COMP 474/6741 Intelligent Systems (Winter 2020)

Project Assignment #1

Due date (Moodle Submission): Monday, March 2
Counts for 15% of the total marks (half of the course project)

This is the first of two project assignments in our course. Note that the assignments are not 'stand-alone', but rather build on top of each other to form a bigger project.

The overall goal is to build an *intelligent agent* that can answer university-related questions using a knowledge graph and natural language processing. The first assignment focuses on constructing the knowledge graph, whereas the second will target the NL interface and the overall integration.

University Knowledge Graph. The goal in this part is the construction of a knowledge graph, built using standard W3C technologies, in particular RDF and RDFS. You have to model your graph to represent the information below.

Universities. Information about universities:

1. Name
2. Link to the university's entry in *DBpedia*.

Courses. Information about *courses* offered at a university, including:

1. Course name (e.g., "Intelligent Systems")
2. Course subject (e.g., "COMP", "SOEN")
3. Course number (e.g., "474")
4. Course description (e.g., "Knowledge representation and reasoning. Uncertainty and conflict resolution . . . ")
5. A link (`rdfs:seeAlso`) to a web page with the course information, if available

Topics. Course *topics*, extracted from title and description (e.g., "Expert system", "knowledge representation"). Each topic must be linked to its corresponding URI in DBpedia. For example, the topic "Expert system" in a course description would be linked to http://dbpedia.org/resource/Expert_system.

Students. Information about students:

1. Name (first, last)
2. ID Number
3. Email
4. Completed courses with their grades (note: you do not have to model courses-in-progress)

Your first task is to created an **RDF Schema** (in Turtle format) to capture these information. Guidelines:

- Re-use existing vocabularies where appropriate (e.g., FOAF).
- If you need to define new concepts and/or properties:

  - extend existing vocabularies where possible
  - each of your new classes and predicates must have a *label* and *comment* in English (you can add additional languages if you like).

Automated Knowledge Base Construction. A major task of this assignment is to *automate* the knowledge base construction, rather than writing triples by hand. Towards this end, you have to develop (a) a dataset with pertinent information (e.g., about courses) and (b) one or multiple Python programs that can transform the dataset into RDF triples. For example, you could add pages from the graduate calendar (https://www.concordia.ca/academics/graduate/calendar/current/encs/computer-science-courses.html) to your dataset and come up with a strategy to convert it into triples for the RDF Schema you defined above. In particular, note that you will need an approach for *linking* URIs in your graph to DBpedia.

Notes:

- For this project, you can limit your work to information related to *Concordia University.*
- Student information does not have to be real data (that is, you can invent names, id numbers, grades, etc.). However, you have to add at least 10 different students with some course & grade information to your graph.

Knowledge Base Queries. Write SPARQL queries that retrieve the following information from your knowledge base:

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)

Report. Write a report about your knowledge graph with the following information:

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).

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.

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

Additionally, for **graduate students (COMP 6741):**

Link Analysis: Analyse the performance of your linking approach on your dataset. Provide statistics for at least 100 generated links showing (1) the source text (surface form) in the dataset to be linked; (2) the link generated by you, and (3) the expected, correct link. Compute the *accuracy* of your links as the percentage of correct links (e.g., if 80 of 100 links you generated are correct, your accuracy is 80%).

Deliverables. Your submission must include the following deliverables within a single `.zip` archive:

RDF Schema: Your RDFS file in Turtle format.

Dataset: Your dataset used to construct the knowledge base (e.g., course descriptions, academic calendars, etc.), in their original and any converted formats. Provide the source information for all files (filename, source URL).

KB Construction: Your Python program(s) to (1) automatically construct the knowledge base from the dataset above, including those to (2) link URIs to DBpedia. It must be possible to re-construct your knowledge base using the submitted tools and dataset.

Knowledge Base: Your complete, constructed knowledge base (in N-Triples format).

Queries and their result: Text files with your queries (`q1.txt`, `q2.txt`, . . .) and the full output of your queries when run on your KB (`q1-out.ttl`, `q2-out.ttl`, . . .).

Report: The project report, as detailed above, as PDF.

Submission. You must submit your code electronically on Moodle by the due date (late submission will incur a penalty, see Moodle for details). Include a signed (by all team members) *Expectation of originality* form (see https://www.concordia.ca/encs/students/sas/expectation-originality.html) with your submission.

Demo. You must also demo your code to your TA in one of the lab sessions (time slots for the demo will be reserved through Moodle). Note that **all** group members must be present for the demo and ready to answer questions about the code and data.