IIC3432 — Advanced topics in databases

# Linked Data

## 1. Setting up the scene

In this class we will explore how to query Linked Open Data using a simple Python library called RDFLib.

For this you will have to be using Python: `https://www.python.org/downloads/`

To install RDFLib, it is probably easiest to use pip:

```
$ pip install rdflib
```

If you don't like this option go to `https://github.com/RDFLib/rdflib` and get the library manually.

## 2. Basic commands

The main command we will need to use in RDFLib is loading a resource from an online location, and iterating over the triples:

```
import rdflib
g=rdflib.Graph()
g.load('http://dbpedia.org/resource/Semantic_Web')

for s,p,o in g:
    print s,p,o
```

We will also often need to compare an URI/literal to a string; this can be done using the URIRef command as follows:

```
import rdflib
from rdflib import URIRef

g=rdflib.Graph()
g.load('http://dblp.l3s.de/d2r/resource/authors/Cristian_Riveros')

for s,p,o in g:
    if (p == URIRef("http://xmlns.com/foaf/0.1/name")):
        print(s,p,o)
```

Apart from these basic commands, you can find more detailed examples and documentation of RDFLib at:

`http://rdflib.readthedocs.io/en/stable/index.html`

# 3. Graph patterns over Linked Data

We will start by exploring the Linked Data version of DBpedia. For this, you should load the resource for Berlin:
`http://dbpedia.org/resource/Berlin`
Using this as your entry point into the Linked Data world of DBpedia, try to get the answer to following queries:

1. The population of Berlin.
2. The oldest person born in Berlin.
3. The list of all the people that were born in Berlin and that studied in a university that is located in the US.

Now try to formulate the above queries in SPARQL and answer them over the DBPedia endpoint. Do you get the same answers?

# 4. Path queries over DBPedia

Recall from last time that the query computing the Bacon Number in DBpedia did not run very efficiently. In this task, our objective is to explore how to implement this query over Linked Data.
We will split this task into several subtasks you should tackle:

1. Load the IRI of Kevin Bacon using RDFLib.
2. Find all the actors with Bacon Number 1.
3. Find all the actors with Bacon Number 2.
4. Can you make the previous query more efficient? In particular, if you implemented it in an obvious way, then you will load the same movie many times. Perhaps you can store the information you already have and check later if you visited it. There are two ways to do this: (i) you can create your own data structures to store the data; and (ii) RDFLib allows you to define your database and manipulate it; check the documentation above to see how.
5. Using the above method for avoiding unnecessary requests to an IRI, try to execute the query looking for actors with a Bacon number. Stop when you find 500 actors, or when more than 5 minutes went on.
6. Try to implement the Bacon query using some standard graph search algorithm if you have not done this already. Use both DFS and BFS.
7. Can you return paths when you run BFS/DFS? What is the difference in paths between the two algorithms.

If problems 6 and 7 are taking too much time move to the next section and finish them at home :)

# 5. Queries in DBLP

DBLP is an online repository of Computer Science publications. It contains information about the authors and papers they have written. This allows us to examine how some authors are connected through collaboration links, similarly to the notion of Bacon number.
You can access the DBLP bibliography at `http://dblp.uni-trier.de/` and search for the publications of your favourite Computer Scientist. DBLP has a SPARQL endpoint (sort of) at `http://dblp.l3s.de/d2r/snorql/`. Using RDFLib try to answer the following queries:

- Find all the papers written by this guy: `http://dblp.l3s.de/d2r/resource/authors/Cristian_Riveros`. Using your browser to access the above URI will give you a quick view of triples connected to this entity.
- Now find all the papers published in conference proceedings by the same author.
- Find all his co-authors.
- Find the co-authors of his co-authors (people with Riveros number 2).
- Find the number of papers written by people with Riveros number 2. Make sure you eliminate the duplicates.
- Find all the authors with Riveros number (stop when you reach 100, or run over 5 minutes).
- Do the same, but starting with the following URI: `http://dblp.l3s.de/d2r/resource/authors/Michael_Stonebraker`. Which one executes faster?
- As in the case of DBpedia, compare how different implementations of this query perform when using BFS/DFS to traverse the Linked Data cloud.