Alícia Chimeno Sarabia
Adrià Casanova Lloveras May 29, 2024

Knowledge Graphs LAB Assignment

Semantic Data Management Project

This project is organized within the specified GitHub repository [1] , where all scripts pertinent to the sections outlined below are available. Additionally, the applications have been included in the provided ZIP file.

Abstract - There are several ways to deal with Semantic Data, with graphs being a prominent method. Knowledge graphs, in particular, are specially good and simple to model data that can be grouped into taxonomies. This Semantic Data Management project focuses on the creation and exploration of a knowledge graph using RDFLib and SPARQL.
key words - *ontology creation, data instantiation, TBOX, ABOX, ontology querying*

B Ontology creation

The creation of a knowledge graph involves defining an ontology, which includes defining the TBox to establish the schema, and loading the data into the ABox to populate the graph with instances and relationships.

B.1 TBOX definition

After evaluating all available resources, we chose an external tool called **RDFLib** to create the TBOX. We decided to use the **RDFS knowledge graph language** instead of OWL or RDF. RDFS provides the necessary flexibility and simplicity for our project's requirements, making it a more suitable choice for our specific needs. Additionally, RDFLib offers robust support for RDFS, facilitating efficient and effective development.

The graphical representation of the knowledge graph is depicted in the following figure. However seems not readable, we provide the link to the graph.

Note that inferred triples, of the kind (:Paper, rdf:type, rdf:Class), are not represented in the graph to make it more understandable. Apart from rdf and rdfs, we used the xsd namespace. xsd provides datatype definitions that allow us to restrict the range of properties involving literals, like (:name_author, rdfs:range, xsd:string).

The overall URL we defined for our ontology is *http://SDM.org/Lab2/*.

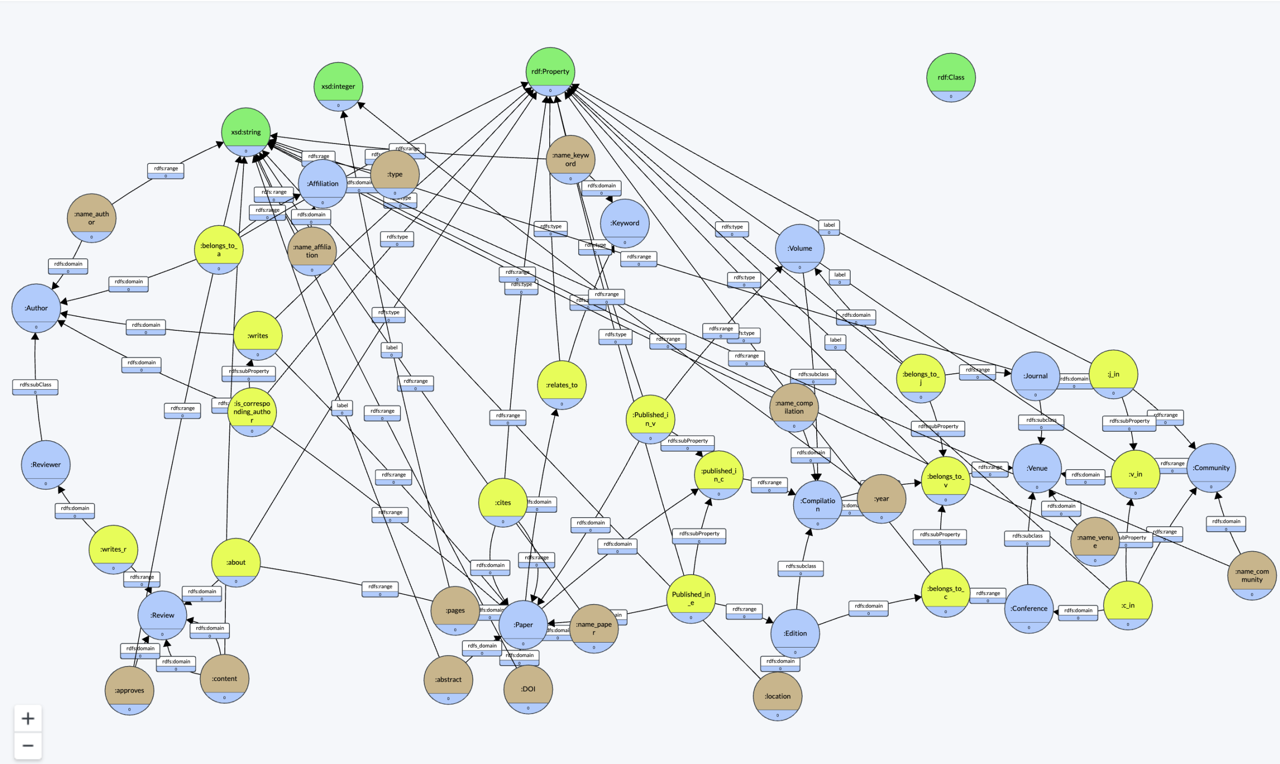

Figure 1: Representation of the knowledge graph. [2]

B.2 ABOX definition

We defined the ABOX from a set of csv files containing the graph data. We had two types of csv's.

On the one hand, some csv's contained data about nodes' attributes (colored in brown in the graph representation), which were modeled as literals in the graph. Given a csv file containing data of a certain class, for each row we defined a URI and asserted all triples relating that URI to a literal (found in a column of the csv). That is, we asserted a member of the class and its properties.

The rest of csv's corresponded to properties with domain and range from our graph's namespace. Given a certain property, its csv file contained a row for each of its instances and two columns. One identified the subject node and the other referenced the object node. Hence, we asserted every instance of the property by traversing the csv.

The classes *review* and *reviewer* and the properties *writes_r* and *about* were instanced in a different way, however. We had a csv file with columns identifying the reviewer, paper, content and approval of each review. Now, from this csv file we asserted the aforementioned classes and properties in a similar way than the rest.

In a similar way, we had one csv file for both *writes* and *is_corresponding_author*. This file contained a binary column that distinguished which property had to be asserted between the related author and paper.

Our TBOX had 5 *rdfs:subClass* instances. For most of them we simply asserted the nodes of the subclass and let inference assert the proper *rdf:type* instances. Realize that subclasses share attributes with their superclasses. For example, a journal will have an attribute *name_venue* even though it is a journal, which simplifies the notation. The *reviewer* and *author* were more complicated to handle, however. As we have already explained, reviewers were asserted from the reviews csv file. Nonetheless, those nodes were also in the authors csv file. To avoid having two nodes representing the same individual, we only asserted as authors those authors who did not participate in any review. Note that later reviewers would be inferenced as authors.

When a property had a subproperty, we only asserted the subproperty, since the property would be inferenced.

Finally, we will discuss how URI's were created. All nodes share the same URL, *http://SDM.org/Lab2/*. When asserting a node, we would define its URN as the concatenation of its class' URN and its row number in its csv. Then, we joined nodes csv's and properties csv's to assert properties with the correct node URI's.

We also considered the option of using row ids as URN's. However, many characters used, for example in paper's titles, are invalid in URI's. We could replace or remove invalid characters, but then URN's would not be unique. That is why we ended up discarding this solution.

We checked data quality by dropping rows with null or duplicate ids in csv's.

B.3 Create the final ontology

We are considering the *RDFS (Optimized)* inference regime entailment. Thanks to it, we have saved all *rdf:type* links except for the ones with object *rdf:Property*. In the TBOX, we have achieved this because every class is the domain or range of at least one property, so the links will be inferenced. Now, since every node in the ABOX has an attribute, the domain property of attributes will inference the *rdf:type* links of these nodes.

RDFS (Optimized), however, does not infer the *rdf:type* links with object *rdf:Property*, so we specified them in the TBOX.

To briefly describe our knowledge graph, we will provide a summary of our instances, computing the **number of classes, the number of properties, number of instances for the main classes and number of triples using the main properties.**

B.3.1 Number of classes

To compute the number of classes, we debated between the following two queries because they gave us different results:

```sparql
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT (COUNT(DISTINCT ?class) AS ?numberOfClasses)
WHERE {?class rdf:type rdfs:Class.}
```

and

```sparql
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT (COUNT(DISTINCT ?class) AS ?numberOfClasses)
WHERE {?s rdf:type ?class .}
```

In the first query, we queried in the tbox and retrieved 28 classes because we loaded rdf and rdfs. In the second query, we queried on the abox only got 18 classes because we only used rdf. The result classes are listed in Table 4.

B.3.2 Number of properties

Same for the property query, we queried:

```sparql
SELECT DISTINCT ?property
WHERE {?s ?property ?o .}
```

The result is that we have a list 5 of 36 properties.

B.3.3 Number of instances

Table 1: Summary Statistics

Category	Number of Instances
Papers	7985
Authors	2916
Editions	30
Volumes	814
Compilations	844
Conferences	20
Journals	20
Venues	40
Communities	7

Table 2: Statistics of the Knowledge Graph

Statistic	Value
Number of Classes	18
Number of Properties	36

B.3.4 Number of triples using the main properties

We computed the query in 2, and got the following result

Table 3: Triples Count for Selected Properties

Property	Count
rdf:type	21400
:relates_to	6000
:about	5985
:writes_r	5985
:content	5985
:approves	5985
:cites	5907
:writes	4675
:name_author	2915
:belongs_to_a	2915

B.4 Querying the ontology

1. Find all Authors.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX : <http://SDM.org/Lab2/>
```

```

SELECT ?a
WHERE {?a rdf:type :author .
}
LIMIT 10
```

Listing 1: Example SPARQL Query

2. Find all properties whose domain is Author.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX : <http://SDM.org/Lab2/>

SELECT ?p
WHERE {
  ?p rdfs:domain :author .
  #?p rdf:type rdf:Property.
}
```

Listing 2: Query 2

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

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX : <http://SDM.org/Lab2/>

SELECT ?p
WHERE {
    #?p rdf:type rdf:Property .
    {?p rdfs:domain :conference} UNION {?p rdfs:domain :journal }
}
```

Listing 3: Query 3

4. Find all the papers written by a given author that where published in database conferences.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX : <http://SDM.org/Lab2/>

SELECT ?a ?p
WHERE {
  ?p rdf:type :paper .
  ?p :published_in_c ?cmpl .
  ?cmpl :belongs_to_v ?ven .
  ?ven :v_in ?com .
  ?com :name_community "Database" .
  ?a :writes ?p .
  ?a rdf:type :author .
}
ORDER BY ?a
```

Listing 4: Query 4

5. (Paper Ranking). Identify the top 5 papers in the database community. Our goal is to find the most cited papers within this community, specifically those that are most frequently cited by other papers in the same community. A paper is considered part of the database community if it was published in a venue associated with this community. Also indicate the venue which the paper belongs to.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX : <http://SDM.org/Lab2/>

SELECT ?ven1 ?p1 (COUNT(?p1) AS ?timescited)
WHERE {
  #?p rdf:type :paper .
  ?p :published_in_c ?cmpl .
  ?cmpl :belongs_to_v ?ven .
  ?ven :v_in ?com .
  ?com :name_community "Database" .

  ?p :cites ?p1 .

  #?p1 rdf:type :paper .
  ?p1 :published_in_c ?cmpl1 .
  ?cmpl1 :belongs_to_v ?ven1 .
  ?ven1 :v_in ?com1 .
  ?com1 :name_community "Database" .
}
GROUP BY ?p1 ?ven1
ORDER BY desc(?timescited)
Limit 5
```

6. (Conference ranking). Identify the top conference for attending in the NLP field. We need to find the conference with the highest number of papers containing the keyword ”NLP” and also the highest frequency of citations to other NLP-related papers.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX : <http://SDM.org/Lab2/>

SELECT ?conference (COUNT(?paper) AS ?nlp_paper_count) (SUM(?citeCount)
    AS ?sum_citations_nlp)
WHERE {
  ?paper rdf:type :paper .
  ?paper :relates_to ?keyword .
  ?keyword :name_keyword "NLP" .
  ?paper :published_in_c ?compilation .
  ?compilation :belongs_to_v ?conference .
  ?conference rdf:type :conference .

  ?paper :cites ?citedPaper .

  ?citedPaper rdf:type :paper .
  ?citedPaper :relates_to ?keyword .
```

```
  #?keyword :name_keyword "NLP" .


  {
    SELECT ?citedPaper (COUNT(?citingPaper) AS ?citeCount)
    WHERE {
      ?citingPaper :cites ?citedPaper .
    }
    GROUP BY ?citedPaper
  }
}
GROUP BY ?conference
ORDER BY DESC(?nlp_paper_count) DESC(?sum_citations_nlp)
LIMIT 5
```

B.5 Conclusions

Knowledge graphs stand out above other types of graphs for analysis because:

- They leverage inferred knowledge, thus reducing the need for extensive manual definition within the graph.
- They incorporate taxonomies, which organize classes into hierarchical structures based on their characteristics or relationships, enhancing the graph traversal and the organization and facilitating more efficient navigation and exploration of interconnected data.

Appendices

No.	Class
1	rdf:Property
2	rdfs:Class
3	rdf:List
4	rdfs:Datatype
5	rdfs:ContainerMembershipProperty
6	:author
7	:reviewer
8	:affiliation
9	:paper
10	:review
11	:volume
12	:compilation
13	:keyword
14	:conference
15	:venue
16	:edition
17	:journal
18	:community

Table 4: Enumeration of Classes

Table 5: Enumeration of Properties

No.	Property
1	rdf:type
2	rdfs:subPropertyOf
3	rdfs:subClassOf
4	rdfs:domain
5	rdfs:range
6	:about
7	:belongs_to_a
8	:belongs_to_c
9	:belongs_to_v
10	:belongs_to_j
11	:cites
12	:is_corresponding_author
13	:writes
14	:published_in_e
15	:published_in_c
16	:published_in_v
17	:relates_to
18	:writes_r
19	:name_author
20	:DOI
21	:abstract
22	:name_paper
23	:pages
24	:name_venue
25	:location
26	:name_compilation
27	:year
28	:approves
29	:content
30	:name_affiliation
31	:type
32	:name_keyword
33	:name_community
34	:c_in
35	:j_in
36	:v_in

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://SDM.org/Lab2/>

SELECT ?property (COUNT(?s) AS ?numTriples)
WHERE {
  {
SELECT ?property WHERE {
      VALUES ?property {
        rdf:type
        rdfs:subPropertyOf
        rdfs:subClassOf
        rdfs:domain
        rdfs:range
        :about
        :belongs_to_a
        :belongs_to_c
        :belongs_to_v
        :belongs_to_j
        :cites
        :is_corresponding_author
        :writes
        :published_in_e
        :published_in_c
        :published_in_v
        :relates_to
        :writes_r
        :name_author
        :DOI
        :abstract
        :name_paper
        :pages
        :name_venue
        :location
        :name_compilation
        :year
        :approves
        :content
        :name_affiliation
        :type
        :name_keyword
        :name_community
        :c_in
        :j_in
        :v_in
      }
    }
  }
  ?s ?property ?o .
}
GROUP BY ?property
ORDER BY DESC(?numTriples)
LIMIT 10
```


Figure 2: Caption

References

[1] Adrià Casanova Alicia Chimeno. KnowledgeGraphs_SDM. https://github.com/airdac/SDM-Knowledge_Graphs, 2024.

[2] Grafo. Graph visualization. https://app.gra.fo/editor/542c0c59-d7ab-45dd-8315-3d6241cbd984/public?token=93c70021a27f7e578c3269be6a0fa03d76c1f66faaabb4c58137e4b9db7837a6, 2024.