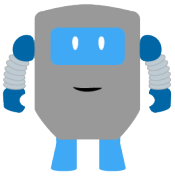
COMP 474/6741 Intelligent Systems (Winter 2022)

Project Assignment #1

Due date (Moodle Submission): Monday, March 21st
Counts for 18% of the total marks

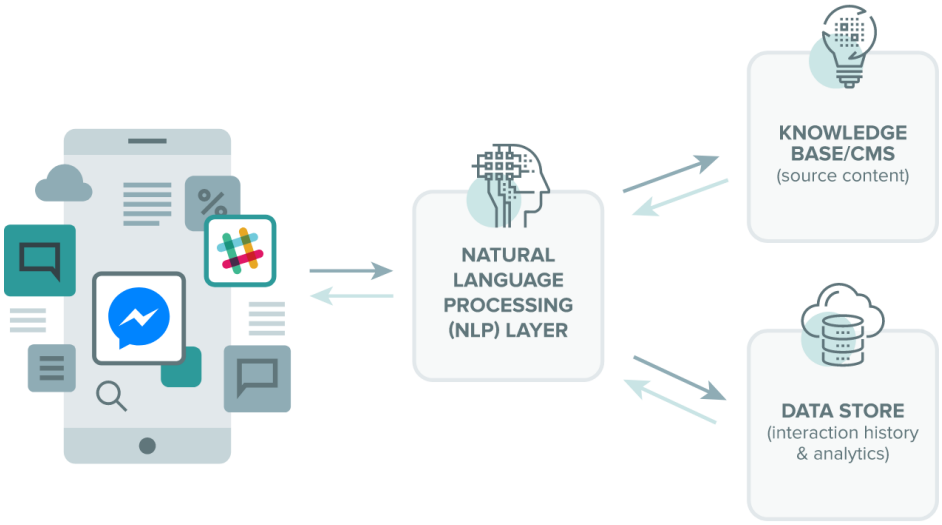
This is the first of two project assignments in our course. Note that the assignments are not 'stand-alone', but rather build on top of each other to form a bigger project.



The overall goal of this project is to build **Unibot**, an *intelligent agent* that can answer university course- and student-related questions, using a knowledge graph and natural language processing.

For example, Unibot will be able to answer questions such as, *"What is course COMP 474 about?"*, *"Which topics is Jane competent in?"* or *"Which courses at Concordia teach deep learning?"*

Project structure. This first assignment primarily focuses on constructing the knowledge base, whereas the second part will target some enhancements, the natural language processing interface, as well as the overall integration.



Unibot's Knowledge Graph Vocabulary. To be able to answer questions, *Unibot* needs knowledge about courses, lectures, and their content. Thus, the first step in this part of the project 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 at least the information below (you can add more information that you find useful).

Universities. Information about universities:

1. Name of the university
2. Link to the university's entry in *DBpedia* and/or *Wikidata*

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 credits
5. Course description (e.g., "Knowledge representation and reasoning. Uncertainty and conflict resolution . . . ")
6. A link (`rdfs:seeAlso`) to a web page with the course information, if available.
7. Course outline, if available.

Lectures. Information about *lectures* in a course, including:

1. Lecture number (sequential count)
2. Lecture name, if available (e.g., "Knowledge Graphs")
3. Lecture content, such as:
   - Slides
   - Worksheets
   - Readings (book chapters, web pages, etc.)
   - Other material (videos, images, etc.)

   (create a subclass hierarchy for different lecture content types)
4. A link (`rdfs:seeAlso`) to a web page with the lecture information, if available

Topics. Information about the *topics* that are covered in a course:

1. Topics must have *provenance* information, that is, you must record where the topic was identified as being covered in the course (e.g., the graduate calendar, the course outline or a specific lecture)
2. A topic must be linked to DBpedia and/or Wikidata. For example, the topic "Knowledge graph" in a course description could be linked to https://dbpedia.org/resource/Knowledge_graph.

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, but you do have to model the possibility that a student re-takes a course (e.g., to obtain a passing grade).
5. Competencies, which is defined as a set of topics, based on the courses a student successfully passed (e.g., if a student passes COMP474, a course that includes the topic "Knowledge Graphs", then this student is now considered competent in this topic)

Your first task is to created a **Vocabulary** using RDFS (to be submitted in Turtle format) to capture these information. Guidelines:

- Re-use existing vocabularies where appropriate (e.g., FOAF,[1] VIVO[2]).
- 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).

Competency Questions. Write ten different competency questions that you would like your bot to be able to answer. The first three must be a generalized version of the questions shown in the introduction, e.g.,

- Which [topics] are covered in [course] [lecture]?

For the possible formats and examples of CQs, please check Lecture #5 on knowledge base design.

Create at least one SPARQL query for each competency question that validates your design.

Automated Knowledge Base Construction. To fill our agent's brain, we need to create a *knowledge base* using the vocabulary defined above. Towards this end, in this first part you have to populate the knowledge base with information about Concordia's courses from the open datasets available at https://opendata.concordia.ca/datasets/.

For the *content* of a course (lectures, labs, tutorials, etc.), you will have to add two courses to your knowledge base:

1. Our course, COMP474/6741
2. A second course that you took in the past (you can pick one as a team, but at least one team member must have taken the course)

In this first part, collect and organize the material for the two courses into suitable datasets (e.g., folders with lectures, slides, etc.).[3] Give each item an automatically generated URI that you can use to identify the item. You can use `file://` URIs that resolve locally,[4] for example, to refer to a set of slides in our course, you could generate a URI like `file:///home/unibot/COMP474_6741/lectures/slides01.pdf`

In the second part of the project, we will automatically *extract* information about topics using NLP techniques from these datasets. For this first part of the project, simply create a few triples manually that demonstrate how the knowledge base will look like (e.g., modeling a few topics for our course).

Make sure that these data triples are *in a different namespace* from your vocabulary.

Knowledge Base Queries. Set up a query server using Apache Fuseki[5] where you load and query your generated triples. Run the queries you developed for the competency questions and provide sample output for each. In addition, provide queries and their output for statistics about your KB, such as the total number of triples and the number of course URIs.

[1] See http://xmlns.com/foaf/spec/
[2] See https://wiki.lyrasis.org/display/VIVODOC112x/VIVO+Ontology+Domain+Definition
[3] You do not need to add video files, since we are not processing these further within this project.
[4] You could also setup a document server in order to create HTTP-resolvable URIs for the course content, using a server like Couchbase or CouchDB, but this is not required for this project.
[5] See https://jena.apache.org/documentation/fuseki2/

Report. Write a report about your design & implementation with the following information:

Competency Questions: The competency questions that you used for designing your agent.

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.

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

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 automatically construct the knowledge base from the dataset above. In other words, it must be possible to re-construct your knowledge base using the submitted Python tools and datasets.

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

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

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/ginacody/students/academic-services/expectation-of-originality.html) with your submission.

Demo. We will schedule demos sessions for your project on Moodle. All team members must be present for the demo.