COMP 6721 Applied Artificial Intelligence (Winter 2022)

Worksheet #8: Knowledge Graphs & Intelligent Agents, Part II

N-Triples. Quick refresher: Using the N-Triples serialization format, write an RDF triple describing Concordia's *homepage*:

.....

Your first Vocabulary. Define the fact that `Student` is a `class` (as opposed to an instance, like *Jane*). Use the following prefix definitions and define `Student` as part of the `ex` namespace (`ex:Student`):

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix ex: <http://example.org/> .
```

Add the triple: .....

Creating Instances. Now add another triple stating that Jane (`ex:jane#me`) is of type `ex:Student`:

.....

Subclasses. For now at least, every *Student* is a *Person* (sorry, robots!). Define this fact as a triple (use `foaf:Person` for the namespace):

.....

Note: use the same `ex:` namespace for the new subclass as before for `Student`.

Are we there yet? Ok, let's look at these three triples (written in pseudocode for brevity):

```
<LS-210> <teaches> <COMP472/6721> .
<professor> <is a> <slide> .
<student> <handed in by> <assignment> .
```

Are these *syntactically* legal triples? (Spoiler alert: yes, we could write each of them using perfectly fine RDF URIs.) So what exactly is wrong here?

Construct	Syntactic form	Description
Class (a class)	**C** `rdf:type rdfs:Class`	**C** (a resource) is an RDF class
Property (a class)	**P** `rdf:type rdf:Property`	**P** (a resource) is an RDF property
type (a property)	**I** `rdf:type` **C**	**I** (a resource) is an instance of **C** (a class)
subClassOf (a property)	**C1** `rdfs:subClassOf` **C2**	**C1** (a class) is a subclass of **C2** (a class)
subPropertyOf (a property)	**P1** `rdfs:subPropertyOf` **P2**	**P1** (a property) is a sub-property of **P2** (a property)
domain (a property)	**P** `rdfs:domain` **C**	domain of **P** (a property) is **C** (a class)
range (a property)	**P** `rdfs:range` **C**	range of **P** (a property) is **C** (a class)

Properties. We now define a *property*, `studiesAt`, so that we can indicate at which university a student is studying. Write the triple defining `studiesAt` as a *property* (use the `ex:` namespace as before):

.....

(Note: properties should also have labels & comments, but we omit this here for brevity.)

Domain & Range. We now have to add *domain and range restrictions* for our property to avoid problems like the ones shown above. For the *domain* of our `studiesAt` property, we only permit `ex:Student` resources and for the *range*, we only admit `ex:University` resources. Write the two triples:

1. .....
2. .....

⌨ **FOAF.** A widely used vocabulary for describing people and their (social) networks is *Friend-of-a-Friend* (FOAF), which you've seen before:

```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
```

1. Assume Joe has a photo of him published under `http://facebook.me/joe.png` (not a real URL). How can you add this information to the knowledge graph using FOAF (hint: look up the vocabulary using the prefix URL above):

   .....

2. Again using FOAF, model that *Jane* is 22 years old:

   .....

⌨ **Linked Data.** How is Concordia University in the DBpedia knowledge graph *linked* to Wikidata? Find the *property* and *object* for:
<http://dbpedia.org/resource/Concordia_University> .....

⌨ **SPARQL.** Your first SPARQL query: What can you find in DBpedia with: (use the public SPARQL endpoint at https://dbpedia.org/sparql/):

```
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?s
WHERE { ?s geo:lat "45.497002"^^xsd:float .
        ?s geo:long "-73.578003"^^xsd:float . }
```

⌨ **Your own AI Agent.** Consider the output of a commercial AI, for example the *Google Assistant*, when you ask a question like *"What is Concordia University?"*: You'll typically see a definition as part of the answer that often comes from Wikipedia *("Concordia University, commonly referred to as Concordia, is a public comprehensive research university located in Montreal, Quebec, Canada...")*. Write a SPARQL query that retrieves this information from DBpedia:

```
SELECT ?desc
WHERE {

   . . .

}
```

To achieve this translation from question to query automatically, the AI needs an additional natural language processing (NLP) layer, which we'll cover later in this course.