Concordia University

Engineering and Computer Science

COMP 6741

Intelligent Systems

Project-1 Report

Roboprof

Submitted To: Prof. Dr. René Witte

Date : 22 March, 2024

Team **AK_G_04**

Aryan Saxena (40233170)

Benjamin Douglas (40264251)

GitHub Link

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

Table of Contents

1. Vocabulary

1.1 Vocabulary

For our project, we used some of the public and most widely used vocabularies.

Prefix	URI
rdf	*<http://www.w3.org/1999/02/22-rdf-syntax-ns#>*
rdfs	*<http://www.w3.org/2000/01/rdf-schema#>*
foaf	*<http://xmlns.com/foaf/0.1/>*
vivo	*<http://vivoweb.org/ontology/core#>*
ex	*<http://example.org/ns#>*
dbo	*<http://dbpedia.org/ontology/>*
dbr	*<http://dbpedia.org/resource/>*

Vocabulary files used for this project:

1. Courses.ttl
2. Lectures.ttl
3. Students.ttl
4. Univerisites.ttl
5. Topics.ttl

1.2 Schema

This section provides a detailed description of the different classes used for building the schema.

1. University :

```
ex:University a rdfs:Class ;
   rdfs:label "University"@en ;
       rdfs:comment "An institution of higher education and research."@en .
```

2. Course:

```
ex:Course a rdfs:Class ;
   rdfs:label "Course"@en ;
       rdfs:comment "A unit of teaching that typically lasts one academic term, is led by one or more
    instructors, and has a fixed roster of students."@en .
```

3. Lecture:

```
ex:Lecture a rdfs:Class ;
   rdfs:label "Lecture"@en ;
       rdfs:comment "A lecture is a specific instance of instruction within a course."@en .
```

4. Topic :

```
ex:Topic a rdfs:Class ;
   rdfs:label "Topic"@en ;
       rdfs:comment "A specific subject matter covered within the academic content of a course."@en .
```

5. Student:

ex:Student a rdfs:Class ;
rdfs:label "Student"@en ;
rdfs:subClassOf foaf:Person ;
rdfs:comment "An individual who is studying at a university or other place of higher education."@en

1.3 Property

1. hasName

ex:hasName a rdf:Property ;
rdfs:label "has Name"@en ;
rdfs:domain rdfs:Resource ;
rdfs:range rdfs:Literal .

2. hasLink

ex:hasLink a rdf:Property ;
rdfs:label "has Link"@en ; rdfs:domain rdfs:Resource ;
rdfs:range rdfs:Literal .

3. offersCourse

ex:offersCourse a rdf:Property ;
rdfs:label "offers Course"@en ;
rdfs:domain ex:University ;
rdfs:range ex:Course .

2. Knowledge Base Construction

2.1 Dataset

The following datasets are used for the creation of knowledge graph:

data.csv : contains all the courses details obtained
lectures.csv : contains all the lecture details with content link
student.csv : contains all the student details with necessary data
topics.csv : contains the topics present in the lectures with the topic links

2.2 Development Process

The knowledge base development process contains 4 different graphs each one for course, student, lecture and topic. The following code snippet provides the detailed manner how the knowledge base is build.

1. Course

```
catalog_df = pd.read_csv(coursecsv, encoding='utf-16')
for index, row in catalog_df.iterrows():
    course_id_str = str(row['Course ID'])
    course_uri = URIRef(f"http://example.org/vocab/course/{urllib.parse.quote(course_id_str)}")
    g.add((course_uri, RDF.type, ex.Course))
    g.add((course_uri, ex.subject, Literal(row['Subject'], datatype=XSD.string)))
    g.add((course_uri, ex.number, Literal(row['Catalog'], datatype=XSD.string)))
    g.add((course_uri, ex.description, Literal(row['Long Title'], datatype=XSD.string)))
    g.add((course_uri, ex.credits, Literal(row['Class Units'], datatype=XSD.decimal)))
    if not pd.isnull(row['Pre Requisite Description']):
        g.add((course_uri, ex.preRequisiteDescription, Literal(row['Pre Requisite Description'], datatype=XSD.string)))
    if not pd.isnull(row['Equivalent Courses']):
        g.add((course_uri, ex.equivalentCourses, Literal(row['Equivalent Courses'], datatype=XSD.string)))


g.serialize(destination=coursePath, format='turtle')
Executed at 2024.03.22 22:41:48 in 2s 922ms

 <Graph identifier=N86fd04b70bdc4a10bfac7f15624d6b2e (<class 'rdflib.graph.Graph'>)>
```

2. Student

```
students_df = pd.read_csv(studentcsv)
for index, row in students_df.iterrows():
    student_uri = ex[f'student/{urllib.parse.quote(row["id_number"])}']
    g.add((student_uri, RDF.type, ex.Student))
    g.add((student_uri, foaf.name, Literal(f"{row['first_name']} {row['last_name']}", datatype=XSD.string)))
    g.add((student_uri, foaf.mbox, Literal(row['email'], datatype=XSD.string)))
    g.add((student_uri, ex.studentID, Literal(row['id_number'], datatype=XSD.string)))

    for i in range(1, 5):
        course_uri = ex[f'course/{row[f"course_{i}_id"]}']
        grade = row[f"course_{i}_grade"]
        course_completion = BNode()
        g.add((student_uri, ex.completedCourse, course_completion))
        g.add((course_completion, RDF.type, URIRef("http://example.org/vocab/CompletedCourse")))
        g.add((course_completion, ex.course, course_uri))
        g.add((course_completion, ex.courseGrade, Literal(grade, datatype=XSD.string)))

    for comp_field in ['competent_in_course_id1', 'competent_in_course_id2']:
        competency_course_uri = ex[f'course/{row[comp_field]}']
        g.add((student_uri, ex.hasCompetency, competency_course_uri))


g.serialize(destination=studentPath, format='turtle')
```

Executed at 2024.03.22 22:41:54 in 29ms

```
<Graph identifier=Nc5248e64957f4e27ae60dc60e4f1a2e7 (<class 'rdflib.graph.Graph'>)>
```

3. Lecture

```
lectures_df = pd.read_csv(lecturescsv)
for index, row in lectures_df.iterrows():
    # Create a URI for the lecture based on its ID
    lecture_uri = URIRef(f"http://example.org/vocab/lecture/{urllib.parse.quote(str(row['LectureID']))}")

    g.add((lecture_uri, RDF.type, ex.Lecture))

    course_uri = URIRef(f"http://example.org/vocab/course/{urllib.parse.quote(str(row['CourseID']))}")
    g.add((lecture_uri, ex.isPartOfCourse, course_uri))

    g.add((lecture_uri, ex.lectureNumber, Literal(row['LectureNumber'], datatype=XSD.integer)))
    g.add((lecture_uri, ex.lectureName, Literal(row['LectureName'], datatype=XSD.string)))
    g.add((lecture_uri, ex.lectureContentLink, Literal(row['ContentLink'], datatype=XSD.anyURI)))
    g.add((lecture_uri, rdfs.seeAlso, URIRef(row['SeeAlsoLink'])))

g.serialize(destination=lecturesPath, format='turtle')
```

Executed at 2024.03.22 22:41:56 in 14ms

```
<Graph identifier=Nb8067f20e5fe4238bb1446b79e420627 (<class 'rdflib.graph.Graph'>)>
```

4. Topic

```python
topics_df = pd.read_csv(topicscsv)
for index, row in topics_df.iterrows():
    topic_uri = URIRef(f"http://example.org/vocab/topic/{urllib.parse.quote(str(row['TopicID']))}")
    g.add((topic_uri, RDF.type, ex.Topic))
    g.add((topic_uri, ex.topicName, Literal(row['TopicName'], datatype=XSD.string)))
    g.add((topic_uri, ex.topicProvenance, Literal(row['TopicProvenance'], datatype=XSD.string)))
    g.add((topic_uri, ex.topicLink, URIRef(row['TopicLink'])))
    lecture_uri = URIRef(f"http://example.org/vocab/lecture/{urllib.parse.quote(str(row['LectureID']))}")
    g.add((topic_uri, ex.isTopicOfLecture, lecture_uri))
    course_uri = URIRef(f"http://example.org/vocab/course/{urllib.parse.quote(str(row['CourseID']))}")
    g.add((topic_uri, ex.isTopicOfCourse, course_uri))

g.serialize(destination=topicsPath, format='turtle')
```

Executed at 2024.03.22 22:41:59 in 21ms

```
<Graph identifier=N7e824f5fb4714305a5c81933865a748e (<class 'rdflib.graph.Graph'>)>
```

5.

```
dbr:Bay_River_College a dbo:University ;
    rdfs:label "Bay River College"@en .

dbr:Beedie_School_of_Business a dbo:University ;
    rdfs:label "Beedie School of Business"@en .

dbr:Bishop_Feild_College a dbo:University ;
    rdfs:label "Bishop Feild College"@en .

dbr:Booth_University_College a dbo:University ;
    rdfs:label "Booth University College"@en .

dbr:Bora_Laskin_Faculty_of_Law a dbo:University ;
    rdfs:label "Bora Laskin Faculty of Law"@en .

dbr:Bow_Valley_College a dbo:University ;
    rdfs:label "Bow Valley College"@en .

dbr:Brescia_University_College a dbo:University ;
    rdfs:label "Brescia University College"@en .

dbr:Camosun_College a dbo:University ;
    rdfs:label "Camosun College"@en .

dbr:Canadian_Association_of_Research_Libraries a dbo:University ;
    rdfs:label "Canadian Association of Research Libraries"@en .

dbr:Canadian_College_of_Naturopathic_Medicine a dbo:University ;
    rdfs:label "Canadian College of Naturopathic Medicine"@en .

dbr:Canadian_Institute_for_Theoretical_Astrophysics a dbo:University ;
```

3. Graph Queries

3.1 List all courses offered by [university]

3.2 In which courses is [topic] discussed?

3.3 Which [topics] are covered in [course] during [lecture number]?

3.4 List all [courses] offered by [university] within the [subject] (e.g., “COMP”, “SOEN”).

3.5 What [materials] (slides, readings) are recommended for [topic] in [course] [number]?

3.6 How many credits is [course] [number] worth?

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

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

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

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

3.11 What grades did [student] achieve in [course] [number]?

3.12 Which [students] have completed [course] [number]?

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

Each query is stored in a corresponding .csv file named accordingly (e.g., q1.csv for query 3.1), ensuring that the results are not only well-documented but also formatted for readability and potential further analysis.