Knowledge Graphs Lab: Team OD-11F

DAVID JUNQUERO GONZALEZ - VÍCTOR SENDINO GARCIA
MEYSAM ZAMANI FOROOSHANI

david.junquero@est.fib.upc.edu - victor.sendino@est.fib.upc.edu
meysam.zamani@est.fib.upc.edu

A) Exploring DBpedia

A.1) Get the classes defined in the ontology.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
SELECT ?x
WHERE {
  ?x    rdf:type  owl:Class .
}
```

A.2) Get the datatype properties defined in the ontology.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
SELECT ?x
WHERE {
   ?x rdf:type owl:DatatypeProperty.
}
```

A.3) Get the object properties defined in the ontology. What is the difference between datatype and object properties?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
SELECT ?x
WHERE {
   ?x rdf:type owl:ObjectProperty .
}
```

DatatypeProperty and ObjectProperty describe what kind of values a triple with the property should have. Datatype properties relate individuals to literal data (e.g., strings, numbers, datetimes, etc.) whereas object properties relate individuals to other individuals.

A.4) Get the labels of all the properties (both datatype and object) defined in the ontology.

```
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#>
SELECT ?label1
WHERE {
        ?x rdf:type ?z.
      ?x rdfs:label ?label1.
      FILTER (?z = owl:DatatypeProperty ||  ?z = owl:ObjectProperty
   )
        FILTER (lang(?label1) = "en")

}
```

A.5) Find the class representing an Actor in the dataset (using filters).

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?x ?z
WHERE {
    ?x rdfs:label ?label1.
    filter contains(?label1,"actor")
    filter langMatches(lang(?label1),'en')
}
```

A.6) Find the super class for the class Actor.

```
PREFIX onto: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?x
WHERE {
   onto:Actor rdfs:subClassOf ?x .
}
```

A.7) Find all the actors in the dataset.

```
PREFIX onto: <http://dbpedia.org/ontology/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?x
WHERE {
    ?x rdf:type onto:Actor.
}
```

A.8) Get different classes that are defined as range of the properties that have the classActor defined as their domain.

```
PREFIX onto: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT DISTINCT ?y
WHERE {
    ?x rdfs:range ?y.
    ?x rdfs:domain onto:Actor.
}
```

A.9) Find the super property of the goldenRaspberryAward property.

```
PREFIX onto: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

```
SELECT ?x
WHERE {
    onto:goldenRaspberryAward rdfs:subPropertyOf ?x.
}
```

A.10) Return all the properties that have the class Actor as either their range or domain.

```
PREFIX onto: <http://dbpedia.org/ontology/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?x
WHERE {
    ?x ?y onto:Actor.
    FILTER (?y = rdfs:range ||  ?y = rdfs:domain)
}
```

A.11) Return all persons that are not actors.

```
PREFIX onto: <http://dbpedia.org/ontology/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?x
WHERE {
    ?x rdf:type onto:Person.
    MINUS { ?x rdf:type onto:Actor }
}
```

A.12) Return the path (in properties and classes) between the Actor and Person classes.

```
PREFIX onto: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?p1 ?c1 ?p2
WHERE {
    onto:Actor ?p1 ?c1.
    ?c1 ?p2 onto:Person
}
```

B) Analytical queries on top of QBAirbase

B.1) List the country, station type, latitude, and longitude details of each station.

```
PREFIX sch: <http://qweb.cs.aau.dk/airbase/schema/>
PREFIX prop: <http://qweb.cs.aau.dk/airbase/property/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT str(?country_name) as ?country, str(?type_name) as ?
   station_type, ?latit, ?longit
WHERE {
?station prop:type ?type_name.
?station prop:latitudeDegree ?latit.
?station prop:longitudeDegree ?longit .
?station sch:inCountry ?country.
?country prop:country ?country_name.
}
```

B.2) List the 10 highest averages of C6H6 emission and the country and the year on which they were recorded.

```
PREFIX sch: <http://qweb.cs.aau.dk/airbase/schema/>
PREFIX prop: <http://qweb.cs.aau.dk/airbase/property/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?c6h6, str(?country_name) as ?country, ?year
WHERE {
?obs sch:C6H6 ?c6h6 .
?obs sch:sensor ?sensor .
?sensor prop:statisticShortName "Mean"^^xsd:string .
?obs sch:station ?station .
?station sch:inCountry ?IRI .
?IRI prop:country ?country_name .
?obs sch:year ?yr .
?yr prop:yearNum ?year .
}
ORDER BY DESC (?c6h6)
LIMIT 10
```

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

```
PREFIX prop: <http://qweb.cs.aau.dk/airbase/property/>
PREFIX sch: <http://qweb.cs.aau.dk/airbase/schema/>
SELECT str(?type) as ?Station_type, str(?city_name) as ?City_name,
   avg(?no2) as ?no2_Value, avg(?so2) as ?so2_Value, avg(?pb) as ?
   pb_value, avg(?pm10) as ?pm10_value
WHERE {
{
?station sch:inCity ?city.
?city prop:city ?city_name.
?station prop:type ?type.
?observation sch:station ?station.
?observation sch:NO2 ?no2.
}UNION{
?station sch:inCity ?city.
?city prop:city ?city_name.
?station prop:type ?type.
?observation sch:station ?station.
?observation sch:SO2 ?so2.
}UNION{
?station sch:inCity ?city.
?city prop:city ?city_name.
?station prop:type ?type.
?observation sch:station ?station.
?observation sch:Pb ?pb.
}UNION{
?station sch:inCity ?city.
?city prop:city ?city_name.
?station prop:type ?type.
?observation sch:station ?station.
?observation sch:PM10 ?pm10.
}}
ORDER BY ASC(?type)
```

B.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: Display the country and city names (without including their IRI), type of pollutant and their values in 2010. Sort the output.

```
PREFIX sch: <http://qweb.cs.aau.dk/airbase/schema/>
PREFIX prop: <http://qweb.cs.aau.dk/airbase/property/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX qb: <http://purl.org/linked-data/cube#>
SELECT str(?country_name) as ?country str(?city_name) as ?city ?
   pollutant ?value
WHERE {
?station prop:type ?type .
?obs sch:station ?station .
?station prop:type "Industrial"^^xsd:string.
?obs rdf:type qb:Observation .
?obs ?p ?value .
?station sch:inCountry ?countryiri .
?countryiri prop:country ?country_name.
?station sch:inCity ?cityy .
?cityy prop:city ?city_name.
?obs sch:sensor ?sensors .
?sensors sch:measures ?component .
?component prop:caption ?pollutant.
FILTER (isLiteral(?value))
?obs sch:year ?year .
?year prop:yearNum "2010"^^xsd:integer .
}
GROUP BY ?country_name ?city_name ?pollutant
order by desc (?value)
```

Second: For all the background stations, display the top 25 countries where O3 had maximum values until 2014.

```
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 ?country ?yn (max(?o3) as ?maxo3)
WHERE {
?s property:type ?b .
?s property:ozoneClassification ?class .
?obs schema:station ?s .
?obs schema:O3 ?o3 .
?obs schema:year ?year .
?year property:yearNum ?yn .
{ ?s schema:inCity ?city .
?city schema:locatedIn ?country .
} UNION {
?s schema:inCountry ?country .
}
?obs schema:sensor ?sensor .
?sensor property:statisticShortName "Max"^^xsd:string .
FILTER(?b = "Background"^^xsd:string && ?yn >= 2000 && ?yn <= 2014
   && ?o3 >300 )
}
GROUP BY ?country ?yn ?o3
order by desc(?o3)
limit 25
```

Third: Display 50 country names having the highest values of emitted CO including the start and end date when those values are measured.

```
PREFIX sch: <http://qweb.cs.aau.dk/airbase/schema/>
PREFIX prop: <http://qweb.cs.aau.dk/airbase/property/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX qb: <http://purl.org/linked-data/cube#>
select str(?country_name) as ?country (max(?co) as ?maxco) ?
   startdate ?enddate
where{
?observation sch:CO ?co.
?observation sch:station ?station.
?station sch:inCountry ?countryiri .
?countryiri prop:country ?country_name.
?observation sch:sensor ?sensor .
?sensor prop:startDate ?startdate.
?sensor prop:endDate ?enddate .
?observation sch:year ?yearN.
?sensor prop:statisticShortName "Max"^^xsd:string.
}
order by desc (?co)
limit 50
```

C) Ontology creation

C.1) TBOX definition

C.1.1) 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 agraphical form (the lecturer should not install any additional tool to validate this part).

To create the TBOX we used Protege. First we generated the classes inside our ontology. We created them using the Entities/Classes tab inside Protege. We took into account if there are subclass relations between classes (like for Conference and DatabaseConference) as well as disjointedness. We have all the concepts in the statement.

Regarding the hierarchies, we have a few of them. First, we distinguish between Author and Reviewer, that have a superclass named Person that holds the properties like name, age, etc. Author and Reviewer are not disjoint, because a person can be both. We also have the class Paper, and four kind of papers that are subclasses and disjoint between them. For Conference and Journal, both have one specific subclass (Conference Database and JournalVolume). We also have ConferenceEdition and JournalVolume, in order to keep track of at which edition/journal, a paper was published, and at the same time know the number of editions/volumes a certain Conference/Journal has. ConferenceEdition and JournalVolume are disjoint (assumming that a certain paper cannot be published in both a conference and a journal), and have a superclass named Publications that allows to share the property publishedIn.

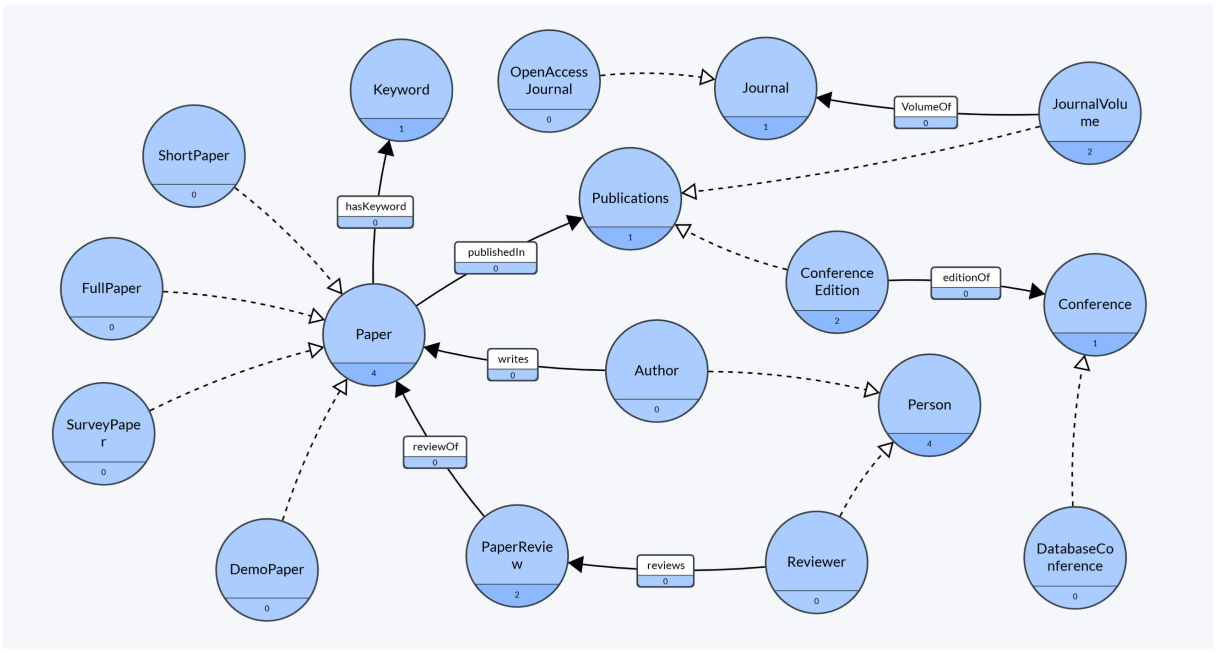
Then, we created the Object Properties that relate individuals to other individuals (with the Entities/Object properties tab of Protege). We also indicated the domain and range for this properties. For instance the "writes"property that links Author (domain) and Paper (range). The properties generated are the same we had in the first lab of the Course.

Lastly, we generated the Datatype Properties, that link individuals with literal data (with the Entities/Data properties tab of Protege). The domain are the individuals and the range are the type of the literals. For example we have the "age"property, whose domain is Person, and the range is xsd:nonNegativeInteger, since age is a non negative integer. The most part of types are xsd:string, except for some numeric concepts like age, volume, nPages, edition, and also a xsd:boolean to indicate the decision in a PaperReview (Accepted: 1 or Rejected: 0).

C.1.2) Provide a visual representation of the TBOX (You can use https://gra.fo/ for instance).

To visualize the TBOX, First we export owl from Protégé with RDF/XML syntax and then we used the recommended tool for creating visual representation which is Grafo. you can finf the visual representation of the TBOX in figure 1:

FIGURA 1 – visual representation of the TBOX



C.2) ABOX definition

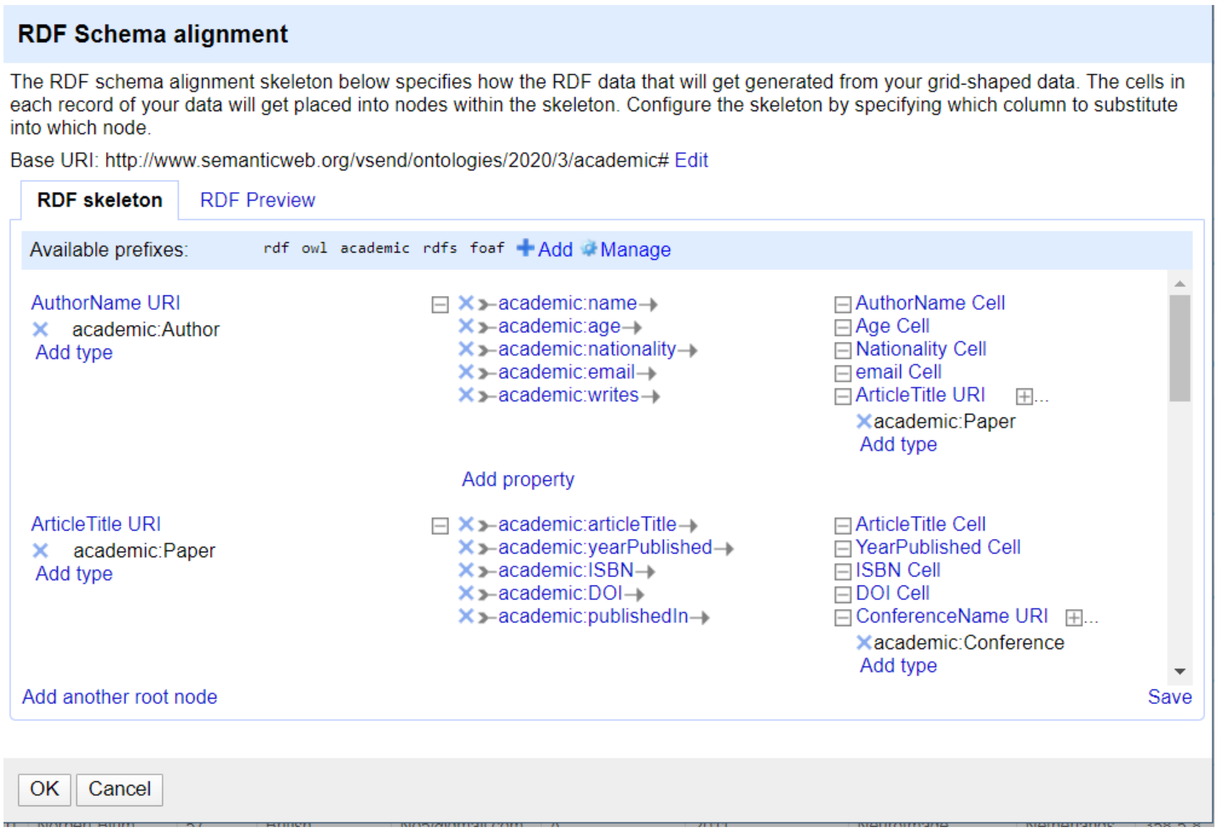
C.2.1) Explain the method used to define the ABOX.

In this step we used Open Refine with the RDF extension. We had two different CSV containing all the information for papers published in a Conference and in a Journal respectively. With Open Refine, we read the CSV's and convert the data to RDF. First we introduce a CSV file and create a new project. Next we click on extensions/RDF/edit RDF skeleton and a new window opens.

Here we must specify as base URI http://www.semanticweb.org/vsend/ontologies/-2020/3/academic# and also we upload the ontology we had in the owl file, so Open Refine can access to it and know the classes and properties defined. Then we must link the data from each row of the CSV to the triplets of our graph. From the information in a row, we can fill plenty of triplets, specifying the subject, predicate and object. We must select from which cell of the CSV will come the information and for the target, the class or property of our ontology. Once we have all done, we export the file in a .rdf format. Because we have two different CSV, we must repeat this process twice and thus we have one .rdf file for Conference data and another for Journal data.

Since Open Refine can have access to our previously defined TBOX in a .owl file, if we use it during the process, we are already automatically linking TBOX and ABOX. In the following figure we have an small fragment of our Open Refine project that shows the linkage for an Author and a Paper individual.

FIGURA 2 – Open refine transformation from CSV to RDF



RDF Schema alignment

The RDF schema alignment skeleton below specifies how the RDF data that will get generated from your grid-shaped data. The cells in each record of your data will get placed into nodes within the skeleton. Configure the skeleton by specifying which column to substitute into which node.

Base URI: http://www.semanticweb.org/vsend/ontologies/2020/3/academic# Edit

C.3) Linking ABOX to TBOX

C.3.1) Provide the SPARQL queries required to create the link between the ABOX and TBOX.

As explained before, Open Refine can link directly TBOX and ABOX, so we don't have queries for this section. Once we upload to GraphDB the .owl file with the TBOX and the .rdf file with the ABOX, we have everything linked successfully.

C.3.2) 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.

TBOX

Class count	17
Object property count	7
Data property count	18
SubClassOf	10
DisjointClasses	2
ObjectPropertyDomain	7
ObjectPropertyRange	7
DataPropertyDomain	18
DataPropertyRange	18

ABOX_Conference

Class count	7
Individual count	3769
Annotation propert count	20
ClassAssertion	4122
AnnotationAssertion	18043

ABOX_Journal

Class count	7
Individual count	3803
Annotation propert count	20
ClassAssertion	4152
AnnotationAssertion	18525

C.4) Queries on top of the Ontology

C.4.1) (Assuming the TBOX exist) Find all the Authors.

```
PREFIX academic: <http://www.semanticweb.org/vsend/ontologies
   /2020/3/academic#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?x
WHERE {
    ?x rdf:type academic:Author.
}
```

C.4.1) (Assuming the TBOX does not exist) Find all the Authors.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?x ?t
WHERE {
    ?x rdf:type ?t.
    filter contains(str(?t),"Author")
}
```

C.4.2) (Assuming the TBOX exist) Find all the properties whose domain is Author.

```
PREFIX academic: <http://www.semanticweb.org/vsend/ontologies
   /2020/3/academic#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?x WHERE {
  ?x rdfs:domain academic:Author.
}
```

C.4.2) (Assuming the TBOX does not exist) Find all the properties whose domain is Author.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?x WHERE {
  ?x rdfs:domain ?t.
  filter contains(str(?t),"Author")
}
```

C.4.3) (Assuming the TBOX exist) Find all the properties whose domain is either Conference or Journal.

```
PREFIX academic: <http://www.semanticweb.org/vsend/ontologies
   /2020/3/academic#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?x
WHERE {
    ?x rdfs:domain ?z.
    FILTER (?z = academic:Conference ||  ?z = academic:Journal)
}
```

C.4.3) (Assuming the TBOX does not exist) Find all the properties whose domain is either Conference or Journal.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?x
WHERE {
    ?x rdfs:domain ?z.
    FILTER (strEnds(str(?z),"Conference") ||  strEnds(str(?z),"
   Journal"))
}
```

C.4.4) (Assuming the TBOX exist) Find all the things that Authors have created (either Reviews or Papers)

```
PREFIX academic: <http://www.semanticweb.org/vsend/ontologies
   /2020/3/academic#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?x ?y ?z ?t
WHERE {
    ?x ?y ?z.
    ?z rdf:type ?t.
    ?x rdf:type ?a
    FILTER (?a = academic:Author || ?a = academic:Reviewer)
    FILTER (?t = academic:PaperReview || ?t = academic:Paper)
}
```

C.4.4) (Assuming the TBOX does not exist) Find all the things that Authors have created (either Reviews or Papers)

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?x ?y ?z ?t
WHERE {
    ?x ?y ?z.
    ?z rdf:type ?t.
    ?x rdf:type ?a
    FILTER (strEnds(str(?a),"Author") || strEnds(str(?a),"Reviewer"
   ))
    FILTER (strEnds(str(?t),"PaperReview") || strEnds(str(?t),"
   Paper"))

}
```

Assumption: Due to not having access to the ontology on the second set of queries, one way of adapting the queries is to filter all classes in the environment and finding the required ones according to their name (converted to String). While in some situations this might retrieve information from unrelated sources, we assume and we made sure to filter them in a way that these classes contain the information needed without referencing the TBOX.