COMP 474 UU - Intelligent Systems
Assignment 1

Conupedia - Project Report

Matthew Massey
Rani Rafid
Roman Lewandowski
Sean Goharzadeh

09-03-2020

Contents

1. Vocabulary

1.1 Vocabulary Re-Used

In accordance to what is considered to be best practice when creating our knowledge base, all the vocabulary and prefixes used in our project is based on existing vocabularies when possible. The following are the vocabularies used in our knowledge base, as well as their original URI and what they are used for.

- **dbr (http://dbpedia.org/resource/)**. Describes a DBpedia resource. As of the first stage of the Conupedia project, this prefix is only used to point to Concordia's own DBpedia page to indicate that a course is provided by Concordia University.
- **foaf (http://xmlns.com/foaf/0.1/)**. Refers to the "friend-of-a-friend" vocabulary, which describes people, their details, and their relations with one another. In the Conupedia project, foaf is used to establish that a student is indeed a person, and is also used for connecting (fictional) students to personal information with the following predicates:
  - **firstName**, the relation between a student and their given name.
  - **lastName**, the relation between a student and their family name.
  - **mbox**, the relation between a student and their email address. Note that this is an inverse function relation, meaning that students must have a unique email address. Multiple subjects with the same email address object would indicate that the subjects are equivalent, or the same person.
- **owl (http://www.w3.org/2002/07/owl)**. Refers to the Web Ontology Language, OWL, which is used for describing the structure of knowledge and its relations. In the Conupedia project, OWL is used for its "sameAs" predicate, linking the subject (a course) to an object (its one or multiple topics.)
- **rdf (http://www.w3.org/2000/01/rdf-schema)**. The general RDF specification is used in the Conupedia project to establish that each course's code from the Concordia Open Data indeed refers to a course object as defined by schema.org, using rdf:type.
- **rdfs (http://www.w3.org/2000/01/rdf-schema)**. Refers to RDF schema, a vocabulary for modelling RDF data. In the Conupedia project, rdfs is used for the following predicates:
  - **domain**, to connect the "took" and "grade" vocabulary extensions defined in the Conupedia Ontology (see the following section) to its domain, a person (who took a course).
  - **label**, to label a student's grade assessment with a course.
  - **range**, to connect the "took" and "grade" vocabulary extensions defined in the Conupedia Ontology (see the following section) to its range, a course and a letter grade respectively.
  - **subClassOf**, to establish that a student is a subclass of person.

- **schema (http://schema.org/version/6.0/)**. Schema.org is described as "a collaborative, community activity with a mission to create, maintain, and promote schemata for structured data on the Internet." Because of the extensive reach of Schema.org, the schema prefix is by far the most used one in the Conupedia project, and describes a large number of relations between subjects and objects, such as:
  - **courseCode**, the relation between a course and its code (eg, COMP474.)
  - **name**, the relation between a course and its name (eg, Intelligent Systems.)
  - **numberOfCredits**, the relation between a course and its number of credits.
  - **coursePrerequisites**, the relation between a course and its list of prerequisites.
  - **provider**, the relation between a course and the university providing it.
  - **description**, the relation between a course its description, usually one paragraph long.
- **xsd (http://www.w3.org/2001/XMLSchema)**. The xsd prefix is used for XML schemas. In the Conupedia project, xsd is used to establish that the number of credits in a class subject is an object meant to be interpreted as a floating point number.

1.2 Vocabulary Extensions Developed

The vocabularies defined specifically for the Conupedia project itself are the Course vocabulary extension, and the Conupedia Ontology vocabulary extension. We found these to be necessary to develop because not all courses can be identified by an existing URI. In fact, many courses are technically seminars offered with limited availability, or are no longer offered by Concordia University today. In such circumstances, adequate URIs might have ceased existing. For this reason, as well as for consolidating all URIs to a single referable location, the Conupedia project provides its own URI for courses as well as the relations they play a part in (cpc and cpo respectively.) They serve the following purpose:

- **cpc (http://www.conupedia.sytes.net/Course/)** The Conupedia Course vocabulary extension is for giving a home to each course that was captured from Concordia's Open Data. A typical course URI from the Conupedia knowledge base looks like this: <http://www.conupedia.sytes.net/Course/005484>.
- **cpo (http://wwww.conupedia.sytes.net/Ontology/)** The Conupedia Ontology vocabulary extension is built for defining students and relationships pertaining to students, connecting them to courses and assessments for said courses. The following are included within the cpo vocabulary:
  - **took**, the relation between a student and an assessment. Each assessment is also labeled with the course that is being assessed.
  - **grade**, a grade object, expressed as a letter grade (eg, B+), which is connected to an assessment.

2. Knowledge Base Construction

2.1 Dataset

The central dataset used for populating our knowledge base is none other than the raw data provided by Concordia's own Open Data project, which is free of restrictions from copyright or patents (Concordia University, 2020). The CSV files provided here were instrumental in providing information with regards to which courses exist, along with their names, codes, descriptions and prerequisites.

Students, however, along with their course history and grades, are completely fictitious and are not based on any existing datasets.

2.2 Tools and Process

2.2.1 AWS and Virtuoso

Conupedia was given a home by establishing Amazon's AWS backend server running from Ubuntu, upon which was installed Virtuoso's open source server, serving as the crucible upon which the Conupedia database and SPARQL endpoint rests. Ports were configured to be bound to 80 and 443, and a public IP address (Elastic IP) was provided by the AWS console itself.

Using noip.com, Conupedia's current domain, conupedia.sytes.com, was obtained. Subsequently, https connections were allowed by using self-signing certificate authority. With all of the above steps and precautions aside, and with Virtuoso documentation having been followed appropriately, Conupedia now has a completely functional backend.

2.2.2 Course Info Parser

The course information parser contains a two-parameter function that takes an input file and an output file name. Using a .json format document processed from the Concordia Open Data .cvs file, the purpose of this tool is to create rdf-turtle format tuples, which can be fed into the knowledge base. The function begins by declaring all prefixes at the beginning of the output file, and then, one course at a time, the tool processes the information from the .json file and creates an appropriate and well-formatted triple.

The subject of each triple is a unique identifier number for each course, which exists within the Open Data .json files mentioned above. Each such course has the following properties, expressed in triples:

- A course code, eg: "COMP474", which is a string literal.
- A number of credits, which is a floating point number.

- A name, eg: "Intelligent Systems", which is a string literal.
- An "is-part-of" property pointing to the corresponding level of education on Concordia's website. eg: <https://www.concordia.ca/academics/undergraduate.html>
- Pre-requisites, which are string literals. The decision to use string literals stemmed from the fact that the formatting in the Open Data files is inconsistent.
- A provider, pointing to the University that provides the course. In this iteration of the function, all courses are provided by Concordia University, Montreal, and therefore the object of each tuple is invariably the URI <http://dbpedia.org/resource/Concordia_University>.

2.2.3 Description Parser

The description parsing tool, description.py, serves the dual purpose of both providing description tuples to our knowledge base as well as forwarding these tuples to the topic finder tool.

The module begins by declaring the prefixes needed to create the course-described by-description tuples. For the purpose of serving as a predicate, Schema.org's "schema:description" is used. The module then reads from the description.json file, provided by Concordia's raw Open Data dataset, and parses each of the descriptions from the file into the object of the matching course's tuple. As with the other tools responsible for creating triples with regards to courses, the subject of each triple is the internal code present in the .cvs files in Concordia's Open Data project. While parsing the input file, a series of regular expression replacements are made to remove extraneous information and strange formatting artifacts from the descriptions.

The purpose of these changes is two-fold. First, on a simply pragmatic level, the extraneous information is likely to already be found in other tuples within our knowledge base. For instance, tuples already exist for listing course prerequisites, which are present in many of the descriptions provided by the Open Data json file. A very large number of the course descriptions are also merely referrals to other places, such as the Graduate or Undergraduate calendars.

The second, and arguably more important reason for this filtering, though, is because the module responsible for creating the course-has topic-topic tuples reads these topics directly from the descriptions, effectively making the description parser act as a filter. As such, it's important to remove as much of the text that isn't part of the course's true description as possible. For example, leaving the sentence "Students must take PHIL 201 as a prerequisite" can cause the topic finder to misinterpret "PHIL" as being a reference to the city of Philadephia.

Items removed by this filtering process include the following:

- Mentions of course prerequisites
- Notes concerning credits and concurrent courses
- Boilerplate artifacts
- Audio-visual element artifacts
- Superfluous punctuation such as asterisks (*) and tildes (∼)
- Double-quotes (") which cause errors in string parsing in later stages of the pipeline.

2.2.4 Topic Finder

The topic finder module, topics.py, works hand-in-hand with the description parser module, using its output file as its own input. The purpose of this module is to find the topics of a given course by reading the description string and identifying relevant keywords.

Given that the description parser's output is a .ttl file, it was necessary to convert it to a .json file for easier access to functions provided by python's own built-in json module. For this purpose, the topic finder module makes a call to the format converter module described later on in the "Minor Tools" section of this report.

With the description of a course now easily identified, the topic finder module connects to DBpedia spotlight and throws the description into its API with a confidence interval of 0.35, which was tested as being the most reliable for the purpose of finding relevant topics. While the DBpedia spotlight API allows only a limited number of connections, the topic finder module is able to bypass this restriction by connecting to another secondary module, the VPN-based API Limit Bypasser module (also described in further detail later on). Once these results are obtained, they are, as with the other major modules, stored into a .ttl file, formatted from well-formed tuples that use the unique course identifier number provided by Concordia's Open Data project as a subject. The URIs obtained by the DBpedia Spotlight API are the object, and the owl:sameAs property, the predicate.

2.2.5 Minor Tools

2.2.5.1 Format Converter

The Format Converter, rdf_converter.py, is a small but effective single-function module created to convert files from one rdf format to another, making it possible to switch between .ttl, .xml and .json on the fly. This works by opening a target file, reading its contents, and then creating a session with the server, with a POST request containing the file's contents. The request is sent to an online rdf converter, and the response is obtained and stored to a file path defined by the function's second parameter, the output name. Both input and output must be of .ttl, .xml or .json formats.

This particular tool proved most useful for converting output files from other tools into .json formats, allowing usage of other built-in python tools such as the json module, which allowed the easy organization of data into dictionaries containing entities.

2.2.5.2 OpenData Module

The OpenData Module, open_data.py, is for connecting to Concordia's OpenData platform and using API calls to retrieve information about courses, which are in turn parsed into tuples using the major tools described earlier on. It returns an object of .json format.

2.2.5.3 VPN-Based API Limit Bypasser

Given that Dbepdia Spotlight enforces strict limitations as to the number of API calls made by a certain IP address, it became clear early on that there would be no way to make an adequate number of calls to the API in order to satisfy the needs of the project.

In order to circumvent these restrictions, a VPN-based API limit bypass module, called nordvpn.py, was created to instantiate a new NordVPN session every time API call limit was reached. This wrapper module works on

top of an existing NordVPN command-line application, which was paid for and obtained through subscription. Through this, the functions within the module fire through a list of over 40 different countries, cycling through this list in such a fashion that the number of calls to the API is now effectively limitless.

3. Queries and Sample Outputs

Courtesy of Virtuoso, Conupedia has its own SPARQL endpoint, which can be accessed from `http://conupedia.sytes.net/sparql`. The following are six queries that can be run from this query editor, as well as a portion of the results of these queries.

3.1 Number of Triples in the Knowledge Base

```
SELECT (COUNT(*) AS ?triples)
WHERE
{
?a ?b ?c .
}
```

Result:

Triples
79962

3.2 Number of Students, Courses and Topics

```
SELECT (COUNT(?a) AS ?Students),(COUNT(?c) AS ?Courses),COUNT(DISTINCT ?topic) AS ?Topics

WHERE {
{ ?a rdfs:subClassOf foaf:Person . }
    UNION
{ ?c rdf:type schema:Course  . }
    UNION
{ ?course owl:sameAs ?topic . }
}
```

Result:

Students	Courses	Topics
13	7050	9420

3.3 All Topics Covered by a Given Course, and their Link to DBpedia

```
SELECT   ?topic, ?link
WHERE {
    ?courseId schema:courseCode "COMP249" .
    ?courseId owl:sameAs ?Link .
    ?link rdfs:label ?Topic .
    FILTER (langMatches( lang(?topic), "EN" ) )
}
```

Result:

Topic	Link
"Dynamic dispatch"	http://dbpedia.org/resource/Dynamic_dispatch
"Recursion"	http://dbpedia.org/resource/Recursion
"O"	http://dbpedia.org/resource/O
"Polymorphism (biology)"	http://dbpedia.org/resource/Polymorphism_(biology)
"Generic programming"	http://dbpedia.org/resource/Generic_programming
"Inheritance"	http://dbpedia.org/resource/Inheritance

3.4 All Completed Courses and Grades of a Given Student

```
SELECT "Nick", "Lawson", ?Course, ?Grade  WHERE {
?s rdfs:subClassOf foaf:Person .
?s foaf:firstName "Nick" .
?s foaf:lastName "Lawson" .
?s cpo:took ?c .
?c rdfs:label ?u .
?c cpo:grade ?Grade .
?u schema:name ?Course .
}
```

Result:

callret-0	callret-1	Course	Grade
Nick	Lawson	Object-oriented Programming I	A
Nick	Lawson	Information Systems Security	D+
Nick	Lawson	Intelligent Systems	A
Nick	Lawson	Artificial Intelligence	A+
Nick	Lawson	Ethnic Communities In Canada	C

3.5 All Students Familiar with a Given Topic

```
SELECT ?FirstName, ?LastName, "COMP248", ?Grade  where {
?s rdfs:subClassOf foaf:Person .
?s foaf:firstName ?FirstName .
```

```
?s foaf:lastName ?LastName .
?u schema:courseCode "COMP248" .
?s cpo:took ?c .
?c rdfs:label ?u .
?c cpo:grade ?Grade .
FILTER(?Grade != "F") .
}
```

Result:

FirstName	LastName	callret-2	Grade
Nick	Lawson	COMP248	A
Gale	Winston	COMP248	A-
Amy	Weinstein	COMP248	B+
Toby	Keith	COMP248	C
Pete	Sampres	COMP248	B

3.6 All Topics a Given Student is Familiar With

```
SELECT  DISTINCT  ?Name ,?Last, ?Topic, ?Grade
WHERE {
    ?s rdfs:subClassOf foaf:Person .
    ?s foaf:firstName "Tyler" .
    ?s foaf:lastName "Perry" .
    ?s cpo:took ?assessment .
    ?s foaf:firstName ?Name.
    ?s foaf:lastName ?Last .

    ?assessment  rdfs:label ?courseId .
    ?assessment  cpo:grade ?Grade .

    ?courseId schema:courseCode ?Course .
    ?courseId owl:sameAs ?link .
    ?link rdfs:label ?topic .

    ?courseId schema:courseCode ?Course .
    ?courseId owl:sameAs ?link .
    ?link rdfs:label ?Topic .

    #FILTER(?Grade != "F").

    FILTER (langMatches( lang(?Topic), "EN" ) ).
}
```

Result: (only the first 10 are shown here):

Name	Last	Topic	Grade
Tyler	Perry	"Programming language"	C
Tyler	Perry	"Object-oriented programming"	C
Tyler	Perry	"Database design"	C
Tyler	Perry	"Relational model"	C
Tyler	Perry	"Management system"	C
Tyler	Perry	"SQL"	C
Tyler	Perry	"Relational database"	C
Tyler	Perry	"Entity–relationship model"	C
Tyler	Perry	"Relational algebra"	C
Tyler	Perry	"Functional dependency"	C

4. The Conupedia Chatbot [Project 2]

4.1 Report Updates and Changes from Project 1

Given that the feedback received from the first version of this project and report were tremendously positive, only minor changes were made to the sections prior to section 4. Changes to the report prior to this point were limited to only minor grammatical corrections. Save for minor hickups in the hosting of the Conupedia database, which were accounted for and quickly resolved, the first iteration of the project was used exactly as it was submitted for review and grading.
Going forward, in order to finalize the project and create a functional grounding-based chatbot, the additions to the Conupedia project are described in section 4.2.

4.2 Chatbot Methods

On top of the tools specified in section 2.2 of this report, the following tools were created and used in order to create a functioning Conupedia Chatbot.

4.2.1 Session Initiatior

The session initiator, session.py, acts as a relay point between the chatbot's natural language interface and its source of information, the Conupedia database that was created for this purpose and which now in www.conupedia.sytes.net. In order to do so, this important tool initiates a connection over HTTP using Python's http library, and sends a GET request to the the Conupedia SPARQL endpoint. As such, while the Chatbot's interface method takes in questions as input from the user, it is the session initiator that holds the responsibility of retrieving that answer and passing it back to the interface to be parsed back into a response. For ease of parsing, the answer received by the SPARQL endpoint is stored into a python dictionary.

If an invalid query parameter is sent and the response received from Conupedia is anything other than the standard '200 OK' code, a ValueError is raised, stating what was done wrongly.

4.2.2 Chatbot Interface

The chatbot interface, chatbot.py, is the entry point for the user, which takes in questions written in natural language and, provided there is one, responds with the answer. It can be run via command line. The chatbot is capable of answering four distinct questions pertaining to information about Concordia University's courses and students, namely:

- What is <course> about?
- Which courses did <student> take?

- Which courses cover <topic>?
- Who is familiar with <topic>?

With each of the above being stored as a natural language question to SPARQL query pair, the chatbot interface is able to send a query to the session initiator tool. Before doing so, the question is put to lower case and stripped of its leading and trailing characters in order to ensure that no answer is overlooked because of these minor differences. Once this is done and the session initiator receives the question, it attempts to match it with one of the questions stored within its small bank of questions it is able to respond to, to in turn pass it to the Conupedia SPARQL endpoint and send back the response as soon as it is produced. After the session initiator's response has been received, the chatbot interface can then send it back to the user in the form of a natural language response.

The chatbot's question bank itself is a simple list of patterns, with the patterns being, themselves, lists of strings. The format of this list is [Question start, question end, SPARQL query, response structure]. For instance, the question *What is <course> about?* is associated with the tuple:

```
["what is "," about?","SELECT  ?Description WHERE { ?u schema:courseCode '\%s' .?u
schema:description ?Description .}", "\%s is described as being about \%s"]
```

By cross-referencing the question start and ending, the chatbot is immediately able to identify what question is being asked of it here, and knows that what is between this start and end is the subject of the question. Then, the third item of the list, the SPARQL query, is what gets sent via HTTP request to the SPARQL endpoint, with the subject of the question being the variable to query about. Last, with the response having been received, the fourth item of the list is used in order to plug in the subject in the response. Examples of questions and their resulting answers can be seen in section 4.3.

Should the chatbot fail to retrieve an answer, be it due to the question being badly formed or outside of the scope of the chatbot's capabilities, it instead produces the generic response "Sorry, I couldn't find anything to help you with that".

4.3 Sample Questions and Outputs

- **Which courses did Jane Doe take?**

  Jane Doe has taken SOEN321 where they were awarded a grade of D
  Jane Doe has taken COMP474 where they were awarded a grade of C-
  Jane Doe has taken COMP472 where they were awarded a grade of A-
  Jane Doe has taken SOEN331 where they were awarded a grade of A
  Jane Doe has taken ARTE354 where they were awarded a grade of D+

- **Who is familiar with Computers?**

  The topic of Computers is understood by Nick Lawson
  The topic of Computers is understood by Earl Jones
  The topic of Computers is understood by Lisa Boomer
  The topic of Computers is understood by Mike Larry
  The topic of Computers is understood by Andrea Viola
  The topic of Computers is understood by John Doe
  The topic of Computers is understood by Jane Doe

- **Which courses cover Database design?**

  Database design is a topic covered by BTM382
  Database design is a topic covered by COMP353

- **What is SOEN287 about?**

  SOEN287 is described as being about Internet architecture and protocols. Web applications through clients and servers. Markup languages. Client-side programming using scripting languages. Static website contents and dynamic page generation through server-side programming. Preserving state (client-side) in web applications. Lectures: three hours per week. Tutorial: two hours per week.

- **What is true love?**

  Sorry, I couldn't find anything to help you with that

Bibliography

Amazon. (2020). *Amazon Web Services.* Retrieved from https://aws.amazon.com/

Brickly, D., Miller, L. (2014). *FOAF Vocabulary Specification 0.99.* Retrieved from http://xmlns.com/foaf/spec/

Concordia University. (2020). *Concordia's Open Data.* Retrieved from https://www.concordia.ca/web/open-data.html/

Leipzing University, University of Mannheim. (2019). *DBpedia.* Retrieved from https://wiki.dbpedia.org/

OpenLink Software. (2020). *Virtuoso Open-Source Edition.* Retrieved from http://vos.openlinksw.com/owiki/wiki/VOS/

Tefincom & Co. (2020). *NordVPN.* Retrieved from https://nordvpn.com/

W3C Schema.org Community Group. (2020). *Schema.org.* Retrieved from http://www.https://schema.org/

World Wide Web Consortium. (2000). *Rdf Schema.* Retrieved from http://www.w3.org/2000/01/rdf-schema

World Wide Web Consortium. (2002). *Web Ontology Language.* Retrieved from http://www.w3.org/2002/07/owl

World Wide Web Consortium. (2001). *XML Schema.* Retrieved from http://www.w3.org/2001/XMLSchema