Intelligent Systems

COMP 474 -- March 2020

Project Assignment #1

Team information	
Team Number 07	
Member Name	**Member SID**
Walid Ennaceur	40060735
Melanie Taing	40009850
Alexandre Therrien	40057134
Guénolé Le Masne de Chermont	40157385

0. Table of Contents

1. Introduction

In this project, we are going to create a knowledge graph about Concordia University. A university such as Concordia contains courses and students. This can be further broken down into course subjects, course numbers, student grades, student term enrollment, etc. The knowledge graph will connect those different aspects of the university together. This will enable us to ask questions to the graph, in the form of queries, and receive answers in the form of a list of triples. The project is separated into five components: course extraction, student creation, spotlight annotations, graph population and queries.

To complete the courses extraction, we save each web page from Concordia that gives at least one course description in HTML format. The reason why they are stored in HTML files is to enable the use of the software offline. The HTML files could then be updated when there is internet access. The next step is to extract the content of each html file, and to retrieve the courses using regular expressions. The courses are stored into a Course object which contains the title, number, subject, and the description relevant to the course. These courses are added to a list which will be used for spotlight annotation and graph population.

In the case of student creation, we created ten users defining their name, student id, and email. These students also have a curriculum, which is a list that contains the following elements in each entries: course acronym, course number, the term's season, the term's year and the grade obtained in the course. This list is then used to populate the graph.

As for spotlight annotations, these were collected using DBpedia Spotlight. Because there was a limited number of requests a user can make to the API, it was decided that a server with DBpedia Spotlight's model would be the solution. However, because that solution required a computer's full power, another computer was required. It was better using SSH to connect to the computer which served as the server to avoid any latency related to the server's management of memory. Once all the requests had been filed, it was important to make the topics into a set to prevent duplication of data. The final step was to connect the topics to the different courses' description or title they were found in, by using hashes to find the similarity of the terms.

Lastly, for populating the graph, we start by defining the uri prefixes that we use to generate the triples. We define that Concordia University is a public university. Then, we define every topic as a type ex:topic. Those topics are specified to be offered at Concordia. Similarly, we define each specific course, which is uniquely defined by the course acronym and number, as a course. We then add more details to the course, as well as the source of the data retrieved. On the other hand, we define each student as a type ex:student. Then we add triples that provide information about their name, the classes they are enrolled in, the semesters they were enrolled in, their student id, their email and their grades in the courses they have completed.

Lastly, the queries are used to obtain specific information about the elements in the knowledge graph. For instance, the number of triples in the graph, the number of topics, courses and students. It could also answer specific questions, for example, who is knowledgeable in classes regarding a certain topic.

2. Vocabulary

Vocabulary: Description how you modeled the schema for your knowledge base, including the vocabularies you reused, any vocabulary extensions you developed, etc. Give brief justifications where appropriate (e.g., choice of existing vocabulary).

We developed our schema the following way: …

The prefixes we used for our graph are the following:

```
@prefix course: <http://www.example.org/course/> .
@prefix dbo: <http://dbpedia.org/ontology/> .
@prefix dbp: <http://dbpedia.org/property/> .
@prefix exprop: <http://www.example.org/property/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix sch: <http://schema.org/> .
@prefix student: <http://www.example.org/student/> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```

Note the following vocabulary in bold are variables

The vocabulary we used for the Concordia University triple is the following: dbo:type, dbp:uniname. This is the structure:

```
<http://dbpedia.org/page/Concordia_University>
       dbo:type <http://dbpedia.org/page/Public_university> ;
       dbp:uniname "Concordia University" .
```

The vocabulary used to define Course Acronyms triple are the following: course, a, exprop:offered_at. This is the structure:

```
course:HIST
       a course:CourseAcronym ;
       exprop:offered_at <http://dbpedia.org/page/Concordia_University> .
```

We define courses with the following vocabulary: a, rdfs:label, sch:name, rdfs:subClassOf and course. The structure of every course triple is the following:

```
<http://www.example.org/course/subject_accronym/subject_number>
        a course:Course ;
          rdfs:label "course_description" ;
          sch:name "course_name" ;
          rdfs:subClassOf course:subject_accronym .
```

Where subject_accronym, subject_number, course_name and course_description are variables holding course information extracted from the concordia website.

Students are defined with the vocabulary : a, student, sch:name, a, exprop:enrolled_in, exprop:enrolled_to and exprop:identified_by. The structure is the following.

```
<http://www.example.org/student/student_email> a student:Student ;
   sch:name "user_full_name" ;
   exprop:enrolled_in
       <http://www.example.org/course/subject_accronym/subject_number/student_email/t
erm/year>,
       <http://www.example.org/course/subject_accronym/subject_number/student_email/t
erm/year>;
   exprop:enrolled_to
       <http://www.example.org/course/subject_accronym/subject_number>,
       <http://www.example.org/course/subject_accronym/subject_number>;
   exprop:identified_by "student_id" .
```

Where subject_accronym, subject_number are variables extracted from course extraction and student_email, user_full_name, term, year are extracted from student information created before in our student class.

Grades of class in a term are defined with the vocabulary exprop:completed_with, exprop:enrolled_semester and exprop:enrolled_year. The structure is the following.

```
<http://www.example.org/course/subject_accronym/subject_number/student_email/term/yea
r> exprop:completed_with grade ;
   exprop:enrolled_semester <http://www.example.org/semester/term> ;
   exprop:enrolled_year "year" .
```

Where subject_accronym, subject_number are variables extracted from course extraction and student_email, term, year, grade, year are extracted from student information created in our student class.

3. Knowledge Base Construction

Knowledge Base Construction: Describe (a) your dataset and (b) your process and developed tools for populating the knowledge base from the dataset. Describe how to run the tools to create the knowledge base. Explain your process for linking entities to DBpedia.

To concretely build the knowledge graph as a turtle RDF file, we first added all the prefixes in the knowledge graph. After that, we created Concordia University's triples, which were described in the vocabulary section, with the information added as a string following the prefix of the URI which are : http://dbpedia.org/property/ and http://dbpedia.org/ontology/.

After that, we created the courses triples. To do so, we used Concordia University's Undergraduate and Graduate calendar. More specifically, the pages containing course descriptions of each department. The triples for courses and topics are automatically generated through a series of steps. First, the course description pages are downloaded from Concordia University's website. Second, the courses and course descriptions are extracted using regular expressions. Third, the information regarding those courses are stored in a text file for further reference. Using the extracted course information from the text file, we can create Course triples.

We used DBpedia Spotlight to parse the course title and course descriptions to find related entities with a confidence level of 0.35. This confidence level was carefully selected to minimize the amount of garbage links and ensure that at least one entity would be found.

The DBpedia API restricts the number of requests per user, and so to bypass this restriction, we downloaded a DBpedia model (2016) and used Apache Jenna Fuseki to query entities locally. However, because the resources being used by the DBpedia model were enormous, a second computer came in handy. That client computer would then create a SSH connection with the server to query the information it needs and report back the response to the client computer. Once all the topics had been collected, we applied a filter to remove all the duplicates. Finally, we used that set of topics to identify them in the different courses' title and description to associate to which course node we had to add a URI reference for a topic. These entities were then used to create Topic triples.

To create the students triple, our method _generate_students linked to the Student's class creates 10 fictitious students with name, email, student id and course curriculum. Then for each student, we added triples that corresponded to the relationship of a student enrolled to a course and the course that have the student enrolled, the year and semester these classes were followed by the student and the grades s/he had in his/her time at Concordia.

4. Queries

Queries: Describe your translation of the knowledge base queries above into SPARQL. Provide (brief) example output for each query.

We had to do 6 queries generated in 6 different text files:

1. Total number of triples in the KB
2. Total number of students, courses, and topics
3. For a course c, list all covered topics using their (English) labels and their link to DBpedia
4. For a given student, list all courses this student completed, together with the grade
5. For a given topic, list all students that are familiar with the topic (i.e., took, and did not fail, a course that covered the topic)
6. For a student, list all topics (no duplicates) that this student is familiar with (based on the completed courses for this student that are better than an “F” grade)

The output generated is written in text files containing the answer of the query.

For query 1, we used the aggregate function COUNT() in a SELECT to get the total number of triples.The query we used is the following :

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

The output generated is the number of triples which is : 79131.

For query 2:

```
SELECT (COUNT(DISTINCT ?s) as ?studentCount)
(COUNT(DISTINCT ?c) as ?courseCount)
(COUNT(DISTINCT ?t) as ?topicCount)
WHERE
{
   ?s a student:Student .
   ?c a course:Course .
   ?t a topic: .
}
```

The output generated is the number of triples which is : `10 4078 8142`

For query 3, we wanted to have all the topics and their english label so we chose the ACCO220 which has the label : "This course provides an introduction to accounting principles underlying the preparation of financial reports ..."

```
SELECT ?topic ?label
WHERE {
    <http://www.example.org/course/COMM/223> sioc:topic ?topic .
    ?topic rdfs:label ?label .
} GROUP BY ?topic
```

Results

```
http://dbpedia.org/resource/Comm Comm
http://dbpedia.org/resource/Technology Technology
http://dbpedia.org/resource/Marketing_management Marketing Management
http://dbpedia.org/resource/Marketing Marketing
```

For query 4, we want to get the courses completed by the student corresponding to the id (here 422222) and his grades for each course.

```
SELECT ?completed_course ?grade
WHERE {
   ?student a student:Student .
   ?student exprop:identified_by "422222" .
   ?student exprop:enrolled_in ?completed_course .
   ?completed_course exprop:completed_with ?grade .
}
```

The output looks like that :
http://www.example.org/course/**COMP/326**/frank@outlook.com/FALL/2010 **32**
http://www.example.org/course/**SSDB/428**/frank@outlook.com/WINTER/2008 **43**
http://www.example.org/course/**ADED/202**/frank@outlook.com/FALL/2014 **55**

The query outputs a list of courses with the grade corresponding to each course.In **red** we have the course the student with id = 422222 was enrolled in. And in **green** we have his grade.

For query 5, we are given a topic, in this case the topic is the uri http://dbpedia.org/resource/Intensify in this case the course found that is linked to that topic is TESL 491, which the 3 following students took and passed

```
SELECT ?s
WHERE
{
   ?s        rdf:type              student:Student .
   ?c        rdf:type              course:Course .
   ?t        rdf:type              topic: .
   ?c        sioc:topic            ?t .
   ?s        exprop:enrolled_to    ?c .
             exprop:enrolled_in    ?curriculem_course .
   ?curriculem_course exprop:completed_with  ?grade.
   FILTER( ?course = <{topic_uri}> )
   FILTER contains(?curriculem_course, ?c)
   FILTER ( ?grade >= 50 )

}
```

Output is:
http://www.example.org/student/bianca@concordia.ca
http://www.example.org/student/julie@snyder.com
http://www.example.org/student/roger@moores.ca

```
SELECT ?topic
WHERE {
   ?student a student:Student .
   ?student exprop:identified_by "333333" .
   ?student exprop:enrolled_in ?enrolled_in .
   ?enrolled_in exprop:completed_with ?grade .
   ?enrolled_in course: ?course .
   ?course sioc:topic ?topic .
   FILTER (?grade >= 50)
}
```

Results
http://dbpedia.org/resource/Adult_education
http://dbpedia.org/resource/Education
http://dbpedia.org/resource/Norfolk_Scope
http://dbpedia.org/resource/Ethics
http://dbpedia.org/resource/Facilitator
http://dbpedia.org/resource/Failure
http://dbpedia.org/resource/Mindfulness
http://dbpedia.org/resource/Motivation

http://dbpedia.org/resource/Psychological_stress
http://dbpedia.org/resource/Self-care
http://dbpedia.org/resource/Self-monitoring
http://dbpedia.org/resource/Self-reflection

5. Conclusion

To conclude, this project was made to be able to get any information of students or courses at Concordia University. This was done by creating a knowledge graph, and connecting our information in the form of triples in the Turtle RDF format.

The first step was to extract the courses that were on Concordia's web pages. We did so by downloading the corresponding HTML files, extracting the important content from them, and storing that information into a file.

The second step was to create ten random students with a random curriculum at Concordia (random grades, random courses they registered to, along with the term and year they registered to the class).

The third step was to get all the topics from DBpedia Spotlight from the courses' title and description. Because the API limited our number of requests, we used our computer as a Spotlight server to which we sent requests to. This demanded the use of a second computer, to which we established a SSH connection to the first one. Finally, we reconnected the topics to the courses they belonged to.

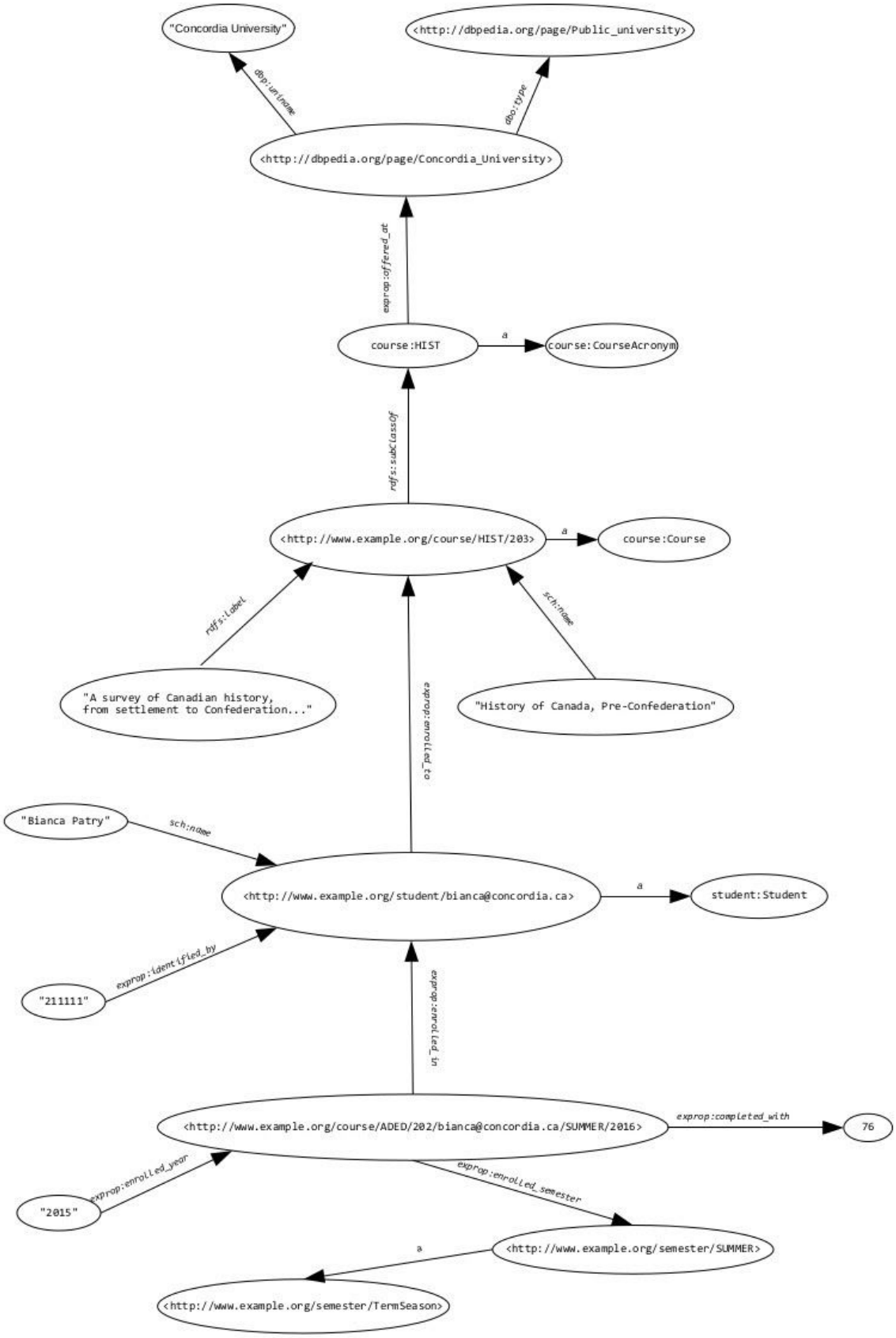
The final step was to create the knowledge graph with all the information we had. By using rdflib and respecting the format for RDF Turtle, we were able to do it.

As requested from the project's guideline, a set of queries have also been implemented to show the behavior of our knowledge graph.

6. References & Resources

https://www.w3.org/Submission/sioc-spec/

https://github.com/dbpedia-spotlight/dbpedia-spotlight/wiki/Run-from-a-JAR


"Concordia University"
<http://dbpedia.org/page/Public_university>
dbp:urlname
dbo:type
<http://dbpedia.org/page/Concordia_University>
exprop:offered_at
course:HIST
a
course:CourseAcronym
rdfs:subClassOf
<http://www.example.org/course/HIST/203>
a
course:Course
rdfs:label
sch:name
"A survey of Canadian history,
from settlement to Confederation..."
"History of Canada, Pre-Confederation"
exprop:enrolled_in
"Bianca Patry"
sch:name
<http://www.example.org/student/bianca@concordia.ca>
a
student:Student
exprop:identified_by
"211111"
exprop:enrolled_in
<http://www.example.org/course/ADED/202/bianca@concordia.ca/SUMMER/2016>
exprop:completed_with
76
exprop:enrolled_year
exprop:enrolled_semester
"2015"
<http://www.example.org/semester/SUMMER>
a
<http://www.example.org/semester/TermSeason>