UNIVERSITÉ Concordia UNIVERSITY

COMP 441/6741: INTELLIGENT SYSTEMS

WINTER 2024

PROJECT ASSIGNMENT #1

Instructor

Dr. René Witte

Submitted By

Saurabh Sharma

Anubhav Mahajan(40267770)

GitHub URL:

https://github.com/saurabhs679/COMP-6741-Project

We certify that this submission is the original work of members of the group and meets the Faculty's Expectations of Originality

Signature 1 ID Number:40267770 Date-22-03-2024

Signature 2 ID Number: 40226298 Date-22-03-2024

Vocabulary

The RDF schema outlines the structure for representing university-related data, containing courses, lectures, students, topics, and universities. Each class is defined, along with their respective properties for example course class with properties such as course name, student ID, lecture content, and topic name. Object properties establish relationships between entities, indicating which lectures are part of a course and which topics are covered in a course or lecture. The schema provides a structured framework for organizing and querying university data.

The knowledge base schema is represented using RDF (Resource Description Framework), with specific classes and properties developed to describe the domain's entities and relationships. Here is a summary of the modeling decisions:

Standard Vocabularies:

Standard vocabularies such as rdf, rdfs, and their associated properties (rdf:type, rdfs:label, rdfs:domain, rdfs:range) are utilized for defining classes, properties, and their characteristics

Classes and Properties:

Classes - ex:Course, ex:Lecture, ex:Student, ex:Topic, and ex:University. These classes provide a structured framework for categorizing and organizing information.

Properties - Properties explain an entity's attributes and relationships. Each property is connected with a domain and a range, which define its applicability and possible values. ex:CourseName, ex:CourseNumber, and ex:Credits are characteristics that record precise facts about courses, whereas ex:IDNumber, ex:firstName, and ex:lastName describe student-related information.

Vocabulary Extensions:

We also developed a few vocabularies extension such as ex:topic_in_course, ex:topic_in_lecture to define the relationship between different classes like first one shows relationship between course and topic . Similarly, ex:topic_in_lecture links individual lectures to the topics they address. These extensions enhance the schema relation, enabling more accurate querying and displays more comprehensive representation of the knowledge base.

This design structure lays the groundwork for efficient organization, retrieval, and integration of information.

Knowledge Base Construction

Our dataset consists of five distinct datasets, each based on specific classes and their corresponding properties within our knowledge base schema. The datasets contain structured information relevant to courses, lectures, students, topics, and universities.

Courses Dataset: Contains information about various courses offered, including course name, number, credits, description, subject, outline, and webpage link.

Lectures Dataset: Provides details about individual lectures, such as lecture name, number, content, and webpage link.

Students Dataset: Includes information about enrolled students, such as student ID, first name, last name, email, and completed courses.

Topics Dataset: Contains data regarding topics covered within lectures or courses, along with topic names and DBpedia links for additional context.

Universities Dataset: Provides information about universities, including university name and DBpedia link.

We are using python script to parse and read all the datasets and converting it into a structured format compatible with the Turtle serialization.We are doing this for all 5 datasets merging them together with main vocubalary to populate the knowledge graph.

To run the tools and create the knowledge base using the provided Python notebook, follow these steps:

Prepare CSV Files:

Ensure that you have the necessary CSV files containing the data for students, courses, universities, lectures, and topics. Each CSV file should follow a specific format with columns representing different attributes such as student ID, course name, lecture content, etc.

Open the Python Notebook:

Open the Python notebook file containing the code for populating each aspect of the knowledge base using the rdflib library.

Execute Cells:

Execute each cell in the notebook using a Python interpreter or a Jupyter notebook environment. Each cell corresponds to a section of code responsible for populating a specific aspect of the knowledge base (e.g., student data, course data, etc.).

Verify Output:

After executing each cell, verify that the output is generated successfully without any errors. The output will typically be in the form of RDF triples serialized in Turtle format, printed to the notebook's output cell.

Load RDF Triples:

Once you have generated RDF triples for each aspect of the knowledge base, load them into a triple store or graph database for storage and querying. You can use a triple store such as Apache Jena.

Execute SPARQL Queries:

Within the notebook, execute SPARQL queries directly against the populated RDF graph. Use the rdflib.plugins.sparql.prepareQuery function to prepare the query, and then execute it using the RDF graph object. For example, you can execute a SPARQL query to retrieve competencies related to a specific course as follows:

```
from rdflib.plugins.sparql import prepareQuery
query = prepareQuery(f"""
   PREFIX ex: <http://example.org/>
   SELECT ?competency
   WHERE {{
     ?course ex:completedCourse "COMP 6481" ;
          ex:competentIn ?competency .
```

```
    }}
""", initNs={"ex": ex})

# Execute the query and print the competencies
results = g1.query(query)
for row in results:
    print(row.competency)
```

By following these steps and executing the provided Python notebook, you can effectively populate the knowledge base from the CSV files and perform querying and analysis tasks.

Graph Queries

1. **List all courses offered by [university]:**
    - This query retrieves the names and numbers of all courses offered by a specific university.
    - It uses the ex:CourseName and ex:CourseNumber properties to fetch the course details.
    - The ?course variable represents each course in the graph that matches the criteria.
    - It selects the course name and course number using the SELECT clause.
    - The ?course variable is bound to instances of the ex:Course class.

Output-

```
Course Name: PPS
Course Number: 6481
Course Name: Intelligent Systems
Course Number: 6741
```

2. **In which courses is [topic] discussed?**
    - This query finds all courses where a specific topic is discussed.
    - It looks for instances of the topic with the given name using ex:TopicName.

- It then retrieves the names of courses where the topic is discussed using the ex:topic_in_course property.
- The ?courseName variable represents the names of courses where the topic is discussed.

Output-

```
PPS
```

3. **Which [topics] are covered in [course] during [lecture number]?**
    - This query retrieves the names of topics covered in a specific course during a particular lecture.
    - It selects topics associated with the given lecture number using the ex:lectureNumber property.
    - The ?topicName variable represents the names of topics covered in the specified lecture of the course.

Output-

```
Algorithm efficiency
```

4. **List all [courses] offered by [university] within the [subject] (e.g., "COMP", "SOEN"):**
    - This query fetches the names and numbers of courses offered by a university within a specific subject area.
    - It filters courses based on the subject using the ex:Subject property.
    - The ?university variable represents the university, and the ?courseName and ?courseNumber variables represent course details.

Output-

```
Course Name: PPS
Course Number: 6481
Course Name: Intelligent Systems
Course Number: 6741
```

5. **What [materials] (slides, readings) are recommended for [topic] in [course] [number]?**
    - This query fetches the study material for the topic.
    - It uses ex:TopicName and ex:topic_in_lecture to traverse the graph to fetch details to filter and output is found in ex:Content

Output-

```
Slides: aa.pdf
Worksheets:
Readings: Textbook Data Structures and Algorithms in Java, 6th edition, by M.
T. Goodrich, R. Tamassia, and M. H. Goldwasser, Wiley, 2014
```

6. How many credits is [course] [number] worth?

- This query retrieves the number of credits associated with a specific course number.
- It filters courses based on the course number using the ex:CourseNumber property.
- The ?credits variable represents the number of credits associated with the course

Output-

```
4
```

7. For [course] [number], what additional resources (links to web pages) are available?

- This query retrieves the webpages associated with a specific course number.
- It focuses on extracting additional resources specified by the predicate ex:WebpageLink

Output-

```
https://www.concordia.ca/academics/graduate/calendar/current/gina-cody-school
-of-engineering-and-computer-science-courses/computer-science-and-software-en
gineering-master-s-and-phd-courses.html
```

8. Detail the content (slides, worksheets, readings) available for [lecture number] in [course] [number].

- This query retrieves the content associated with a specific course number for the given lecture
- It uses ex:CourseNumber, ex:lectureNumber, and ex:Content to specify relationships in the RDF graph and give output

Output-

```
Slides: slides03.pdf
Worksheets: worksheet03.pdf
```

```
Readings: [Yu14, Chapter 4] (RDFS),  [Yu14, Chapter 7] (FOAF)
Slides: Recursion.pdf
Worksheets:
Readings: https://www.geeksforgeeks.org/introduction-to-recursion-data-struct
ure-and-algorithm-tutorials/
```

9. What reading materials are recommended for studying [topic] in [course]?

- This query retrieves reading materials recommended for studying a specific topic in a course.
- It filters content containing "Readings" by using predicates like ex:CourseName, ex:topic_in_lecture, and ex:Content to specify relationships in the RDF graph.

Output-

```
Slides: aa.pdf
Worksheets:
Readings: Textbook Data Structures and Algorithms in Java, 6th edition, by M.
T. Goodrich, R. Tamassia, and M. H. Goldwasser, Wiley, 2014
```

10. What competencies [topics] does a student gain after completing [course] [number]?

- This query identifies the competencies (topics) acquired by a student upon completing a specific course.
- It utilizes predicates such as ex:completedCourse and ex:competentIn to specify relationships in the RDF graph and retrieve the desired competency information.

Output-

```
Run Time Errors
Hash table
```

11. What grades did [student] achieve in [course] [number]?

- This query retrieves the grades achieved by a specific student in a particular course.
- It extracts the first name, last name, and grade of the student using predicates such as ex:firstName, ex:lastName, ex:completedCourse, and ex:grade from the RDF graph

Output-

```
Name: Anubhav Mahajan, Grade: A
```

```
Name: Saurabh Sharma, Grade: B
```

12. Which [students] have completed [course] [number]?

- This query identifies which students have completed a specific course.
- It retrieves the first name and last name of the students who completed the course identified by the course number "COMP 6481", using predicates such as ex:firstName, ex:lastName, and ex:completedCourse from the RDF graph.

Output-

```
Name: Anubhav Mahajan
Name: Saurabh Sharma
```

13. Print a transcript for a [student], listing all the course taken with their grades.

- This query generates a transcript for a specific student, listing all the courses taken along with their respective grades.
- It retrieves the completed courses and corresponding grades for the student named "Anubhav" using predicates such as rdf:type, ex:firstName, ex:completedCourse, and ex:grade from the RDF graph.

Output-

```
Transcript for Anubhav:
Completed Course: COMP 6481
Grade: A
```

Triplestore and SPARQL Endpoint Setup

We are using Apache Jena Fuseki to set up a Triplestore and SPARQL Endpoint. We are downloading the Apache Jena Fuseki distribution package and extracting it. We also need to make sure java is installed in our system and its environment path is configured. We can open it from command prompt using fuseki-server command and it runs on localhost 3030. We create a new dataset and Upload RDF data into the newly created dataset using the web interface. To access the SPARQL endpoint provided by Fuseki to query the loaded data, construct and execute SPARQL queries using the web interface.

Conclusion

The initial phase, as outlined in this report, was dedicated to lay the groundwork for our project. Through meticulous parsing of datasets and leveraging Semantic Web technologies, we successfully built a structured knowledge graph encompassing diverse educational domains.

Moving forward, our focus shifts towards refining this foundation and incorporating additional enhancements. The subsequent phase will involve further enriching the knowledge base, implementing advanced features, and developing a natural language processing interface. These endeavors aim to enhance the usability and functionality of our system, facilitating seamless access to educational resources and insights.

By integrating these components into a cohesive system, we aspire to create a robust platform. Through continuous iteration and refinement, we remain committed to pushing the boundaries of knowledge representation and discovery, empowering users with valuable insights and facilitating meaningful interactions with educational data.