COMP 474 - Project Part 1

Xavier Morin
40077865
Gina Cody School of Engineering
Concordia University
Montreal, QC
xmorin10@gmail.com

Axel Bogos
40077502
Gina Cody School of Engineering
Concordia University
Montreal, QC
a_ogos@live.concordia.ca

Walton Jones
40026889
Gina Cody School of Engineering
Concordia University
Montreal, QC
jones.welton.4@gmail.com

Gini Brandon
40066200
Gina Cody School of Engineering
Concordia University
Montreal, QC
gini@forsaw.ca

March 30, 2021

1 Introduction

This assignment is part of the COMP 474 project. In this first part, we present the competency questions, the schema design, the external vocabularies and ontology used, our automated knowledge base population script, and finally showcase queries and results that satisfy each competency question.

2 Competency Questions

We created the following competency questions to shape the design of our agent:

1. Which topics were covered in the second lecture in COMP 474?
2. What is course COMP 474 about?
3. Which courses at Concordia teach about knowledge graphs?
4. Where can I get more information about COMP 474?
5. Is COMP 474 taught at Concordia?
6. Which universities teach courses about database design?
7. How many COMP courses are taught at Concordia university?
8. What is the course website for COMP 353?
9. Which course has the fewest readings?
10. Is Intelligent Systems a COMP or SOEN course?

These questions are meant to holistically demonstrate the extent of the information available in our final knowledge base. Information is obviously limited to our two special courses COMP 474 and COMP 353.

3 Vocabulary

Our vocabulary consists of multiple classes and properties which extend or re-use existing vocabulary wherever possible and logically sensible. We rely on the following external vocabularies and ontologies:

- RDF (`http://www.w3.org/1999/02/22-rdf-syntax-ns#`)
- RDFS (`http://www.w3.org/2000/01/rdf-schema#`)
- Owl (`http://www.w3.org/2002/07/owl#`)
- Teach (`http://linkedscience.org/teach/ns#`)
- Dbpedia resource (`http://dbpedia.org/resource/`)
- Dbpedia ontology (`http://dbpedia.org/ontology/`)
- Dbpedia property (`http://dbpedia.org/property/`)

The classes of our schema and their respective link to common vocabularies are shown in Table 1.

Table 1: Knowledge Base Classes

Class	Subclass Of	Other Ontology Link
University	org:Organization	owl:sameAs dbr:University
Course	teach:Course	X
Lecture	teach:Lecture	X
LectureRelatedEvent	X	X
Lab	LectureRelatedEvent	X
Tutorial	LectureRelatedEvent	X
LectureMaterial	teach:Material	X
Slides	LectureMaterial	X
Worksheet	LectureMaterial	X
Reading	LectureMaterial	X

This class schema aims at both utilizing the flexibility of defining our own properties and classes and the connection with more mainstream vocabularies. Figure 1 is a graph of the class included and their schema. Note that the full graph including all properties is included in the appendix as Figure 2.

Two abstract classes are used to hierarchically structure related entities and in turn define properties with a range using these abstract classes, allowing for a single property to relate together multiple classes in a one-to-many fashion. As mentioned before, these classes are `LectureMaterial` (the parent class of all classes representing the work material of a lecture) and `LectureRelatedEvent` (the parent class of all classes representing learning events related to lectures such as labs and tutorials).

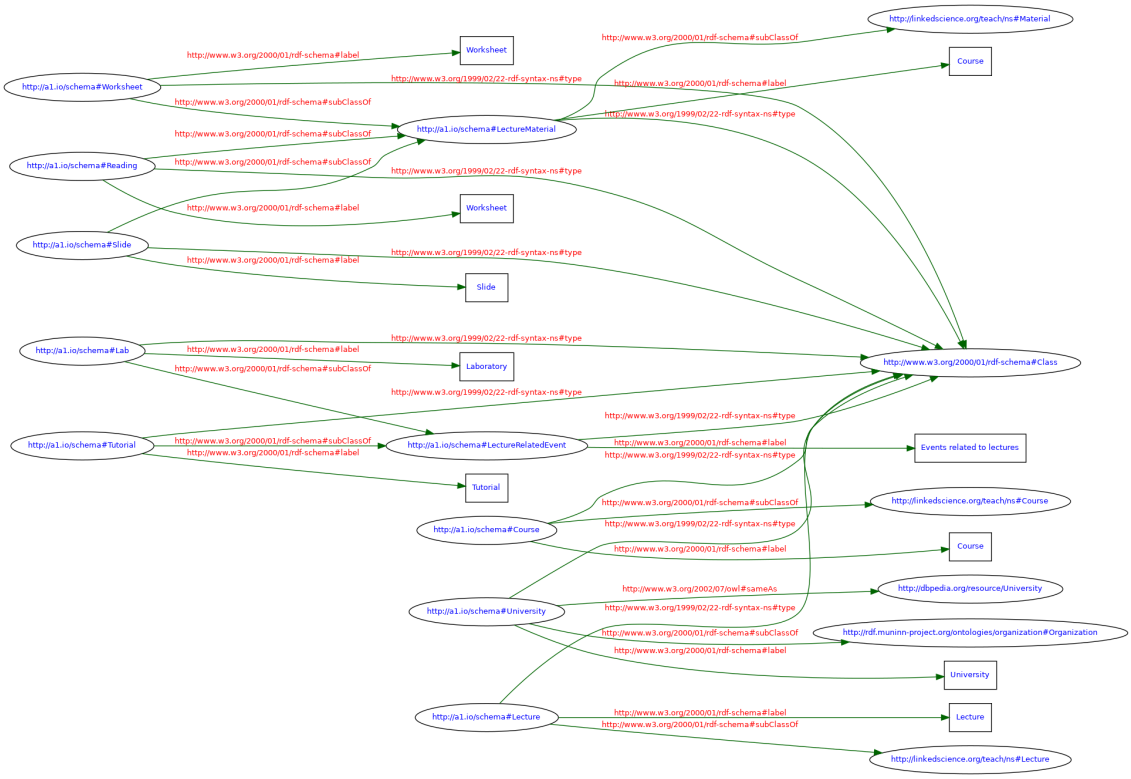

Figure 1: Schema-Defined Class Diagram

While best efforts were made to use existing properties, we sometimes opted to create, what we thought, were more accurate or bespoke properties with properly defined domains and ranges over generic predicates that may have been used multiple times with semantically different meanings. For example, we differentiated between a course's number and a lecture number (which is only meaningful sequentially). All defined properties and their respective domains and ranges can be found in Table 2.

Table 2: Schema-Defined Properties

Property	Domain	Range
HasLecture	Lecture	Course
RelatedToLecture	LectureRelatedEvent	Lecture
Offers	University	Course
HasSubject	Course	Literal
CourseNumber	Course	Literal
LectureNumber	Lecture	Literal
Outline	Course	Literal
HasContent	Course	Literal
HasMaterial	Lecture	LectureMaterial

4 Knowledge-Base Construction

In an effort to use the most recent data, we chose to use the most recently updated `CATALOG.csv` from `https://opendata.concordia.ca/datasets/`. Of course, we first clean the catalog

The dataset is limited in scope, having only a few of the fields relevant to the requirements (course number, code, name, and description). We decided that the dataset should be cleaned by removing entries that had no values in either the 'Course Code' or 'Course Number' columns.

Technically a website column existed, but was sparsely populated; henceforth we conditionally added a relevant triple if and only if that column was populated for a particular row. Similarly, the descriptions are either missing or inaccurate. To fill these gaps for our two 'special' courses, COMP 474 and COMP 353, we needed to either manually source accurate descriptions, course topics, content pieces, and other low-level data or make it up for the purposes of querying. The course materials are all pdfs, sourced directly from the 2021 course moodle page, accessible only to persons enrolled int he course; as such, we cannot provide a useful link to the original source of the data. Therefore, we provide local URIs for resources and example URIs for some of the web-examples in the knowledge graph. Regardless, the data itself is detailed in the following section.

To quickly parse the easy data from the csv dataset, we used the built-in csv python module and converted each row into a dictionary where the key of each value was the value of the first row of the spreadsheet, like so:

```
{...,'Course code':'COMP','Course number':'474',...}
```

Using `rdflib`, we parse the vocabulary graph created in the previous section and, for each course with an existing course code and course number, we create a course, make it offered by the predefined university, then add other course details as available.

The first thing to add after course details are lectures, to which the topics will be attached. Because we are only worried about the two special courses for now, they're the only ones that get lectures added to the graph, and therefore the only ones with topic data and course material. Then, because we decided to have labs and tutorials directly related to the lecture, as we create the lectures we create a corresponding lab and tutorial, which is attached to the lecture with our `RelatedToLecture` property.

After creating our lectures, we loop through our pre-compiled and normalized content folders to generate URIs and assign pieces of content to each lecture: slides and readings for 353 then slides and worksheets for 474. Each material is attached with a type (Readings, Slides, Worksheets, or Other) property to each lecture using a `HasMaterial` property.

Since everything has been zipped, the only steps needed to run the program and create a fresh version of the knowledge base are the following:

1. Unzip the submitted file
2. Navigate to our overarching `COMP474-A1` directory (so local URIs can resolve correctly)
3. Run `python3.6 kbpop.py` to regenerate the final kb, overwriting the one we submitted (usage of python 3.6 is mandatory)
4. We optionally allow the user to regenerate our cleaned catalog file

The resultant knowledge base on which we will run our queries in the next section can be found in `COMP474-A1/out`. We generate both, a turtle format `kb.ttl` and an n-triples format, `kb_ntriples.rdf` on each run, overwriting previously generated versions

5 Queries

The following queries were built to express the competency questions in section 1

```
PREFIX a1_schema: <http://a1.io/schema#>
PREFIX a1_data: <http://a1.io/data#>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX teach: <http://linkedscience.org/teach/ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

---

1. Which topics were covered in the second lecture of COMP 474?
We first query for all COMP 474's lectures, then among those we retrieve the lecture 2 and print its dbp:subject.

Query

```

SELECT ?topic
WHERE{
a1_data:COMP474 a1_schema:HasLecture ?lecture .
?lecture a1_schema:LectureNumber 2 .
?lecture dbp:subject ?topic
}
```

Result

```
topic -
dbr:Knowledge\_graph
```

---

2. What is course COMP 474 about?
This query selects the object of the courseDescription of the course, COMP 474. One might alternatively list the topics of lectures in the course; this is more succinct.

Query

```
SELECT ?Course_Description
WHERE {
a1_data:COMP474 teach:courseDescription ?Course_Description
}
```

Result

```
Course\_Description -
"Rule-based expert systems, blackboard architecture, and agent-based. Knowledge acquisition and r
```

---

3. Which courses at Concordia teach about knowledge graphs?
We check each lecture associated with the course using our HasLecture property, we then take the subject of the lecture and check it against the topic "knowledge graphs." We return the course.

Query

```
SELECT ?Course
WHERE{
a1_data:Concordia_University a1_schema:Offers ?Course .
?Course a1_schema:HasLecture ?Lecture .
?Lecture dbp:subject dbr:Knowledge_graph
}
```

Result

```
Course -
COMP 474
```

---

4. Where can I get more information about COMP 474?
Each course has an associated seeAlso containing the course website, so just return that.

Query

```
SELECT ?link
WHERE{
a1_data:COMP474 rdfs:seeAlso ?link
}
```

Result

```
link -
http://example.com/COMP474
```

---

5. Is COMP 474 taught at Concordia?
This asks a true or false question: does concordia offers COMP 474, the course. Simply translates to an 'ask' query

Query

```
ASK
WHERE{
a1_data:Concordia_University a1_schema:Offers a1_data:COMP474
}
```

Result
```
True
```

---

6. Which universities teach courses about database design?
This query searches for any courses with Database Design as their title, then looks for universities which offer said courses.

Query

```
SELECT ?university
WHERE {
?university a1_schema:Offers ?course .
?course teach:courseTitle "Database␣Design" .
}
```

Result
```
university -
a1_data:Concordia_University
```

---

7. How many COMP courses are taught at Concordia university?
This query returns all courses that match have "COMP" as their subject and that Concordia offers, then counts them.

Query

```
SELECT (COUNT(?course) AS ?count)
WHERE {
a1_data:Concordia_University a1_schema:Offers ?course .
?course  a1_schema:HasSubject "COMP" .
}
```

Result
```
count -
"46"^^xsd:integer
```

---

8. What is the course website for COMP 353?
This query simply looks at the seeAlso attribute of COMP 353 , which is the attribute being used for websites.

Query

```
SELECT ?website
WHERE {
a1_data:COMP353 rdfs:seeAlso ?website
}
```

Result
```
website -
http://example.com/COMP353
```

---

9. Which course has the fewest readings?
This query finds all lectures with readings, and all courses that have said lectures. It then groups the results by course and counts the number of readings in each group. It orders by the count of readings and returns only the first result, i.e. the course with the least readings, as well as how many readings that course has.

Query

```
SELECT ?course (COUNT(?readings) as ?count)
WHERE {
?course a1_schema:HasLecture ?lecture .
?lecture a1_schema:HasMaterial ?readings .
?readings a a1_schema:Reading .
}
GROUP BY ?course
ORDER BY ?count
LIMIT 1
```

Result

```
course -
<http://a1.io/data\#COMP353>

count -
"2"^^xsd:integer
```

---

10. Is Intelligent Systems a COMP or SOEN course?
This query looks for any course with the title "Intelligent Systems" then returns the HasSubject attribute, which is either COMP or SOEN. It needs to select DISTINCT because the Intelligent Systems course has results in the KB for both the graduate and undergraduate version, which are otherwise identical.

Query

```
SELECT DISTINCT ?program
WHERE {
?course a1_schema:HasSubject ?program .
?course teach:courseTitle "Intelligent␣Systems" .
}
```

Result

```
program -
"COMP"
```

KB STATISTICS QUERIES

Total Number of Triples:

Query

```
SELECT (COUNT(*) as ?triples)
WHERE { ?s ?p ?o }
```

Result

```
triples -
"7689"^^xsd:integer
```

Number of Course URIs:

Query

```
SELECT (COUNT(?course) as ?courses)
WHERE {
```

```
?course a a1_schema:Course .
}
```

Result

```
courses -
"1207"^^xsd:integer
```

Number of New Properties Defined:

Query

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

SELECT (COUNT(?property) AS ?count)
WHERE {
?property a rdf:Property
}
```

Result

```
count -
"9"^^xsd:integer
```

Number of New Classes Defined:

Query

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

SELECT (COUNT(?classes) AS ?count)
WHERE {
?classes a rdfs:Class
}
```

Result

```
count -
"10"^^xsd:integer
```

6 Appendix

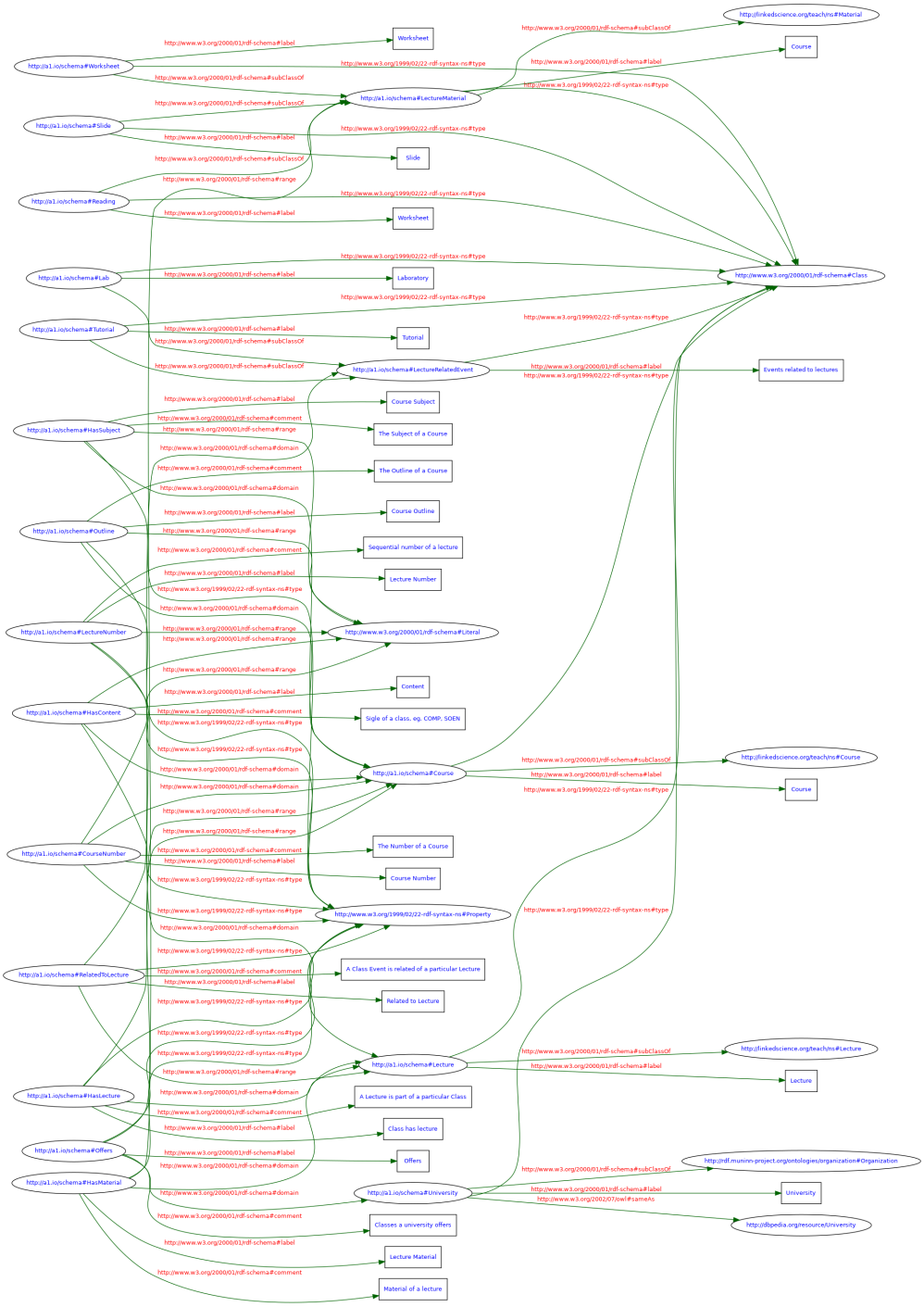

Figure 2: Knowledge Base Full Schema Diagram