Knowledge Graphs

David García Morillo & Nils van Es Ostos

Spring 2024

Introduction

This is a hands-on lab on knowledge graphs. We will use the GraphDB database for working with knowledge graphs. In this lab, a graph database on papers related to Data Science will be create using data requested from Semantic Scholar[2]. All the code used in this lab can be found on GitHub[3].

1 Preprocess

The data used in this lab consists of the data used in the previous lab session: Property Graph DB[4]. However, some modifications were made to this datasets to add some extra features that were not present in the previous lab. The modifications are the following:

- An extra column was added to the *conference_semantics.csv* file that classifies the given venue in as a conference or a workshop.
- A conference chair was added to *conference_semantics.csv*. This person assigns a set of reviewers (typically three) to each paper. These reviewers are randomly chosen among the authors of dataset.

Let us notice, that all the information added has been synthetically generated since the information was not originally in the dataset.

2 TBOX Definition

In this section, the model for the papers infrastructure has been created. The schema of the model is represented in the Fig.(1). Furthermore, the code for the definition of the *TBOX* is found in the jupyter notebook[3] under the section *"B1. TBOX Definition"*.

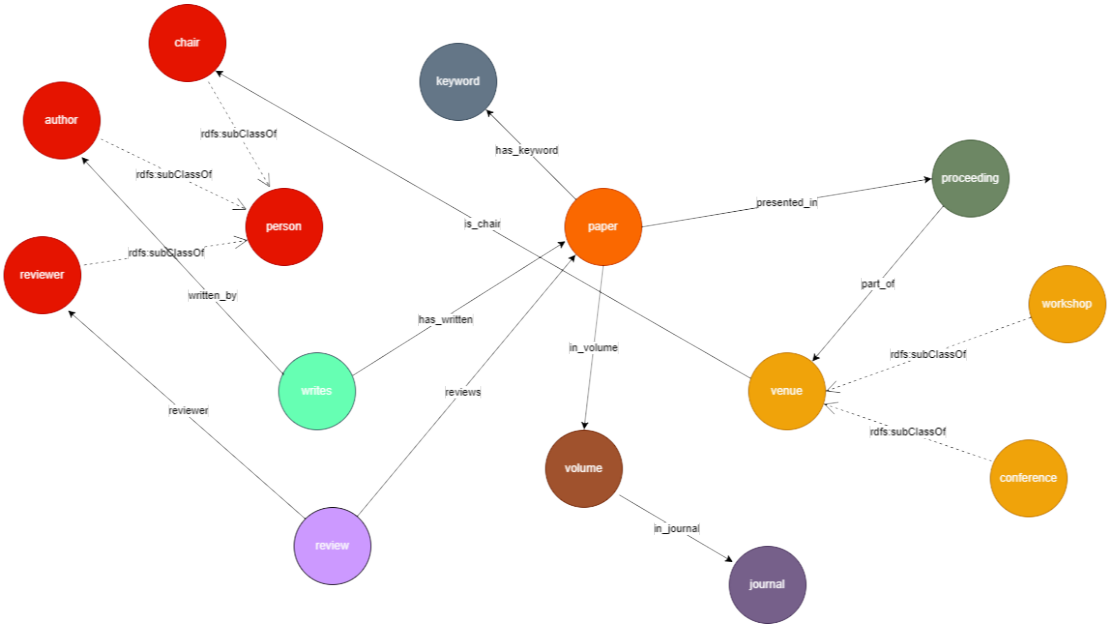

Figure 1: Representation of the knowledge graph.

The model can be summarized in the following points:

1. **Paper:** represents the central class of the model. Among its properties, there is *paper_id*, *paper_title*, *abstract*, *num_pages*, *doi* and *link*. In adittion, Paper maintains the following the following relationships with other classes:
   - **has_written:** relations Writes to a Paper.
   - **reviews:** connects a Review with a certain Paper.
   - **has_keyword:** relations Paper to Keyword(s).
   - **presented_in:** connects a Paper to the Proceeding in which was presented.
   - **in_volume:** connects a Paper to the volume in which was published.
2. **Person:** represents the roles a person can have in the ontology. This class has three subclasses: *Author*, *Reviewer* and *Chair*. Among its properties, there is *person_id*, *person_name*, *email*, *department* and *institution*. In addition, Person has the following relationships:
   - **is_chair:** connects Venue with the Chair of the venue.
   - **reviewed_by:** connects a Review with its Reviewer.
   - **written_by:** connects the node Writes with the Author of a paper.

3. **Venue:** represent the venue to which papers are submitted to be reviewed and presented. This class has two subclasses: *Conference* and *Workshop*. Among its properties, there is *venue_id*, *venue_name*, *venue_year* and *venue_edition*. Whereas its relationships are:
   - **part_of:** connects a Proceeding to its Venue.
   - **is_chair:** connects the Venue with the Chair of the venue.
4. **Proceeding:** refers to Proceeding in which a Paper is presented. Among its properties there is *proceeding_id*, *proceeding_name* and *city*. Proceeding has the following relationships:
   - **presented_in:** connects Paper with the Proceeding in which was presented.
   - **part_of:** connects Proceeding with its respective Venue.
5. **Keyword:** stands for the area of research of a given Paper. Its properties are *keyword_id*, *keyword_name* and *domain*. The relationships of Keyword are:
   - **has_keyword:** connects Paper with Keyword.
6. **Volume:** represent the volume in which a paper is published. Among its properties, there is: *volume_number* and *publishing_year*. This class has the following relationships:
   - **in_volume:** connects Paper with the Volume in which it was published.
   - **in_journal:** connects Volume with the Journal to which belongs.
7. **Journal:** represents the journal in which a certain paper is published. Among its properties, there is: *journal_id* and *journal_name*. Journal has the following properties:
   - **in_journal:** connects Volume with Journal.
8. **Writes:** represents the instance where is encoded if an author is the corresponding author of a paper. Its only property is *is_corresponding_author*. The relationships for Writes are:
   - **written_by:** connects Writes with Author.
   - **has_written:** connects Writes with Paper.
9. **Reviews:** represent a specific review of a paper. Among its properties, there is: *comment* and *acceptanceProbability*. Reviews has the following relationships:
   - **reviewed_by:** connects Review with Reviewer.
   - **reviews:** connects Review with Paper.

Notice that after defining all the classes, subclasses, relationships and properties of the model. The graph was serialized and saved in a file *"tbox.ttl"*.

In addition, it is important to mention that RDF inheritage has to be strictly defined. That is, if a class has a certain property, its subclass defined using *"rdfs:subClassOf"* will not inherit this property unless a subproperty is properly defined using *"rdfs:subPropertyOf"*. The reason for which RDF does not inherit properties automatically primarily is because of its foundational design principles and goals, which emphasize flexibility and decentralized data modeling. Hence, two possible paths can be followed:

1. Define all the subproperties for every subclass.

2. Connect directly with the element of the superclass when having to connect with a certain element that belongs to a subclass.

Since our model has few classes with many properties, we have considered the second option as the right path to follow as a way of keeping the model as simple as possible. However, this will have some consequences on the queries that we will see later.

3 ABOX Definition

In this section, the data was parsed to the *TBOX* previously defined. The code for this part can be found in the section *"B2. ABOX Definition"* from the jupyter notebook[3]. First, the data was cleaned and loaded into Pandas' dataframes. Secondly, connections where created with the different elements in classes, subclasses and relationships. Subsequently, all the information on the dataframes was uploaded in their respective fields. By using this approach, the link between TBOX and ABOX is automatically created. Finally, the graph was serialized and saved in the file *"abox.ttl"* in an analogously way as in the previous section.

4 Create Final Ontology

Once the *TBOX* and *ABOX* are defined, they can be imported in GraphDB. The class hierarchy for this model can be found in Fig.(2). That is, a representation of the data breaking it into multiple entities from higher class (super class) to lower levels (subclass).

Moreover, the relations created between the different classes defined are presented in Fig.(3).

Figure 2: Representation of the class hierarchy.

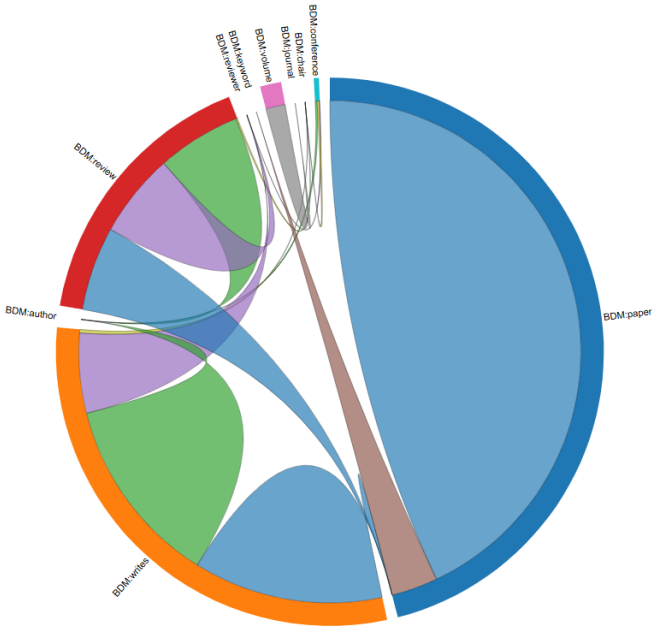

Figure 3: Representation of the class relations.

5 Querying the Ontology

5.1 Query 1

Statement: Find all Authors.

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

SELECT DISTINCT ?id ?name ?email ?department ?institution WHERE {
    ?w LAB:written_by ?a .
    ?w LAB:has_written ?p .
    ?a LAB:person_name ?name .
    ?a rdf:type LAB:author .
    ?a LAB:person_id ?id .
    ?a LAB:email ?email .
    ?a LAB:department ?department .
    ?a LAB:institution ?institution
}
```

Listing 1: First query

In Fig.(4), the results of the query are shown. Notice that not all the authors are shown since there are so many of them.

	id	name	email	department	institution
1	"101251726"	"Michael J. Franklin"	"Shari.Wade@wagner-taylor.net"	"Regular school"	"University of Phoenix-Northern Virginia Campus"
2	"101467742"	"Chao Huang"	"Victoria.Gilbert@whitehead-fischer.com"	"Regular school"	"Tooele Applied Technology College"
3	"102394267"	"Fu Lu"	"Beverly.Boyer@perez-burton.com"	"Regular school"	"Hairmasters Institute of Cosmetology"
4	"103899763"	"Julia Rudnitckaia"	"Paul.Figueroa@clark-arnold.com"	"Regular school"	"Empire Beauty School-West Greensboro"
5	"10661386"	"L. Gladkov"	"Whitney.Crawford@may.com"	"Regular school"	"SUNY College of Technology at Delhi"
6	"10674193"	"Lerato E. Magosi"	"Michelle.Gilmore@moore.com"	"Regular school"	"Dongguk University-Los Angeles"

Figure 4: Some results of the first query.

5.2 Query 2

Statement: Find all properties whose domain is Author.

```
PREFIX LAB: <http://www.example.edu/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

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

SELECT DISTINCT ?property WHERE {
    {
      ?property rdfs:domain LAB:author .
    }
    UNION
    {
      LAB:author rdfs:subClassOf* ?superclass .
      ?property rdfs:domain ?superclass .
    }
}
```

Listing 2: Second query

The properties of the class Author are shown in Fig.(5). Let us notice that in this query, we are making the union of the class Author with its superclass Person. The reason behind this is that as the model is raised, the class Author does not have any properties. This happens because all the properties are accessed using the class Person. Therefore, in our model all the properties of a class are its properties and the properties of its superclass. This are the concequences mentioned in the section where the TBOX was defined.

	property
1	LAB:department
2	LAB:email
3	LAB:institution
4	LAB:person_id
5	LAB:person_name

Figure 5: Results of the second query.

5.3 Query 3

Statement: Find all properties whose domain is either Conference or Journal.

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

SELECT DISTINCT ?property WHERE {
    {
      ?property rdfs:domain LAB:conference .
    }
    UNION
```

```
    {
      LAB:conference rdfs:subClassOf* ?superclass .
      ?property rdfs:domain ?superclass .
    }
    UNION
    {
      ?property rdfs:domain LAB:journal .
    }
}
```

Listing 3: Third query

In Fig.(6), all the properties from Conference and Journal are shown. See that since Conference is a subclass from Venue, the properties for Conference are their own plus the ones of its superclass Venue. The reason is analogous to the previous query.

	property
1	LAB:part_of
2	LAB:venue_edition
3	LAB:venue_id
4	LAB:venue_name
5	LAB:venue_year
6	LAB:journal_id
7	LAB:journal_name

Figure 6: Results of the third query.

5.4 Query 4

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

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

SELECT DISTINCT ?p ?title WHERE {
    ?p LAB:has_keyword ?k .
    ?k LAB:keyword_name "Big Data" .
    ?w LAB:has_written ?p .
    ?w LAB:written_by ?a .
    ?a LAB:person_name "A. Bateman" .
    ?p LAB:paper_title ?title .
}
```

Listing 4: Fourth query

For this last query, we filtered the papers on the keyword "Big Data" since our dataset do not have papers with the keyword "Database". In addition, we chose an arbitrary author, "A. Bateman". The result from this query can be found in Fig.(7).

	p	title
1	LAB:paper/0f5c63182b5d40850c741888a89e6c055a3593af	"The Pfam protein families database: towards a more sustainable future"
2	LAB:paper/26c075104d0ea1177cce4bd2d5c5d9eef93b8a3b	"The MEROPS database of proteolytic enzymes, their substrates and inhibitors in 2017 and a comparison with peptidases in the PANTHER database"
3	LAB:paper/317325439a0ce543d7629848a35adea04b6e7d12	"The InterPro protein families and domains database: 20 years on"
4	LAB:paper/788b43b7c62b497cf69b31544c6f81c6f4856d42	"Pfam: the protein families database"
5	LAB:paper/7f19972754ac0c15329666b3a6efbf569b27d8d5	"The Pfam protein families database in 2019"
6	LAB:paper/d6a5a1e8f56260608d2f7651a2f6aac6a041b57a	"The Pfam protein families database"
7	LAB:paper/f3c56ba45b39ec915477a19779cfc5090be09a73	"The Pfam protein families database"

Figure 7: Results of the fourth query.

References

[1] Learn-SQL v2: Learning Environment for Automatic Rating Notions of SQL: Inicia sessió en aquest lloc. (s. f.). https://learnsql2.fib.upc.edu/moodle/pluginfile.php/5688/mod_resource/content/8/knowledge-graphs.pdf

[2] Semantic Scholar | AI-Powered Research Tool. (s. f.). https://www.semanticscholar.org/

[3] Nilsvanesostos. (s. f.-a). GitHub - Nilsvanesostos/SDM—LAB2-Knowledge-Graphs. GitHub. https://github.com/Nilsvanesostos/SDM---LAB2-Knowledge-Graphs/tree/main

[4] Nilsvanesostos. (s. f.-a). GitHub - Nilsvanesostos/SDM—LAB1-Property-Graphs. GitHub. https://github.com/Nilsvanesostos/SDM---LAB1-Property-Graphs