Master in Data Science

Knowledge Graphs

Semantic Data Management

Year
2023/24

Team members
Fortó, Marc
Vayá, Gabriel

A Introduction

This work presents a comprehensive project focused on modeling and querying graph data related to research publications within the database community domain. The primary aim is to construct a knowledge graph representing the different aspects of those publications like for example, Authors, Papers, Conferences, Journals, and their interrelationships. Key objectives include querying the graph data to retrieve relevant information on the domain, extracting the most of the particular processes undergone with semantic data.

The project highlights the importance of transforming regular data into semantic data, and implementing SPARQL queries in GraphDB, properly modelling it to take advantage of inference done by the platform. Additionally, it emphasizes using real data sources to create the graph and explaining the reasons behind design decisions, which are crucial aspects of any work using semantic data.

B Ontology Creation

The creation of an ontology out of regular data is the cornerstone of any Semantic Data project. It has multiple key aspects that are not trivial and have to be taken the proper care, since a minor error in any of those can result in an incomplete model, or even worse, unreliable information out of the querying.

B.1 TBOX definition

In knowledge graphs, the TBOX of the ontology is the set of rules that define the relationships between the semantic units of our data. For its definition, we have taken into account the directions presented in the project guidelines, as well as included some additional taxonomies in order to optimize some queries that are regarded to be relevant in the management of this data. The TBOX has been created using GRAFO, a visual platform commonly used for data modelling and representation using Graphs. The visual representation of the TBOX can be appreciated in Fig. 1.

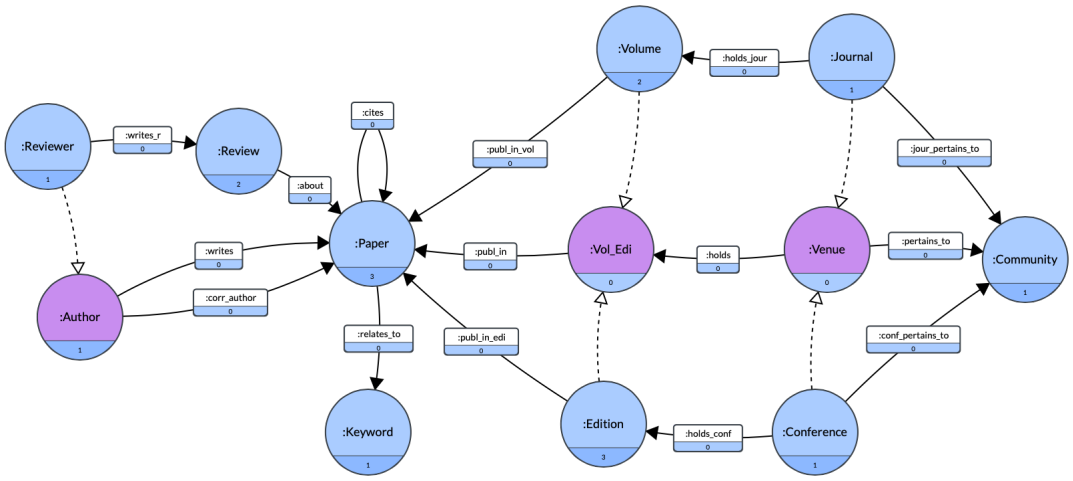

Figure 1: TBOX Knowledge Graph

The **:Paper** is defined as a node, having its title, abstract and DOI as attributes, with **:cites** a reflexive relationship with itself representing the citation of a paper by another paper. Then, instead of defining authors and reviewers separately, we note that both are basically the same semantic entity (or taxonomy), so we define **:Reviewer** as a subclass of **:Author**, each having the name as the corresponding attribute. Now, authors can be the corresponding author of a paper (meaning the main author) or not, hence two different relations ought to be defined, **:corr_author** for the corresponding author, and **:writes** for the rest. When it comes to the reviewers, we consider that each reviewer writes a review through the relationship **:writes_r**, described in the node **:Review** with attributes content (containing the review itself) and decision, describing whether or not the paper is accepted by the reviewer. Obviously, reviews are related to papers, modeled through the relationship **:about**.

Moving on, as stated in the previous work, papers can be published in journals' volumes or conferences' editions. We have considered a **:Journal** node and a **:Conference** node (with name as attribute in both), the first related to the corresponding node **:Volume** through **:holds_jour** with attributes name and year, and the second related to the corresponding node **:Edition** through **:holds_conf** with attributes name, number and year. Finally, editions are related to papers through **:publ_in_edi** and volumes are related to papers through **:publ_in_vol**. It is worth mentioning that the explicitations (names) of the relationships have to be different in order to have the correct results in querying the ontology. Now, in this context we note that another taxonomy can be defined, which easily can facilitate query processing in the future. The semantic structure for paper publications through journals and conferences is completely analogue. Thus, we can define a superclass node **:Venue** having journals and conferences as subclasses, and another superclass **:Vol_edi** having edition and volume as subclasses. This way, venues are related to volumes-and-editions through **:holds** and volumes-and-editions are related to papers through **publ_in**.

However, having operated with GRAFO up to this point, means that, even though the relationships of subclass-superclass are well inferred by the software once we download the .ttl file, it was observed that the relationships between the edges between :Paper, :Vol_edi, :Venue and :Community are not properly defined as subPropertyOf. Consulting the website of GRAFO, the concerns were quickly confirmed, and it is something that needs to be tackled if we desire to extract the most out of the semantic model we established. Nevertheless, the solution is simple, since only a few extra lines are needed in the statement of the concepts to define those relationships in the TBOX, which were manually written before uploading it to GraphDB.

To conclude the TBOX, journals and conferences are included in communities, represented with the node **:Community** with the name as an attribute and, finally, papers are related to topic having certain keywords, represented in the graph with the node **:Keyword** with the corresponding keywords as attributes, the papers being related to it through the edge **:relates_to**. The visual representation was manually built in GRAFO, and a .ttl file was downloaded with the explicitation of the RDFS core properties. The file is included in the deliverable ZIP file.

B.2 ABOX definition

Once the TBOX is created, expliciting the semantic structure of our data, it is time to give to the ontology the real data that forms the knowledge graph. For that, we built a python code that constructed the ABOX programatically. The code is included in the ZIP file as well.

In the code, first we load the .ttl file of the TBOX previously downloaded from GRAFO, and then explicitly define the properties that are going to be predicates in the triples. Some of those include:

- about
- cites
- corr_author
- holds
- holds_conf
- holds_jour
- ...

The paper data used in this work as we inherited from the Property Graphs project, was distributed in several .csv files, one for each node and edge in the property graphs work. For instance, the .csv file for the paper node contains the *paper_id*, *title*, *abstract* and *DOI*, whilst the .csv for the edge between the paper and an author contains the *paper_id*, the name of the author and a boolean column that indicates if this author is the corresponding author or not.

Thus, to build the ABOX, having defined all properties between the concepts as predicates, we create a RDFLib empty graph, and we manually iterate triple by triple of our TBOX, explicitly defining which columns of our .csv files correspond to the subject and object for each predicate in the function *add_data_from_csv*. Once all triplets are loaded, we save the ABOX in a .ttl file, ready to create the final ontology. To give some examples, we used the paper node to load the rdfs:Class **:Paper**, and define its attributes, and used the edge between paper and authors to load the subgraph **:Paper** - (**:writes**/**:corr_author**) - **:Author**. For further specifics see the code attached.

One particular case worth mentioning, is the case about authors and corresponding authors, since the load of this triplet cannot be done directly (we do not have separate .csv for authors and corresponding authors, but a boolean variable that sets them apart, as explained previously). To solve that, first we load every author as normal authors, so not corresponding authors. Afterwards, we iterate again through the edge between the paper and the author, and if the boolean variable is true, we also added the predicate of *:corr_author*.

B.3 Creating the final ontology

In this section, given that in the construction of the ABOX we used the literal fields defined earlier, we noticed that the link between the TBOX and the ABOX was automatically carried by GraphDB. Since we experimented no noticeable problems in the querying stage and the statistics were normal as well, we deemed no further action as necessary. Now we give some basic statistics on the semantic database created in this work.

B.3.1 Statistics of RDF Data

Number of Classes Number of classes: 28

Property	numTriples
rdf:type	15354
relates_to	6000
about	5982
writes_r	5982
content	5982
decision	5982
cites	5907
writes	4675
reviewer_name	2548
abstract	2000
doi	2000
title	2000
vol_name	814
vol_year	814
rdfs:subPropertyOf	46
rdfs:subClassOf	41
rdfs:domain	36
rdfs:range	36
edi_name	30
edi_num	30
edi_year	30
conf_name	20
jour_name	20
keyword	14
jour_pertains_to	13
conf_pertains_to	12
community_name	7

Table 1: Number of Triplets for Each Property

Class	Number of Instances
num_papers	2000
num_authors	2915
num_vol	814
num_ed	30
num_conf	20
num_journal	20
num_ven	40
num_com	7

Table 2: Number of Instances for the Main Classes

Having considered that all is in order, we considered the ontology has been successfully created, and we can now proceed to the last stage of the work, which is querying the ontology.

B.4 Querying the ontology

This is the final stage of the work, where we test the performance of the semantic model, and we see both if the relationships and inference have been well defined, and if the data has been properly loaded in the knowledge graph.

1. Find all Authors

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX : <http://www.gra.fo/schema/sdm/>
SELECT ?a
WHERE { ?a rdf:type :Author . }
```

Listing 1: Find all Authors

2. Find all properties whose domain is Author

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX : <http://www.gra.fo/schema/sdm/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?p
WHERE {
  ?p rdfs:domain :Author .
  ?p rdf:type rdf:Property.
}
```

Listing 2: Find all properties whose domain is Author

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

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://www.gra.fo/schema/sdm/>
SELECT ?p
WHERE {
    #?p rdf:type rdf:Property .
    {?p rdfs:domain :Conference} UNION {?p rdfs:domain :Journal}
}
```

Listing 3: Find all properties whose domain is either Conference or Journal

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

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX : <http://www.gra.fo/schema/sdm/>
SELECT ?a ?p
WHERE {
  ?p rdf:type :Paper .
  ?e :publ_in_edi ?p .
  ?conf :holds_conf ?e .
  ?conf :conf_pertains_to ?com .
  ?com :community_name "Database" .
  ?a :writes ?p .
  ?a rdf:type :Author .
}
```

Listing 4: Find all papers by a given author published in database conferences

The first extra query we have thought to be an interesting way of using the graph is the following:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX : <http://www.gra.fo/schema/sdm/>

SELECT ?rev (COUNT(?p) AS ?numPapers)
WHERE {
    ?v rdf:type :Venue .
    ?v :holds ?ve .
    {
        ?ve :vol_year ?volYear .
        FILTER(?volYear < '2015')
    } UNION {
        ?ve :edi_year ?ediYear .
        FILTER(?ediYear < '2015')
    }
    ?ve :publ_in ?p .
    ?r :about ?p .
    ?r :decision 'FALSE' .
    ?rev :writes_r ?r .
}
GROUP BY ?rev
ORDER BY DESC(?numPapers)
```

Listing 5: How many papers have been rejected by each author acting as a reviewer

In this query, we are retrieving how many papers have been rejected by each author acting as a reviewer before a certain year. For that, since the year is not an attribute of a paper, but of an edition or volume, it is imperative to use the taxonomies defined between conferences and journals, since they provide a much simpler and efficient way to retrieve this information. We start by accessing the venue, and through *:holds* we access the taxonomy between edition and volumes. It is true that this not needed, since we could start from :Vol_edi directly, but this way we prove that the rdfs:subProperty is well defined. It is in the class :Vol_edi that the filter for the year is defined, and then, through :Paper and :Review we access the :Reviewer class, grouping by the reviewers and counting those papers whose review has the value of the attribute *:decision* as false. Some people are hard to impress, this way we can know which reviewers are the most tough and demanding in their reviews!

The second query is written as follows:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX : <http://www.gra.fo/schema/sdm/>

SELECT ?v (COUNT(?p1) AS ?numCites)
WHERE {
  ?p1 rdf:type :Paper .
  ?p1 :cites ?p2 .
    ?ev :publ_in ?p2 .
```

```
    ?v :holds ?ev .
}
GROUP BY ?v
ORDER BY DESC (?numCites)
```

In this last one, we are interested in knowing what are the most relevant conferences and journals based on the number of citations. For that we make again use of the :Venue to simplify enormously the query, and make it easier to compute. Using the reflexive property *:cites*, we retrieve the papers related by citations. Bear in mind that as the property is defined in our graph, the cited paper is in the range of the relationship, hence **p2** is the cited paper and **p1** is the citing paper. Now we just group up by :Venue, and count the number of citations the papers published in each venue accumulate. This query is useful to know what conferences or journals are the most popular and produce better or more relevant papers in the academic world.

C Conclusions

To conclude with, in this work we have learnt a comprehensive manner to transform non-semantic data to a knowledge graph model, and extract the most out of the semantic properties that this field brings to the table. Overall, we are satisfied with both the methodology and the results, and we find that the queries we came up with showed that the solution is well adapted for the retrieval of relevant information on the domain.