor> rdfs:subClassOf ?x.}
```

3. Find all the actors in the dataset.

```
SELECT ?x  WHERE { ?x rdf:type <http://dbpedia.org/ontology/Actor>.}

SELECT ?actor
WHERE { {?actor rdf:type <http://dbpedia.org/ontology/Actor>}
        UNION
        {?actor rdf:type <http://dbpedia.org/ontology/AdultActor>}
        UNION
        {?actor rdf:type <http://dbpedia.org/ontology/VoiceActor>}}
```

Results depend on what we consider as an actor. We found that there are three possibilities for being an actor. However, if we only consider people belonging to the type "Actor" as one, the correct answer would be provided by the first query.

4. Get different classes that are defined as range of the properties that have the class Actor defined as their domain.

```
SELECT DISTINCT(?class)
WHERE { ?x rdfs:domain <http://dbpedia.org/ontology/Actor>.
        ?x rdfs:range ?class.
        ?class rdf:type owl:Class.}
```

5. Find the super property of the goldenRaspberryAward property.

```
SELECT ?property
WHERE {
    ?x rdfs:subPropertyOf ?property.
    FILTER regex(?x, "goldenRaspberryAward")}
```

6. Return all the properties that have the class Actor as either their range or domain.

```
SELECT ?x
WHERE {{?x rdfs:domain <http://dbpedia.org/ontology/Actor> .}
UNION {?x rdfs:domain <http://dbpedia.org/ontology/Actor> .}}
```

7. Return all persons that are not actors In order to ease the interpretations of our explanations here you can find depicted the three super steps that make the algorithm before retrieving the highest number.

```
SELECT ?person
WHERE{ ?person rdf:type <http://dbpedia.org/ontology/Person>.
MINUS {?person rdf:type <http://dbpedia.org/ontology/Actor>}}
GROUP BY ?person
```

For simplicity, we only considered actors people related with the class Actor. However, including the two other options we specified before, it would consist in increasing the MINUS statement to include both options.

B Analytical queries on top of QBAirbase

1. List the country, station type, latitude, and longitude details of each station. Note: Limit the query to 25 results, and extract only the string values of the required object and not the whole IRIs

```
PREFIX schema: <http://qweb.cs.aau.dk/airbase/schema/>
PREFIX property: <http://qweb.cs.aau.dk/airbase/property/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?countryname, str(?style) as ?station_type, ?long, ?lat WHERE {
     SELECT Distinct(?station), ?countryname, ?style, ?long, ?lat WHERE {
     ?obs schema:station ?station .
     ?station schema:inCountry ?country .
     ?station property:longitudeDegree ?long .
     ?station property:latitudeDegree ?lat .
     ?station property:type ?style.
     bind(strafter(str(?country),str(<http://qweb.cs.aau.dk/airbase/data/country/>))
        as ?country2)
     bind(strbefore(str(?country2),str(</>)) as ?countryname)}}
LIMIT 25
```

Table 1: Query B-1

2. List the 10 highest averages of C6H6 emission and the country and the year on which they were recorded. Note: A sensor has a property (defined through the prefix: ¡http://qweb.cs.aau.dk/airbase/property/¿) stastisticShortName, and it can be Mean, Max, etc.

```
PREFIX schema: <http://qweb.cs.aau.dk/airbase/schema/>
PREFIX property: <http://qweb.cs.aau.dk/airbase/property/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT  ?C6H6_AVERAGE, ?countryname, ?yearm WHERE {
     SELECT ?station, (avg(?ch) as ?C6H6_AVERAGE), ?countryname, ?yearm WHERE {
     ?obs schema:C6H6 ?ch .
     ?obs schema:station ?station .
     ?station schema:inCountry ?country .
     ?obs schema:year ?year
     bind(strafter(str(?country),str(<http://qweb.cs.aau.dk/airbase/data/country/>))
        as ?country2)
     bind(strbefore(str(?country2),str(</>)) as ?countryname)
     bind(strafter(str(?year),str(<http://qweb.cs.aau.dk/airbase/data/year/>))
        as ?year2)
     bind(strbefore(str(?year2),str(</>)) as ?yearm)
     }
ORDER BY DESC(?C6H6_AVERAGE)}
LIMIT 10
```

Table 2: Query B-2

3. For each city and property type, give the yearly average emission for NO2, SO2, PB, and PM10.

```
PREFIX schema: <http://qweb.cs.aau.dk/airbase/schema/>
PREFIX property: <http://qweb.cs.aau.dk/airbase/property/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?citym, ?stylem, ?yearm, ?avgno2, ?avgso2, ?avgpb, ?avgpm WHERE {
SELECT ?citym, (str(?style) as ?stylem), ?yearm, ( avg(?no2) as ?avgno2),
    ( avg(?so2) as ?avgso2), ( avg(?pb) as ?avgpb), ( avg(?pm) as ?avgpm) WHERE {
     {{?obs schema:SO2 ?so2.}
     UNION
     {?obs schema:NO2 ?no2.}
     UNION
     {?obs schema:Pb ?pb.}
     UNION
     {?obs schema:PM10 ?pm.}}
     OPTIONAL
     {?obs schema:station ?station .
     ?station schema:inCity ?city .
     ?station property:type ?style.
     ?obs schema:year ?year .}
     bind(strafter(str(?city),str(<http://qweb.cs.aau.dk/airbase/data/city/>))
        as ?city2)
     bind(strbefore(str(?city2),str(</>)) as ?citym)
     bind(strafter(str(?year),str(<http://qweb.cs.aau.dk/airbase/data/year/>))
        as ?year2)
     bind(strbefore(str(?year2),str(</>)) as ?yearm)}
ORDER BY ASC(?yearm) DESC(?citym) DESC(?stylem)}
GROUP BY (?stylem)
```

Table 3: Query B-3

Considerations: the ”GROUP BY” is used in order to ensure that if there is any city with more than one station of the same type, their measurements will be taken into account for computing the same average.

4. Define 3 additional SPARQL queries (and their corresponding interpretation) that you think could be interesting for the domain of analyzing air quality/pollution.

First query. In this query we want to check for each year which city had the higher level of pollution. We assume that the level of pollution is given by the overall addition of all measures (here, we only consider a subset of them). Considering the annual average as the usual level for each component. This might seem unrealistic, but being able to get this information reduce it tis just a matter of including the coefficients that model the importance/danger of each pollution agent.

```
PREFIX schema: <http://qweb.cs.aau.dk/airbase/schema/>
PREFIX property: <http://qweb.cs.aau.dk/airbase/property/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?yearm,?citym, ?countryname, ?Total WHERE {
{SELECT  ?yearm,?citym, ?countryname, ?Total WHERE {
     SELECT ?citym, ?countryname, ?yearm, (SUM(?avg) as ?Total) WHERE{
     {SELECT ?city, ?country, ?year, ( avg(?so2) as ?avg) WHERE {
     ?obs schema:station ?station .
     ?station schema:inCountry ?country .
     ?obs schema:SO2 ?so2.
     ?station schema:inCity ?city .
     ?obs schema:year ?year .}}
     UNION
     {SELECT ?city, ?country, ?year, ( avg(?no2) as ?avg) WHERE {
     ?obs schema:station ?station .
     ?station schema:inCountry ?country .
     ?obs schema:NO2 ?no2.
     ?station schema:inCity ?city .
     ?obs schema:year ?year .}}
     UNION
     {SELECT ?city, ?country, ?year, ( avg(?pb) as ?avg) WHERE {
     ?obs schema:station ?station .
     ?station schema:inCountry ?country .
     ?obs schema:Pb ?pb.
     ?station schema:inCity ?city .
     ?obs schema:year ?year .}}
     UNION
     {SELECT ?city, ?country, ?year, ( avg(?pm) as ?avg) WHERE {
     ?obs schema:station ?station .
     ?station schema:inCountry ?country .
     ?obs schema:PM10 ?pm.
     ?station schema:inCity ?city .
     ?obs schema:year ?year .}}
     UNION
     {SELECT  ?year, ?city, ?country, ( avg(?ch) as ?avg) WHERE {
     ?obs schema:station ?station .
     ?obs schema:C6H6 ?ch.
     ?station schema:inCity ?city .
     ?station schema:inCountry ?country .
     ?obs schema:year ?year .}}

     bind(strafter(str(?city),str(<http://qweb.cs.aau.dk/airbase/data/city/>))
        as ?city2)
```

Table 4: B-4, 1st Query

```
     bind(strbefore(str(?city2),str(</>)) as ?citym)
     bind(strafter(str(?year),str(<http://qweb.cs.aau.dk/airbase/data/year/>))
        as ?year2)
     bind(strbefore(str(?year2),str(</>)) as ?yearm)
     bind(strafter(str(?country),str(<http://qweb.cs.aau.dk/airbase/data/country/>))
        as ?country2)
     bind(strbefore(str(?country2),str(</>)) as ?countryname)}}

ORDER BY ASC(?yearm) DESC(?Total)}

OPTIONAL

{SELECT  ?yearm, (MAX(?Total) AS ?maxTotal) WHERE {
     SELECT  ?yearm, ?city, (SUM(?avg) as ?Total) WHERE{
     {SELECT ?city, ?year, ( avg(?so2) as ?avg) WHERE {
     ?obs schema:station ?station .
     ?obs schema:SO2 ?so2.
     ?station schema:inCity ?city .
     ?obs schema:year ?year .}}
     UNION
     {SELECT ?city, ?year, ( avg(?no2) as ?avg) WHERE {
     ?obs schema:station ?station .
     ?obs schema:NO2 ?no2.
     ?station schema:inCity ?city .
     ?obs schema:year ?year .}}
     UNION
     {SELECT ?city, ?year, ( avg(?pb) as ?avg) WHERE {
     ?obs schema:station ?station .
     ?obs schema:Pb ?pb.
     ?station schema:inCity ?city .
     ?obs schema:year ?year .}}
     UNION
     {SELECT ?city, ?year, ( avg(?pm) as ?avg) WHERE {
     ?obs schema:station ?station .
     ?obs schema:PM10 ?pm.
     ?station schema:inCity ?city .
     ?obs schema:year ?year .}}
     UNION
     {SELECT  ?year, ?city, ( avg(?ch) as ?avg) WHERE {
     ?obs schema:station ?station .
     ?obs schema:C6H6 ?ch.
     ?station schema:inCity ?city .
     ?obs schema:year ?year .}}
     bind(strafter(str(?year),str(<http://qweb.cs.aau.dk/airbase/data/year/>))
        as ?year2)
     bind(strbefore(str(?year2),str(</>)) as ?yearm)}}
ORDER BY ASC(?yearm) }
}
GROUP BY ?yearm
HAVING(?Total = ?maxTotal)
LIMIT 25
```

Table 5: B-4, 1st Query - continuation

Second query: Here we wanted a query that allowed us to find how many times the annual average of a city surpassed the worldwide average. Given pollution measure to be specified by the "user", that in this specific case is SO_2.

```
PREFIX schema: <http://qweb.cs.aau.dk/airbase/schema/>
PREFIX property: <http://qweb.cs.aau.dk/airbase/property/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?city, count(?city) as ?Total WHERE{
SELECT ?city, ?year, ?avso2, ?MEAN WHERE {
     {SELECT ?city, ?year, ( avg(?so2) as ?avso2) WHERE {
        ?obs schema:station ?station.
        ?station schema:inCity ?city.
        ?obs schema:SO2 ?so2.
        ?obs schema:year ?year .}}
     OPTIONAL
     {SELECT ?year, ( avg(?so2) as ?MEAN) WHERE {
        ?obs schema:station ?station.
        ?obs schema:SO2 ?so2.
        ?obs schema:year ?year.}}}
   GROUP BY (?city)
   HAVING (?avso2>?MEAN)
   ORDER BY ASC(?city) ASC(?year)}
```

Table 6: B-4, 2nd Query

Third query: In this last query, we wanted to see what types of stations and how many of them each city had over the years.

```
PREFIX schema: <http://qweb.cs.aau.dk/airbase/schema/>
PREFIX property: <http://qweb.cs.aau.dk/airbase/property/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?city, ?tech, (count(?tech) as ?amount), ?year WHERE{

  SELECT ?city, ?year, ?sensor, (str(?tec) as ?tech)  WHERE{
    ?obs schema:sensor ?sensorm.
    ?sensor property:measurementTechnique ?tec.
    ?obs schema:station ?station.
    ?station schema:inCity ?citym.
    ?obs schema:year ?year.

    bind(strafter(str(?citym),str(<http://qweb.cs.aau.dk/airbase/data/city/>))
        as ?city2)
    bind(strbefore(str(?city2),str(</>)) as ?city)
    bind(strafter(str(?sensorm),str(<http://qweb.cs.aau.dk/airbase/data/sensor/>))
        as ?sensor)
   }
}
GROUP BY (?city + ?tech + ?year)
ORDER BY DESC(?year)
```

Table 7: B-4, 3rd Query

C Ontology Creation

C.1 TBox definition

1. Depending on how you created the TBOX, you need to provide either the SPARQL queries you used for creating the TBOX (in case you used SPARQL), or in case you used another tool, the methodology/method you used and the output generated in a graphical form (the lecturer should not install any additional tool to validate this part).

We had used Protege which offers graphical tools to create ontologies, consequently we didn't perform any SPARQL Queries. Therefore, in order to create the Ontology depicted in the following section (figure 1), the procedure we've followed consisted on first of all declare all the classes and subClasses, following a semantic hierarchy, thus, accordingly with that principle, we've defined for instance SuveyPaper, FullPaper, ShortPaper... as sublclasses of the class paper, and Rewier and Author as sublaccles of the class Scientific and so on..., Once the different classes were created we've defined some ObjectProperties to link them and specified its domain and its range, therefore for example to express the relation between a reviewer and a paper we've created a ObjectProperty called reviews with the domain the class Reviewer and the range the class Paper. Finally only remains specify the different interesting properties of the different classes, and to do that, from the dataProperty tab in entities, for the Scientific Class for example we've created properties such as nameScientific and affiliation and specified the scientific class as their domain.
(in the files attached with this document can by find a document called Scientific.owl which is the ontology created with Protege)

2. Provide a visual representation of the TBOX.

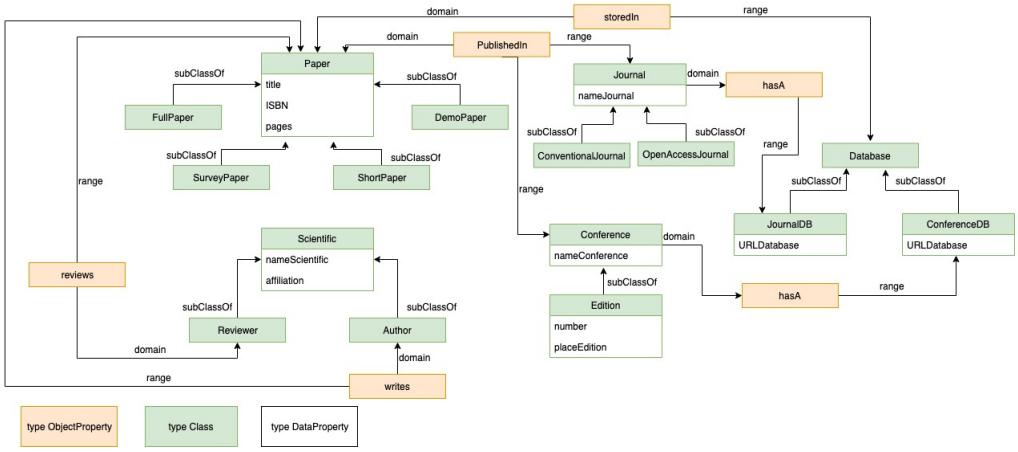

Figure 1: Scientific owl graph

The dataProperties have a relation of rdfs:domain with the classes they have been stated even though it's not depicted in the image, moreover, they link the items with their features, such as the uri identifying a paper with the literals specifying its title and pages for instance.

C.2 ABOX definition

1. Explain the method used to define the ABOX.

To create our TBox we've implemented Open Refine which allowed as to convert the CSV file that we have already prepared on the Property Graph lab session to RDF, and finally with the RDF extension create the triplets. We related each graph instance with the respective csv column that contained its id by using the rdf:type, and the links between the data and its literals using the structure and relations we created in the previous section, which was assigned with the prefix "od".

C.3 Linking ABOX to TBOX

1. Provide the SPARQL queries required to create the link between the ABOX and TBOX

In order to link the TBox and the ABox created previously, first of all it's necessary to upload them to the Virtuoso Workspace by following this steps:

```
1- Download Scientific.owl
2- cd Downloads
3- sudo mv Scientific.owl  /usr/local/virtuoso-opensource/share/virtuoso/vad
4- sudo touch Scientific.owl.graph
5- sudo vi Scientific.owl.graph and add  http://localhost:8890/od
   Repeat the previous process for the Articles_toRDF.ttl
6- Run the serversudo sudo bash â¼/SDM-Software/Virtuoso/start.sh
7- In the Database/Interactive SQL write:
   ld_dir ('/usr/local/virtuoso-opensource/share/virtuoso/vad','*.owl','http://
   localhost:8890/od');
   rdf_loader_run();
   ld_dir ('/usr/local/virtuoso-opensource/share/virtuoso/vad','*.ttl','http://
   localhost:8890/od');
   rdf_loader_run();
```

Table 8: C-3, Graph statistics

Since that in Open Refine the relations between the TBox and ABox has been already defined, once the ABox and TBox is uploaded in virtuoso with the same IRI it's not necessary to use the CONSTRUCT operator.

2. Provide a summary table with simple statistics about the RDF graph obtained, e.g., the number of classes, the number of properties, the number of instances, etc.

The statistics of the graph can be retrieved from the Virtuoso by setting the IRI of the graph of interest in the Linked Data / graph section, therefore, for the IRI 'http://localhost:8890/od' are this:

```
this:Dataset a void:Dataset ;
 rdfs:seeAlso <http://localhost:8890/od> ;
 rdfs:label "Scientific_statistics" ;
 void:sparqlEndpoint <http://localhost:8890/sparql> ;
 void:triples 9054 ;
 void:classes 1009 ;
 void:entities 534 ;
 void:distinctSubjects 534 ;
 void:properties 23 ;
 void:distinctObjects 3045 .
```

Table 9: C-3, Graph statistics

Therefore as it can be appreciated in the figure there are 1009 classes, 534 entities, 534 distinctSubjects, 23 properties, 3045 distinctObjects creating a total of 9054 triples.

C.4 Queries on top of the Ontology

```
PREFIX od: <http://www.semanticweb.org/joanpratsicart/ontologies/2019/3/
    untitled-ontology-3#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
```

Table 10: Used prefixes

1. Find all the Authors.

```
//WITHOUT TBOX

SELECT ?name WHERE { ?a od:writes ?paper.
                     ?a od:nameScientific ?name .}

//WITH TBOX

SELECT ?name WHERE {{?a rdf:type od:Author .
                     ?a od:nameScientific ?name .}
                    UNION
                    {?a od:writes ?paper.
                     ?a od:nameScientific ?name.}}
```

Table 11: C-4, 1st Query

2. Find all the properties whose domain is Author.

```
//WITHOUT TBOX

SELECT DISTINCT(?prop) WHERE {?author od:writes ?paper.
                              ?author ?prop ?c.}

//WITH TBOX

SELECT ?a WHERE { ?a rdfs:domain od:Author.
                 {?a rdf:type owl:ObjectProperty.}
                 UNION
                 {?a rdf:type owl:DataProperty.}}
```

Table 12: C-4, 2nd Query

3. Find all the properties whose domain is either Conference or Journal.

```
//WITHOUT TBOX

SELECT ?prop WHERE {?paper od:publishedIn ?jourconf.
                    ?jourconf ?prop ?b.}

//WITH TBOX

SELECT DISTINCT(?prop) WHERE {{{?prop rdfs:domain od:Journal.}
                               UNION
                               {?prop rdfs:domain od:Conference.}}
                               OPTIONAL
                              {{?prop rdf:type owl:ObjectProperty.}
                               UNION
                               {?prop rdf:type owl:DataProperty.}}}
```

Table 13: C-4, 3rd Query

4. Find all the things that Authors have created (either Reviews or Papers).

```
//WITHOUT TBOX

SELECT ?fa ?name WHERE {{?fa od:title ?name.
                                ?author od:writes ?name.}
                               OPTIONAL
                               {?author od:reviews ?fa}}

//WITH TBOX

SELECT ?b WHERE {{?a rdf:type od:Author.
                  {?a od:writes ?b.}
                  UNION
                  {?a od:reviews ?b.}}
                  UNION
                 {{?a rdf:type od:Reviewer.
                   ?a od:writes ?b.}
                  OPTIONAL
                  {?a od:reviews ?b.}}}
```

Table 14: C-4, 4th Query

Without the TBox

The Tbox ensures us the relation of each item with a class that defines it (specifies what it is), which can be used to easily obtain a given item through the class defining it. However, without the TBOX we have to make assumptions and perform a carefully exploration of the graph, which doesn't ensure its total understanding. Moreover, when introducing data it is necessary to ensure that we respect the semantics hierarchy, which is already guaranteed when using a graph with TBOX.
The assumption of considering that there isn't a TBOX implicitly means that the graph doesn't allow us to make reasoning on the data.