

Graph-based analytics

Jakub Kwiatkowski, José Lozano Dibildox & Nils van Es Ostos

Spring 2024

Introduction

This report corresponds to the graph-based solution used for the development of the BDMA joint project. Our DBMA joint project consists in the creation of *Degree4Free*, a website where students can visit to learn about programs' and scholarships' opportunities in Spain. To do this, a knowledge graph was developed in GraphDB. Hence, the knowledge graph is the tool that will allow us to connect all the data and show the results in a functional website. All the data and code used is available in the folder *KnowledgeGraph* in the GitHub repository[2]; more specifically, in the file *KG.pynb* inside the folder *KnowledgeGraph*.

Additionally, all the data that was used stems from different sources. The information regarding programs, masters and ministry scholarship comes from the *Spanish Ministry of Science, Innovation and Universities*[4]; the data from Erasmus Mundus Programs was scrapped from the *European Education Portal*[3] and the rest of the scholarships come from the *International Scholarships* web repository[5].

Disclaimer: As discussed in the reunion, the reports for both subgroups of the joint project are identical.

1 Preprocess

Although it is not part of these project, we will briefly discuss the preprocess undergone.

1. First of all, all the data was scrapped from the sources previously mentioned.
2. The data was cleaned and treated in Spark. In this stage, some missing information was randomly generated.
3. The data was enriched by using some *Machine Learning* algorithms. Making use of them, properties like feedback(classification of reviews between good and bad ones), topic (recognition of the topic of the reviews) and recommendations were inferred.

4. Finally, all the data was saved in different csv files for the sake of simplicity when loading the data in the ontology.

2 TBOX Definition

In this section, the model for the ontology has been created. The schema of the model can be found in Fig.(1). In addition, the code for this part can be found in the section "*TBOX Definition*" from the jupyter notebook[2].

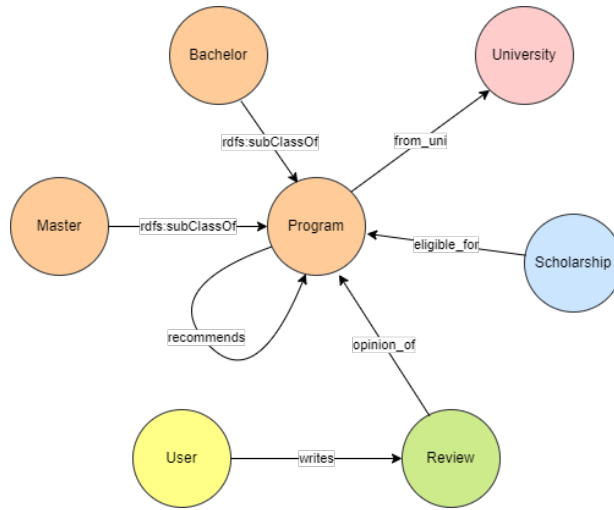


Figure 1: Schema of the Ontology.

The model can be summerized with the following points:

1. **Program:** represents the programs of the database. It is constituted by sub-classes: Bachelor and Master. Among its properties there is: program_ID, title, description, url, duration, ap_date, st_date, score, eng_lvl, spa_lvl and bachelor_needed. Additionally, this class presents the following relationships:
 - **recommends:** represents the recommendation feature. Hence, related Program with Program.
 - **from_uni:** connects a Program with the University where it is taught.
 - **eligible_for:** connects a Scholarship with the Program(s) for which is it eligible.
 - **opinion_of:** connects a certain Review with its respective Program.
2. **University:** refers to the distinct universities on the graph. Among its properties there is: uni_ID, name and location. This class only presents one relationship:

- **from_uni:** connects Program with University.
3. **Scholarship:** refers to the scholarships available for each program. Among its properties there is: `sch_ID`, `award_name`, `deadline`, `score`, `eng_lvl`, `spa_lvl`, `bachelor_needed`, `nationality`, `amount`, `num_awards`, `min_age`, `max_age` and `academic_perf`. This class presents only one relationship as well:
- **eligible_for:** connects Scholarship with Program.
4. **Review:** refers to the different Comment(s) on every Program. Among its properties there is: `review_ID`, `date`, `comment`, `feedback`, `topic`. This class presents the following relationships:
- **opinion_of:** connects Review with Program.
 - **writes:** connects a User with the Review that wrote.
5. **User:** refers to the different users on the ontology. Among its properties there is: `user_ID`, `name`, `email`, `register_date`. This class only has one relationship:
- **writes:** connects User with a Review.

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

3 ABOX Definition

In this section, the data already preprocessed was loaded in the *TBOX* previously defined. The code for this part can be found in the section *"ABOX Definition"* from the jupyter notebook[2]. First, the data was transformed from the Spark's Dataframes to Pandas' Dataframes. Secondly, connections were created with the different elements in classes, subclasses and relationships. Subsequently, all the information on the dataframes was uploaded in their respective fields. Notice that in the pre-processing, all the data was already stored in dataframes in such a way that once the connection to a certain element is made, adding all its properties is straight forward.

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).

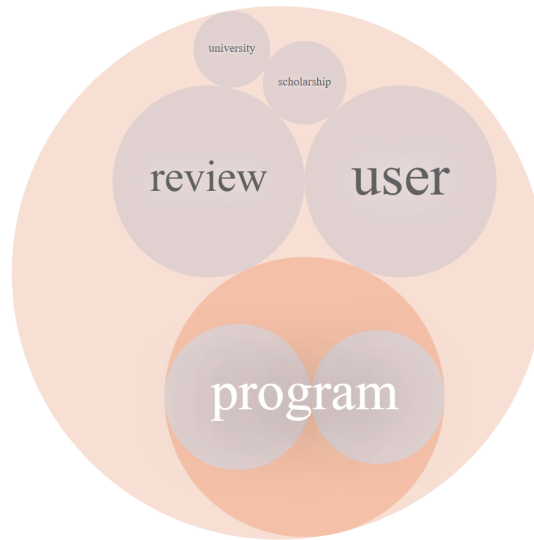


Figure 2: Representation of the class hierarchy.

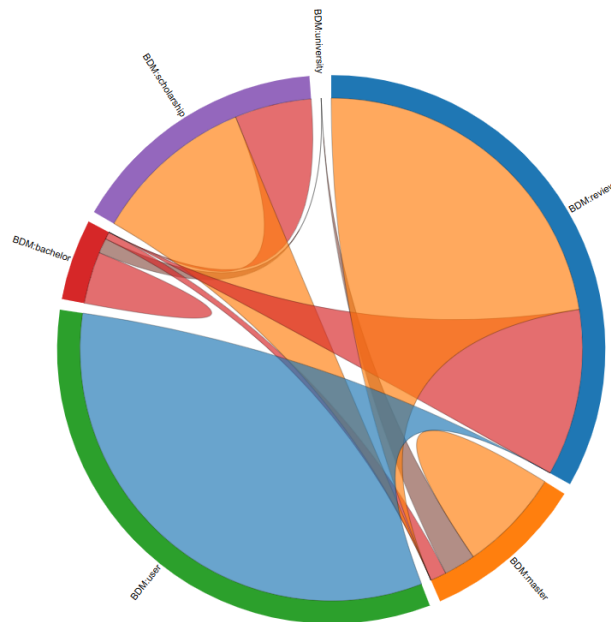


Figure 3: Representation of the class relations.

The distinct amount of instances in the graph can be found in Tab.(1). In addition, the count of distinct instances for each class are shown in Tab.(2).

| Distinct Instances | Count |
|--------------------|---------|
| Triples | 1825226 |
| Subjects | 253472 |
| Predicates | 43 |
| Objects | 477607 |

Table 1: Count of Distinct Instances

| Instances Per Class | Count |
|---------------------|--------|
| Program | 7984 |
| Master | 7446 |
| University | 105 |
| Scholarship | 160 |
| Review | 121075 |
| User | 121075 |

Table 2: Count of Instance Per Class

5 Community Detection

An interesting feature that graphs allow is the identification of communities within a network where nodes are more densely connected to each other than to the rest of the network. This allows us to quickly spot clusters with a certain connection. In this specific case, we used the *Louvain Algorithm* to create communities based on the relation between Program and University.

Notice that this same algorithm applied to certain features in the graph could contribute on exploring the relation of the data more deeply. Some examples like identifying groups of universities with similar scholarship offerings or finding influential reviewers are some ideas that could be tackled.

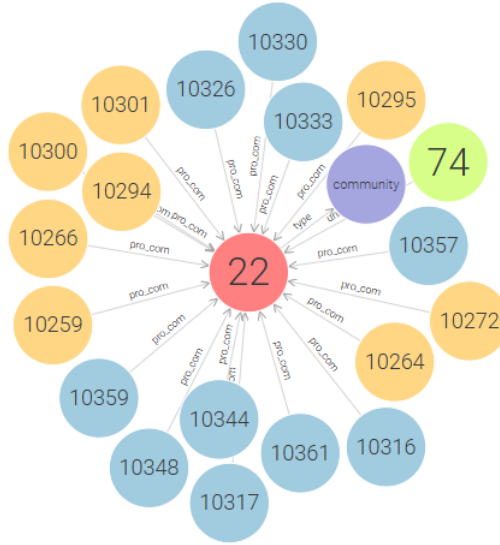


Figure 4: Example of the community of University Alfonso X El Sabio.

In Fig(4), an example of how one of these communities looks is presented. For this specific case, node 22 represent the community itself, node 74 represents the university Alfonso X El Sabio, and all the other nodes make reference to all the programs that are lectured in this university.

6 Why graph databases?

Given the context of relationships between the different entities and the constant need to retrieve information for one specific instance, graph databases are a more suitable technology. As we will see in the next section, querying information about a program (applicable scholarships, from a certain university, etc.) can be done with one traversal, starting from the program node. In a relational context this would require multiple "join" operations which can become complex and slow when scaling the environment and adding more relationships.

7 Querying the Ontology

The querying and use of the graph itself was built around the desired user experience in the *Degree4Free* website. Queries were useful for many different aspects of the website:

7.1 Population of user filters

The main page has two filters: *Program* and *University*.

Program filter:

```
PREFIX BDM: <http://www.example.edu/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT DISTINCT ?programTitle WHERE {
    ?program rdf:type BDM:program .
    ?program BDM:title ?programTitle .
}
ORDER BY ?programTitle
```

University filter:

```
PREFIX BDM: <http://www.example.edu/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT DISTINCT ?universityName WHERE {
    ?university rdf:type BDM:university .
    ?university BDM:name ?universityName .
}
ORDER BY ?universityName
```

For the users to be able to choose a program from a given university we must be able to show the available entries given our graph database. The queries are simple as they only use the *Title* and *Name* attributes respectively to show each option (see Figure 5).

7.2 Main table with programs and universities

A given user may be interested, for example, in the programs offered by the "Barcelona Institute of International Studies", so the accurate information must be shown, given the prior selection of the filters. This requires the use of a SPARQL query template that uses variables to populate it:

Main table data display:

```
PREFIX BDM: <http://www.example.edu/>
SELECT ?program ?programTitle ?universityName WHERE {{
    ?program rdf:type BDM:program .
    ?program BDM:title ?programTitle .
    ?program BDM:from_uni ?university .
    ?university BDM:name ?universityName .
    {f'FILTER (CONTAINS(LCASE(STR(?programTitle)), LCASE("{program_filter}")
    ))}' if program_filter else ''}
    {f'FILTER (CONTAINS(LCASE(STR(?universityName)), LCASE("{
university_filter}")'))}' if university_filter else ''}
```

```
}}  
GROUP BY ?program ?programTitle ?universityName  
ORDER BY ?programTitle
```

In this case, *program_filter* and *university_filter* hold the previously selected filter option so that only the relevant information is retrieved (see Figure 6).

7.3 Retrieval of node-specific attributes

An important feature from the user experience is that they are able to retrieve additional information about the program they are interested in:

- Details of the program
- Requirements and restrictions
- Applicable scholarships
- Similar or recommended programs
- Reviews other students have after studying in said program
- Contact information of students posting reviews

To be able to populate the webpage template with this information, a lot of queries with optional values had to be created for every program:

Retrieving the program details:

```
PREFIX BDM: <http://www.example.edu/>  
SELECT ?description ?bachelor_needed ?duration ?eng_lvl ?spa_lvl ?score ?  
st_date WHERE {{  
  ?program rdf:type BDM:program .  
  ?program BDM:title "{program_name}" .  
  OPTIONAL {{ ?program BDM:description ?description . }}  
  OPTIONAL {{ ?program BDM:bachelor_needed ?bachelor_needed . }}  
  OPTIONAL {{ ?program BDM:duration ?duration . }}  
  OPTIONAL {{ ?program BDM:eng_lvl ?eng_lvl . }}  
  OPTIONAL {{ ?program BDM:spa_lvl ?spa_lvl . }}  
  OPTIONAL {{ ?program BDM:score ?score . }}  
  OPTIONAL {{ ?program BDM:st_date ?st_date . }}  
}}
```

Retrieving applicable scholarships:

```
PREFIX BDM: <http://www.example.edu/>  
SELECT ?scholarshipTitle ?amount WHERE {{  
  ?program BDM:title "{program_name}" .
```



```

?scholarship BDM:eligible_for ?program .
?scholarship BDM:award_name ?scholarshipTitle .
?scholarship BDM:amount ?amount .
}}
```

Retrieving recommended programs:

```

PREFIX BDM: <http://www.example.edu/>
SELECT ?title
WHERE {{
    ?program BDM:title "{program_name}" .
    ?program BDM:recommends ?recommended_program .
    ?recommended_program BDM:title ?title
}}
```

Retrieving opinions of the program and user's data:

```

PREFIX BDM: <http://www.example.edu/>
SELECT ?comment ?date ?feedback ?topic ?user ?name ?email ?register_date
WHERE {{
    ?review BDM:opinion_of ?program .
    ?program BDM:title "{program_name}" .
    OPTIONAL {{ ?review BDM:comment ?comment . }}
    OPTIONAL {{ ?review BDM:date ?date . }}
    OPTIONAL {{ ?review BDM:feedback ?feedback . }}
    OPTIONAL {{ ?review BDM:topic ?topic . }}
    OPTIONAL {{
        ?user BDM:writes ?review .
        ?user BDM:name ?name ;
            BDM:email ?email ;
            BDM:register_date ?register_date .
    }}
}}
```

To visualize the different queries on program information see Figures 7 and 8

Finally, to complete the user experience we wanted them to be able to see the characteristics of the applicable scholarships (see Figure 9), considering again details and requirements or restrictions:

Retrieving the scholarship details:

```

PREFIX BDM: <http://www.example.edu/>
SELECT ?award_name ?nationality ?academic_perf ?amount ?num_awards ?score ?
spa_lvl ?eng_lvl ?bachelor_needed ?max_age ?min_age ?deadline
WHERE {{
    ?scholarship BDM:award_name "{scholarship_name}" .
    OPTIONAL {{ ?scholarship BDM:award_name ?award_name . }}
    OPTIONAL {{ ?scholarship BDM:nationality ?nationality . }}
    OPTIONAL {{ ?scholarship BDM:academic_perf ?academic_perf . }}
}}
```

```
OPTIONAL {{ ?scholarship BDM:amount ?amount . }}
OPTIONAL {{ ?scholarship BDM:num_awards ?num_awards . }}
OPTIONAL {{ ?scholarship BDM:score ?score . }}
OPTIONAL {{ ?scholarship BDM:spa_lvl ?spa_lvl . }}
OPTIONAL {{ ?scholarship BDM:eng_lvl ?eng_lvl . }}
OPTIONAL {{ ?scholarship BDM:bachelor_needed ?bachelor_needed . }}
OPTIONAL {{ ?scholarship BDM:max_age ?max_age . }}
OPTIONAL {{ ?scholarship BDM:min_age ?min_age . }}
OPTIONAL {{ ?scholarship BDM:deadline ?deadline . }}
}}
```

References

- [1] Learn-SQL v2: Learning Environment for Automatic Rating Notions of SQL: Inicia sessió en aquest lloc. (n.d.-b). https://learnsql2.fib.upc.edu/moodle/pluginfile.php/10257/mod_resource/content/2/SDM-project-statement.pdf
- [2] Nilsvanesostos. (n.d.-a). GitHub - Nilsvanesostos/Degree4Free. GitHub. <https://github.com/Nilsvanesostos/Degree4Free>
- [3] Erasmus Mundus catalogue. (n.d.). European Education And Culture Executive Agency. https://www.eacea.ec.europa.eu/scholarships/erasmus-mundus-catalogue_en?f%5B0%5D=studies_for_emjmd_studies_for_emjmd_project%3A23&f%5B1%5D=studies_for_emjmd_studies_for_emjmd_project%3A26&page=1
- [4] Ministry of Science, Innovation and Universities. (n.d.). <https://www.universidades.gob.es/donde-estudiar/>
- [5] International Scholarships. (n.d.) <https://www.internationalscholarships.com/>

8 Appendix

Degree4Free webpage screenshots:

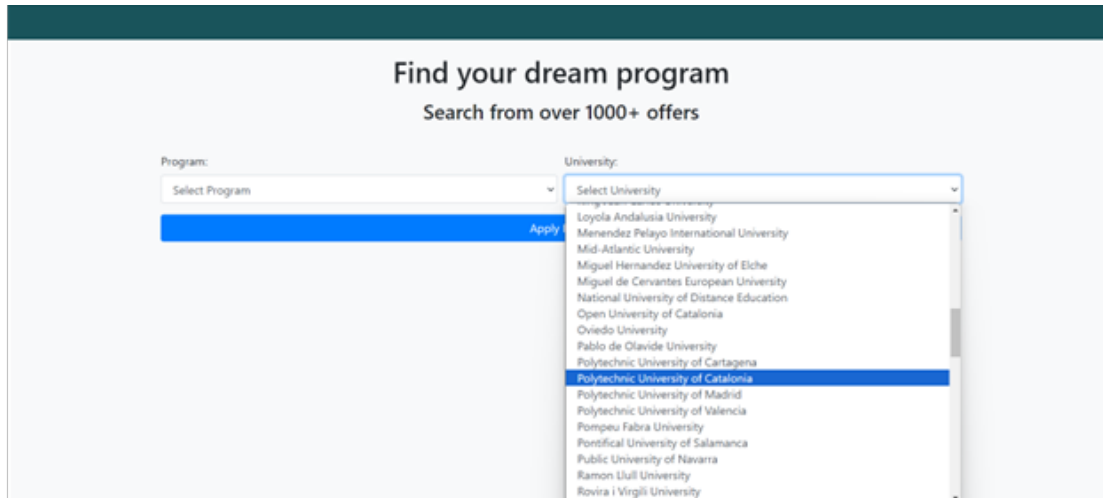
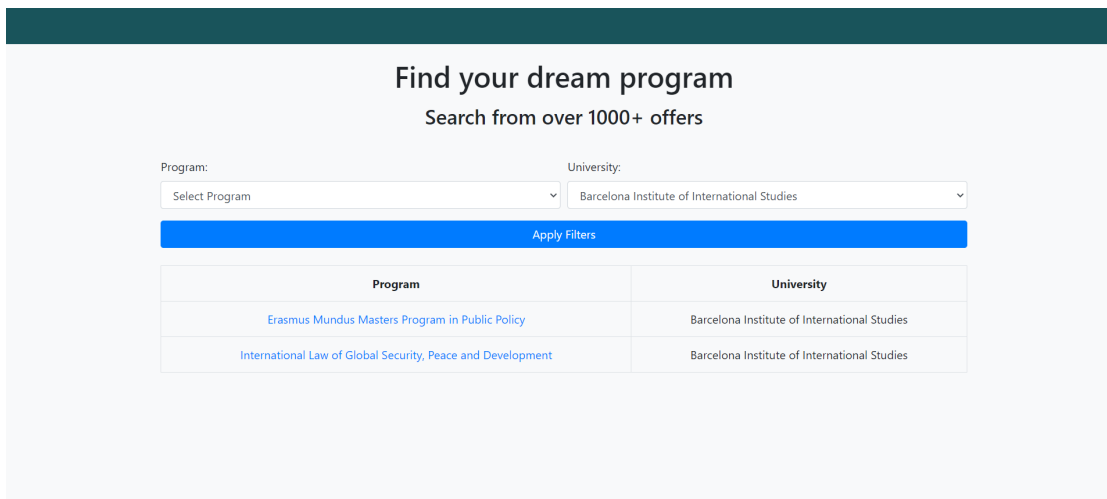


Figure 5: Visualization of filter queries in D4F webpage



| Program | University |
|---|--|
| Erasmus Mundus Masters Program in Public Policy | Barcelona Institute of International Studies |
| International Law of Global Security, Peace and Development | Barcelona Institute of International Studies |

Figure 6: Visualization of main data query in D4F webpage

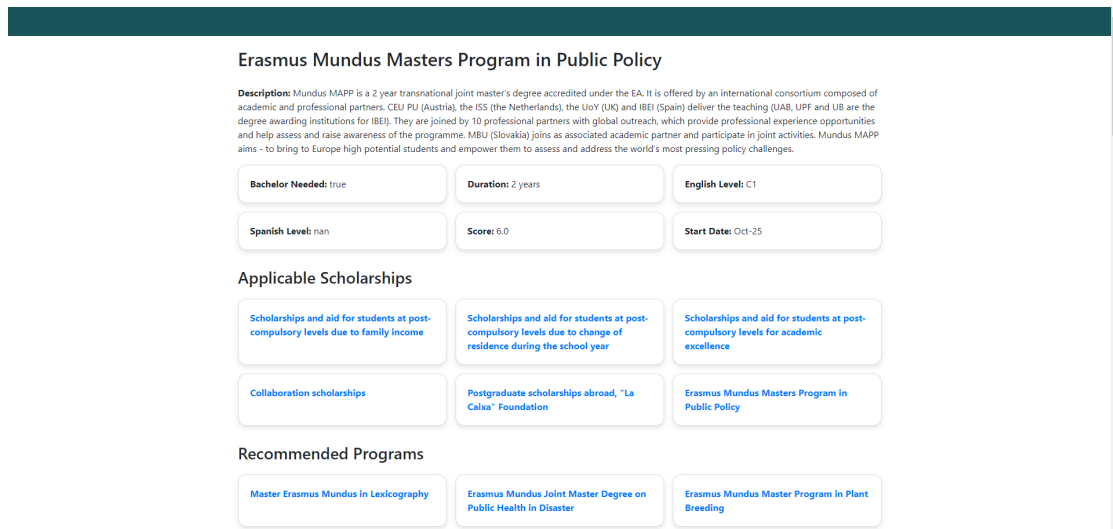


Figure 7: Visualization of program details query in D4F webpage

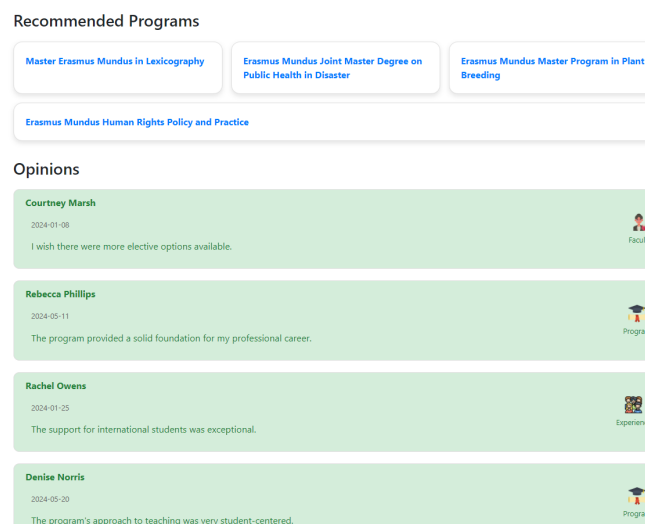


Figure 8: Visualization of program details query in D4F webpage

Scholarships and aid for students at post-compulsory levels for academic excellence

Requirements and Restrictions

| | | |
|----------------------|------------------------|---------------------------------|
| Nationality: Spanish | Amount: 125.0 | Number of Awards: Unrestricted |
| Score: 5 | Bachelor Needed: false | Deadline: 01/02/2025-15/03/2025 |

Figure 9: Visualization of scholarship details query in D4F webpage