FIB

Knowledge Graphs

Semantic Data Management

Authors:
Prashant Gupta
Nikola Ivanović

supervised by:
Oscar Romero Moral
Javier Flores Herrera

2023

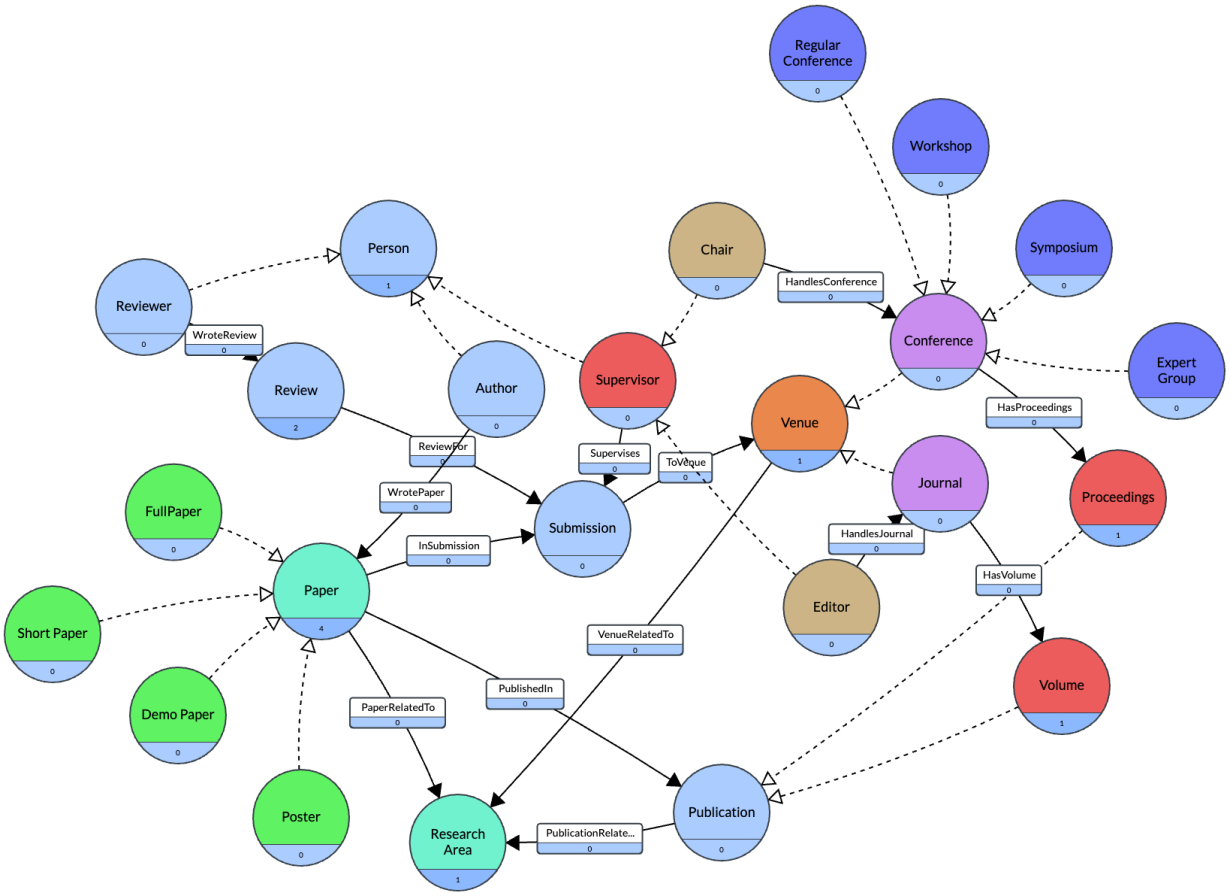
Instruction to run the code

To run the code for the project, follow these instructions:

1. Preprocessing (Python):
   - Open the provided Python notebook in Jupyter Notebook or any compatible environment.
   - Ensure that you have all the required dependencies installed (e.g., Pandas library).
   - Execute the notebook cells sequentially to run the code for processing 2 CSV files into final data.
   - Once executed, the code will generate 2 csv files. One for Journals and other fro Conference

2. Java Code (Apache Jena):
   - Open the project folder in IntelliJ or Eclipse IDE.
   - Ensure that you have Maven installed and configured in your IDE.
   - Build the project using Maven to resolve the project dependencies and download external libraries.
   - After building the project successfully, locate the main file that creates the ABOX. TBOX is already created using gra.fo.
   - Run the main file, either from the IDE's Run configuration or by executing the generated JAR file.
   - The code will use Apache Jena API to use the TBOX (schema) in RDF format and build ABOX (data) files.
   - the ABOX files will be exported in Turtle format.

B.1 TBOX Creation



Graphical Version of the TBox Can be found here

We have defined the TBox with the help of Gra.fo. After creation of ontology we have exported it in the form of TTL(Terse RDF Triple Language). As gra.fo doesn't support the property range type.So, it is manually added, wherever it is needed. The complete file for ttl can be found on github.

We have created a total 22 classes to support our hypothesis in which few of our super classes of other classes. Below our the given details:

1. **Paper:** Parent of “Short Paper”, “Demo Paper”, “Poster” and “Full Paper”. This “Poster” can only be presented in the conference and not in Journals. This restriction is programmed on the ABox level.

2. **Person:** Acts as a parent for “Author”, “Reviewer”, “Supervisor”. We have added the support for multiple authors for any paper, 2 reviewer for each submission, multiple chairs for Conference and same for Journal with the help of editors.
3. **Venue:** Consists of Conferences and Journals.
4. **Publication:** Compose of Proceedings and Volume which relates to Conference and Journals respectively.
5. **Supervisor:** Parents for Chair and Editor which overseas conference and Journals respectively.
6. **Conference:** Parent of “Regular Conference”, “Workshop”, “Symposium” and “Expert Group”.

We decided to save the decisions (accepted/rejected) and text by both reviewers' attributes. That means that for a given submission S, reviewer A and reviewer B will have their decisions and context, based on which we considered our paper as publication (if accepted). If it is accepted, it can be treated as PublishedIn in the proceedings or volume depending on the submission. If it is not accepted, it can be submitted again, that’s why review is related to submission and not to paper. Each Venue’s supervisor will assign reviewers to a submission and submissions can occur as many times as needed depending on the author and until the paper is accepted.

Finally, we have added RelatedArea as a class, it can be connected to multiple classes. Such as Paper, Venue and Publication. We have allowed many-to-many relations between them and make sure that publication’s related area will be part of Venue’s related area. As it makes more sense to do so.

B2. ABOX Creation

To construct the ABOX, we utilized the dblp dataset obtained from https://dblp.uni-trier.de/ to populate the majority of the graph elements. Nonetheless, certain entities such as keywords, reviews, and supervisors were either generated synthetically or chosen randomly since they were not available in the dataset. The conversion of dblp data from XML to CSV format was accomplished using the tool found at https://github.com/ThomHurks/dblp-to-csv. It's worth noting that due to the project's nature as an exercise, we loaded only a subset of the complete dataset.

We used two obtained csv files containing articles (papers published in journals) and inproceedings (papers published in conferences), with each row representing one research paper. Preprocessing the files was performed in python and it roughly consists of the following steps:

- Collect all loaded authors
- Randomly assign exactly two authors as reviewers to each paper (row in the dataset), making sure not to assign the author of the paper
- Randomly assign between three and five venue supervisors to each venue, then join with the data by venue
- Randomly assign between three and five keywords to each paper, publication and venue following the previous method
- Randomly generate two review decisions and texts (using the lorem library) for each paper
- Randomly assign a single author to every paper as submission supervisor
- Randomly assign paper and conference types. When performing this, we made sure to assign the type “Poster” only to papers published in conferences.

The ABOX creation process was carried out using the Apache Jena API. To start, we loaded the saved ontology model (TBOX). This allowed us to retrieve all the classes and properties defined in the model. Subsequently, we read and parsed the CSV files containing the preprocessed publication data, created instances and lined them with the TBOX ontology model. Most of the IRIs were created by applying `URLEncoder.encode` on the corresponding attributes in the preprocessed data. Since the preprocessing was very thorough, this step was quite straightforward. The only logic present in this step is creating the `publishedin` property for papers that had at least one positive review. Once all the nodes and properties were set up, we outputted the ABOX model as an .nt file using the “N-TRIPLE” language format.

B3. Creating the Final Ontology

The linking between TBOX and ABOX via *rdf:type* was performed as part of ABOX creation, as mentioned in B2. This was accomplished by utilizing the same ontology model, which was saved after TBOX creation.

We employed the "RDFS (Optimized)" ruleset for inference while loading our ontology in GraphDB. To demonstrate the capabilities of RDF, we utilized *rdfs:subClassOf*, *rdfs:domain* and *rdfs:range* to infer the *rdf:type* of resources wherever applicable. Thus, the only explicit *rdf:type* links we created were for paper types and conference types, since there are no properties to infer their type from. All other types were inferred.

Base URI: http://www.bdma.sdm#

Class Object Count

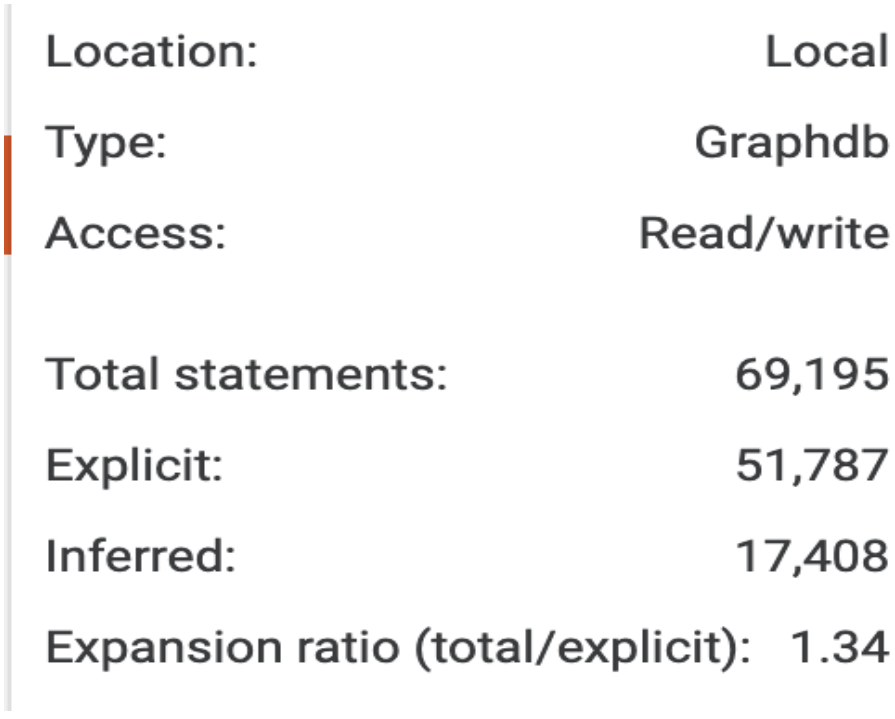
Class	Instances
Chair	79
Conference	21
Editor	72
Journal	19
Proceedings	31
Volume	580
Paper	1979
Submission	1989
ResearchArea	20
Publication	611
Review	4000
Supervisor	150
Venue	40
Author	3163
Reviewer	1242
Person	3164
DemoPaper	566
ExpertGroup	6
FullPaper	576
Poster	254
RegularConference	4
ShortPaper	590
Symposium	4

Workshop	7

Object Property Count

Property	RelationCount
handlesconference	79
handlesjournal	75
hasproceedings	31
hasvolume	580
insubmission	1989
paperrelatedto	6007
publicationrelatedto	2478
publishedin	1508
reviewfor	4000
supervises	1997
tovenue	1989
venuerelatedto	154
wrotepaper	4878
wrotereview	4000

The total statements were 691,195 out of which 17,408 statements were inferred by the above inference regime entailment.



B4. Querying the Ontology

1. Find all Authors.

The given query selects the `?fullName` values of all instances of the `sdm:Author` class in the RDF graph. The results will contain the full names of all the authors in the data.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX sdm: <http://www.bdma.sdm#>
SELECT ?fullName WHERE {
  ?s rdf:type sdm:Author ;
     sdm:fullName ?fullName .
}
```

Result:
https://raw.githubusercontent.com/prashant3167/knowledgegraph/main/results/result1.csv

2. Find all properties whose domain is Author.

The given query retrieves the properties (`?property`) whose domain includes the class `sdm:Author` or any of its superclasses. This is necessary to retrieve the properties of the sdm:Person class as well as the properties of `sdm:Author`.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX sdm: <http://www.bdma.sdm#>

SELECT ?property WHERE {
     sdm:Author rdfs:subClassOf* ?superclass .
     ?property rdfs:domain ?superclass
}
```

Result:
https://raw.githubusercontent.com/prashant3167/knowledgegraph/main/results/result2.csv

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

The given query retrieves the properties (`?property`) whose domain includes the `sdm:Venue` class or any of its subclasses. Thus, the result will include all properties whose domain is sdm:Venue, sdm:Conference or sdm:Journal.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX sdm: <http://www.bdma.sdm#>

SELECT ?property WHERE {
     ?subclass rdfs:subClassOf* sdm:Venue .
     ?property rdfs:domain ?subclass
}
```

Result:
https://raw.githubusercontent.com/prashant3167/knowledgegraph/main/results/result3.csv

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

The following query retrieves the titles (`?title`) of papers written by "Albert Bifet" and published in conference proceedings related to the research area "Big data". The results will include the titles of the relevant papers that meet the specified criteria.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX sdm: <http://www.bdma.sdm#>
SELECT ?title WHERE {
    ?paper sdm:publishedin ?proceedings ;
           sdm:paper_title ?title .
    ?conference sdm:hasproceedings ?proceedings .
    ?conference sdm:venuerelatedto ?researcharea .
    ?researcharea sdm:topic "database" .
    ?author sdm:wrotepaper ?paper ;
            sdm:fullName "Albert Bifet" .
}
```

Result:
https://raw.githubusercontent.com/prashant3167/knowledgegraph/main/results/result4.csv