COMP 474 UU, COMP 6741 UU 2204

```
Home / My courses / COMP-474-2204-UU / 14 February - 20 February / Lab Session #5
```

Lab Session #5

Introduction

Welcome to lab #5. This week we will continue working with SPARQL and knowledge graphs, moving up to the application layer.

Note: All instructions below assume you run Linux, your experience on MacOS or Windows may vary.

Follow-up Lab #4

Example solutions to the DBpedia queries from last week:

1. All universities located in Canada, with their city and optionally (if it exists) their home page:

2. All people who studied at Concordia University (and are listed in DBpedia), together with their description (in English):

```
SELECT ?name ?comment WHERE {
    ?cuperson dbo:almaMater dbr:Concordia_University .
    ?cuperson foaf:name ?name .
    ?cuperson rdfs:comment ?comment .
    FILTER (LANG(?comment) = 'en') .
} ORDER BY ?name
```

You might be surprised that you can run these queries even without any PREFIX declarations: That's because there are a <u>number of predefined prefixes</u> in the DBpedia query interface.

Task #1: Linked Open Data

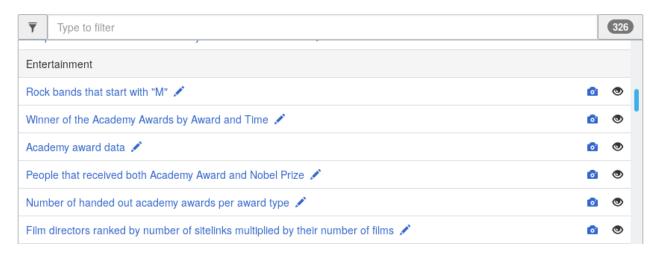
As you've seen, many <u>Linked Open Data</u> knowledge graphs provide a public SPARQL interface (a so-called *SPARQL endpoint*), for example:

- DBpedia
- Wikidata
- NextProt

Some only offer a dataset for download or have other restrictions (such as registering an API key).

Task #1.1: Wikidata

Try out some of the <u>query examples for Wikidata</u> provided at the top of the query page:



Note that Wikidata's SPARQL server offers a number of different result formats (click on the "Play" button on the left to run the query), e.g.:

1. List of parliament buildings with pictures by country (image grid)

2. Locations of Pablo Picasso works (map view)

```
#defaultView:Map

SELECT ?label ?coord ?subj

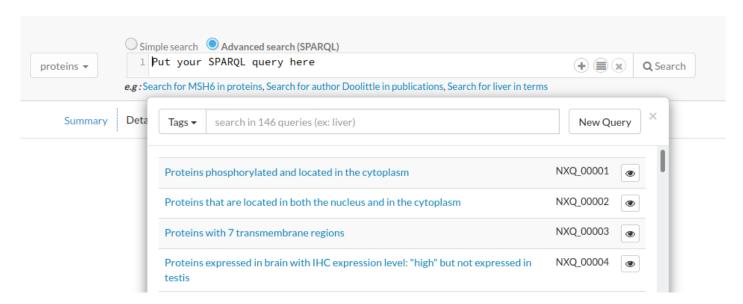
WHERE
{
    ?subj wdt:P170 wd:Q5593 .
    OPTIONAL {
        ?subj wdt:P276 ?loc .
        ?loc wdt:P625 ?coord } .
    ?subj rdfs:label ?label filter (lang(?label) = "en")
}
```

Now it's your turn, write own queries to:

- 1. List all the *member countries of the EU* with their name (look at the example queries above to see how to filter the label for a specific language in Wikidata)
- 2. Same as above, but print out the *capital* of each country on a map (look at the second example query above to see how to retrieve and render coordinates using a map view)

Task #1.2: Get your Protein fix here

Biomedical research was one of the first domains to adopt semantic technologies, since the huge amounts of data from experiments and publications is too much handle for human scientists: That's why you find some many life-science related datasets and ontologies on the LOD cloud.



Try experimenting with a few example queries provided at the <u>NextProt</u> SPARQL endpoint to see how a domain-specific LOD dataset looks like.

Task #2: Entity Tagging with DBpedia-Spotlight

Your next task is to experiment with <u>DBpedia-Spotlight</u> service discussed in the lecture. We already worked using the web-based demo interface during the lecture exercises, now your task is to understand it's <u>API</u>: Spotlight has a restful interface that you can access like this:

```
curl -X GET "https://api.dbpedia-spotlight.org/en/annotate?
text=Concordia%20advanced%20six%20spots%20to%2010th%20place%20among%20Canada%E2%80%99s%20engineering%20schools%20in%20the%20M
-H "accept: application/json"
```

The result is in JSON format:

```
"@text": "Concordia advanced six spots to 10th place among Canada's engineering schools in the Maclean's 2018 Program
Rankings, while computer science advanced three spots into 11th position this year.",
  "@confidence": "0.5",
  "@support": "0",
  "@types": "",
  "@sparql": "",
  "@policy": "whitelist",
  "Resources": [
      "@URI": "http://dbpedia.org/resource/Concordia University",
      "@support": "1835",
      "@types":
Wikidata:Q43229,Wikidata:Q3918,Wikidata:Q24229398,DUL:SocialPerson,DUL:Agent,Schema:Organization,Schema:EducationalOrganizat
      "@surfaceForm": "Concordia",
      "@offset": "0",
      "@similarityScore": "0.9999782207698248",
      "@percentageOfSecondRank": "1.995676280252352E-5"
    },
    {
      "@URI": "<a href="http://dbpedia.org/resource/Canada"">http://dbpedia.org/resource/Canada</a>",
      "@support": "220739",
      "@types":
/"Wikidata:Q6256,Schema:Place,Schema:Country,DBpedia:PopulatedPlace,DBpedia:Place,DBpedia:Location,DBpedia:Country",
      "@surfaceForm": "Canada",
      "@offset": "49",
      "@similarityScore": "0.9997965603941569",
      "@percentageOfSecondRank": "1.318801347665404E-4"
    },
      "@URI": "<a href="http://dbpedia.org/resource/Computer_science"">http://dbpedia.org/resource/Computer_science</a>",
      "@support": "11027",
      "@types": "",
      "@surfaceForm": "computer science",
      "@offset": "124",
      "@similarityScore": "0.9999981246676802",
      "@percentageOfSecondRank": "6.422167580532429E-7"
    }
  ]
```

Make sure you understand the conceptual difference between the *surface form*, which appears in the text you are analyzing, and the link (URI) itself. This is where ambiguities need to be resolved: for example, the surface form "Hilton" could be linked to the company or the person "Paris Hilton", depending on the context.

Try writing a simple Python program that accepts a query string, sends a REST request and displays the detected entities.

Note that for frequent requests, you should setup your own Spotlight server. An easy way to install one is to use the provided *Docker* image, which you can find on its <u>Github page</u>.

Task #3: Even Smarter University Agent

Now you are ready to spruce up your intelligent university agent from the previous week: Give it a simple natural language interface that, for now, only recognizes the pattern "Who is <X>". Obtain <X> using some simple regex-pattern matching from the input string and create a SPARQL query that extracts the answer from your knowledge graph. You can then have a dialog like this:

```
Hello, I am your smart university agent. How can I help you?

> Who is Joe?

Joe is 22 years old and has the email address joe@example.com.
```

Bonus task: Also support the query "What is <X>", where you search for <X> using a suitable SPARQL query on DBpedia, printing out the comment:

```
Hello, I am your smart university agent. How can I help you?

> What is Concordia University?

Concordia University (commonly referred to as Concordia) is a public comprehensive university located in

Montreal, Quebec, Canada. Founded in 1974 following the merger of Loyola College and Sir George Williams University...
```

Now you are almost as good as the Google Assistant!

That's all for this lab!

Last modified: Monday, 15 February 2021, 7:26 PM

■ Worksheet #05

Jump to...

Lecture Slides #07 ▶

You are logged in as Divya Rajendra Bhosale (Log out) COMP-474-2204-UU

Academic Integrity

Academic Assessment Tool

English (en)

Deutsch (de)

English (en)

Español - Internacional (es)

Français (Canada) (fr_ca)

Français (fr) Italiano (it)

(ar) العربية