Python programming — Semantic Web

Finn Årup Nielsen

DTU Compute
Technical University of Denmark

October 14, 2014
What is Semantic Web?

Semantic Web =

Triple data structure (representing subject, verb and object)

+ URIs to name elements in the triple data structure

+ standards (RDF, N3, SPARQL, ...) for machine readable semi-structured data.
Why the Semantic Web?

IBM’s Watson supercomputer destroys all humans in Jeopardy

http://www.youtube.com/watch?v=WFR3I0m_xhE

“[...] they can build confidence based on a combination of reasoning methods that operate directly on a combination of the raw natural language, automatically extracted entities, relations and available structured and semi-structured knowledge available from for example the Semantic Web.” — http://www.research.ibm.com/deepqa/faq.shtml
Example triples

<table>
<thead>
<tr>
<th>Subject</th>
<th>Verb</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>neuro:Finn</td>
<td>a</td>
<td>foaf:Person</td>
</tr>
<tr>
<td>dbpedia:Charlie_Chaplin</td>
<td>foaf:surname</td>
<td>Chaplin</td>
</tr>
<tr>
<td>dbpedia:Charlie_Chaplin</td>
<td>owl:sameAs</td>
<td>fbase:Charlie Chaplin</td>
</tr>
</tbody>
</table>

Table 1: Triple structure

where the the so-called “prefixes” are

PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX neuro: <http://neuro.imm.dtu.dk/resource/>
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
DBpedia

DBpedia extracts semi-structured data from Wikipedias and map and add the data to a triple store.

The data is made available on the Web is a variety of ways: http://dbpedia.org

DBpedia names (URIs), e.g., http://dbpedia.org/resource/John_Wayne

Human readable page, e.g., http://dbpedia.org/page/John_Wayne

Machine readable, e.g., http://dbpedia.org/data/John_Wayne.json
Query DBpedia

SPARQL endpoint for DBpedia:

http://dbpedia.org/sparql

Get pharmaceutical companies with more than 30'000 employees:

```
    ?Company dbpprop:industry ?industry ;
    dbpprop:numEmployees ?numEmployees ;
    foaf:page ?page .
    FILTER (?industry = dbpedia:Pharmaceutical_industry ||
            ?industry = dbpedia:Pharmaceutical_drug) .
    FILTER (?numEmployees > 30000) .
}
ORDER BY DESC(?numEmployees)
```
Linked Data cloud

Huge amount of interlinked data where DBpedia is central

Media, geographical, publications, user-generated content, government, cross-domain, life sciences.

Figure 1: Part of Linking Open Data cloud diagram, by Richard Cyganiak and Anja Jentzsch. CC-BY-SA.
And what can Python do with this Semantic Web?
Python

Query existing triple stores, e.g., DBpedia

Setup a triple store
Getting data from DBpedia

URI for municipality seats in Denmark:

url = "http://dbpedia.org/resource/Category:Municipal_seats_of_Denmark"

Get the data in JSON with "Content-Type" negotiation:

```
import urllib2, simplejson
opener = urllib2.build_opener()
opener.addheaders = [('Accept', 'application/json')]
seats = simplejson.load(opener.open(url))
```

Get the URIs for the municipality seats:

```
uris = [k for k,v in seats.items() if "http://purl.org/dc/terms/subject" in v]
```
**Getting data from DBpedia**

URI for municipality seats in Denmark:

url = "http://dbpedia.org/resource/Category:Municipal_seats_of_Denmark"

Get the data in JSON with "Content-Type" negotiation using the more elegant requests module:

```python
import requests

seats = requests.get(url, headers={'Accept': 'application/json'}).json()
```

Get the URIs for the municipality seats:

```python
uris = [k for k,v in seats.items()
        if "http://purl.org/dc/terms/subject" in v]
```
Get one of the geographical coordinates associated with the first municipality seat by querying DBpedia again, now with a URI for the seat:

```python
seat = simplejson.load(opener.open(uris[0]))
geo = "http://www.w3.org/2003/01/geo/wgs84_pos#"
l_lat = seat[uris[0]][geo + 'lat'][0]['value']
long = seat[uris[0]][geo + 'long'][0]['value']
```

Show the coordinate on an OpenStreetMap map:

```python
url_map = ('http://staticmap.openstreetmap.de/staticmap.php?center=%f,%f' '
  &zoom=8&size=300x200&maptype=mapnik") % (lat, long)
import PIL.Image
import StringIO
buf = urllib2.urlopen(url_map).read()
im = PIL.Image.open(StringIO.StringIO(buf))
im.show()
```
In this case the first municipality seat returned from DBpedia was Hvorslev:

```python
>>> uris[0]
'http://dbpedia.org/resource/Hvorslev'
>>> lat
56.15000152587891
>>> long
9.767000198364258
```

And the generated image:
Construct SPARQL URL for DBpedia

SQL-like SPARQL is the query language in Semantic Web web services.

As an example, formulate a query in SPARQL language for information about pharmaceutical companies with more than 30’000 employees:

```sparql
>>> query = ""
    ?Company dbpprop:industry ?industry ;
    dbpprop:numEmployees ?numEmployees ;
    dbpprop:revenue ?revenue ;
    foaf:name ?name ;
    foaf:isPrimaryTopicOf ?page .
    FILTER (?numEmployees > 30000) .
    FILTER (?numEmployees < 30000000) .
} ""
```
Query the DBpedia so-called “endpoint” for data in CSV format:

```python
>>> import urllib
>>> param = urllib.urlencode({'format': 'text/csv',
                             'default-graph-uri': 'http://dbpedia.org',
                             'query': query})
>>> endpoint = 'http://dbpedia.org/sparql'
>>> csvdata = urllib.urlopen(endpoint, param).readlines()
```

Read the csv data into an array of dictionaries:

```python
>>> import csv
>>> columns = ['uri', 'employees', 'revenue',
             'industry', 'name', 'wikipedia']
>>> data = [dict(zip(columns, row)) for row in csv.reader(csvdata[1:])]
```

There is a non-uniqueness issue because of multiple foaf:names

```python
>>> data = dict([(d['uri'], d) for d in data])
```
Now we got access to the information about the companies, e.g., number of employees:

```python
>>> [d['employees'] for d in data][:6]
```

However, the DBpedia extraction from Wikipedia might not always be easy to handle, e.g., the revenue has different formats and possible unknown currency:

```python
>>> [d['revenue'] for d in data][:12]
 ‘\xc2\xa56,194.5 billion’]
```

(note here is missing the coding of UTF-8 \xc2\xa5 to the Yen sign)

Furthermore, information in Wikipedia (and thus DBpedia) is not necessarily correct.
Reading data with Pandas

Reading of the returned data from DBpedia's SPARQL endpoint make the code a bit cleaner:

```python
>>> import pandas as pd

>>> data = pd.read_csv(endpoint + '?' + param)
>>> data.drop_duplicates(cols='Company')
>>> data[['Company', 'numEmployees', 'revenue']].head(3)

<table>
<thead>
<tr>
<th>Company</th>
<th>numEmployees</th>
<th>revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://dbpedia.org/resource/Pfizer">http://dbpedia.org/resource/Pfizer</a></td>
<td>91500</td>
<td>US$ 58.98 billion</td>
</tr>
<tr>
<td><a href="http://dbpedia.org/resource/Merck_&amp;_Co">http://dbpedia.org/resource/Merck_&amp;_Co</a>.</td>
<td>86000</td>
<td>US$ 48.047 billion</td>
</tr>
<tr>
<td><a href="http://dbpedia.org/resource/Novartis">http://dbpedia.org/resource/Novartis</a></td>
<td>119418</td>
<td>US $58.566 billion</td>
</tr>
</tbody>
</table>
```

Note the data from DBpedia is still dirty, because of the difficulty with extracting data from Wikipedia.
You can also store your own data in Semantic Web-like data structures
Setup up a triple store

See Python Semantic Web book (Segaran et al., 2009)

Simple triple store without the use of URIs:

```python
>>> triples = [('Copenhagen', 'is_capital_of', 'Denmark'),
             ('Stockholm', 'is_capital_of', 'Sweden'),
             ('Copenhagen', 'has_population', 1000000),
             ('Aarhus', 'is_a', 'city'),
             ('Copenhagen', 'is_a', 'capital'),
             ('capital', 'is_a', 'city')]
```

Query the triple store (the Python variable `triples`) for capitals:

```python
>>> filter(lambda (s,v,o): v=='is_capital_of', triples)
[('Copenhagen', 'is_capital_of', 'Denmark'),
 ('Stockholm', 'is_capital_of', 'Sweden')]
```
Python Semantic Web package: rdflib

Example using rdflib (Segaran et al., 2009, Chapter 4+)

>>> import rdflib
>>> from rdflib.Graph import ConjunctiveGraph
>>> g = ConjunctiveGraph()
>>> for triple in triples: g.add(triple)

Query the triple store with the triples() method in the ConjunctiveGraph() class:

>>> list(g.triples((None, "is_capital_of", None)))
[('Stockholm', 'is_capital_of', 'Sweden'),
 ('Copenhagen', 'is_capital_of', 'Denmark')]
Wikidata
**Wikidata/Wikibase**

Recent effort to structure Wikipedia’s semistructured data

Multilingual so each label and description may be in several languages.

*Wikibase* is the program for MediaWiki

Instance on [wikidata.org](http://wikidata.org) under Wikimedia Foundation for Wikipedia

Wikidata have more pages than Wikipedia.
Growth in Wikidata

From Wikidata item creation progress no text (Pyfisch, CC-BY-SA)
Wikidata data model

Entity: Either an “item” (Example: the gene Reelin: Q414043) or a “property”

1. Item
   (a) Item identifier, e.g., “Q1748” for Copenhagen
   (b) Multilingual label, e.g., “København”, “Copenhagen”
   (c) Multilingual description, “Danmarks hovedstad”
   (d) Multilingual aliases
   (e) Interwikilinks (links between difference language versions of Wikipedia)
(f) **Claims**

i. **Statement**

   A. *Property*, e.g., “GND-type” (P107)
   
   B. *Property value*, e.g., “geographical object”

   C. *Qualifiers*

ii. **Reference**

2. **Property**

   (a) *Property identifier*

   (b) Multilingual label

   (c) Multilingual description

   (d) Multiplilingual aliases

   (e) Datatype
Reasonator: Online rendering of Wikidata data

Johann Sebastian Bach (Q1339)

German composer, organist, harpsichordist, violinist, and violinist

Johann Sebastian Bach was a German composer, organist, and musician.

He was born on March 31, 1685 in Eisenach to Johann Ambrosius Bach and Maria Elisabeth Lämmerhirt.

He studied at St. Michael's School. His field of work included classical music and baroque music. He was a member of the Bach family. He worked for Divi Blasi, Mühlhausen, for Augustus III of Poland, for Leopold, Prince of Anhalt-Köthen, for Johann Ernst III, Duke of Saxe-Weimar from January 1703 until August 1703, for Johann Ernst III, Duke of Saxe-Weimar, for Thomaschule zu Leipzig, and for Bachkirche Arnstadt from August 1703 until 1707.

He married Maria Barbara Bach on October 17, 1707 (married until in 1720) and Anna Magdalena Bach on December 3, 1721. His children include Catharina Dorothea Bach, Wilhelm Friedemann Bach, Carl Philipp Emanuel Bach, Johann Gottfried Bernhard Bach, Gottfried Heinrich Bach, Johann Christoph Friedrich Bach, and Johann Christian Bach. He died of stroke and pneumonia on July 28, 1750 in Leipzig. He was buried at St. Thomas Church.

Relatives

Parents

father  Johann Ambrosius Bach
mother  Maria Elisabeth Lämmerhirt

Siblings

brother  Johann Jacob Bach
brother  Johann Christoph Bach

Children

child  Wilhelm Friedemann Bach
child  Carl Philipp Emanuel Bach
child  Johann Christian Bach
child  Johann Gottfried Bernhard Bach
child  Johann Christoph Friedrich Bach

Other

relative Christoph Bach

spouse Anna Magdalena Bach

See the full family tree: inline/new page

External sources

BMLO  b1316
BNF  118897907
BNE  XX982838
GND  11850002X
IMDb  nm0001925
ISNI  0000 0001 2276 4157
Freebase /m/03_10
Find a Grave ID 4237
LCCN  n79021425
MusBrainz  2411766e-9e35-4d58-9464-9413595b84c4
NDL  00432003
NLA  35011573
NTA PPN 068721781
NKC  jj19990000387
KGB  ru/1419990000387

Finn Årup Nielsen  25  October 14, 2014
Programmer’s interface

Ask for Copenhagen (Q1748), get multilingual element in Danish and JSON:

http://wikidata.org/w/api.php?
    action=wbgetentities & ids=Q1748 & languages=da & format=json

What is the country of Copenhagen:

import requests
url = "http://wikidata.org/w/api.php?" + \
    "action=wbgetentities&ids=Q1748&languages=da&format=json"
response = requests.get(url).json()
property = response[‘entities’][’Q1748’][’claims’][’P17’][0]
property[‘mainsnak’][‘datavalue’][‘value’][‘numeric-id’]

Gives “35” (Q35=Denmark).
pywikibot interface

After setup (of user-config.py) you can do:

```python
>>> import pywikibot

>>> data = pywikibot.DataPage(42)
>>> dictionary = data.get()
>>> dictionary['label']['de']
u'Douglas Adams'
>>> [claim['m'][3]['numeric-id'] for claim in dictionary['claims']
    if claim['m'][1] == 21][0]
6581097
>>> print(pywikibot.DataPage(6581097).get()['label']['ro'])
bărbat
```

Data item number 42 is something called “Douglas Adams” in German which has the sex/gender “bărbat” (male) in Romanian.
**pywikibot interface**

Note the pywikibot API is unfortunately shaky. You might have to do:

```python
g>>> import pywikibot
g>>> site = pywikibot.Site('en')
g>>> repo = site.data_repository()
g>>> item = pywikibot.ItemPage(repo, 'Q42')
g>>> _ = item.get()  # This is apparently necessary!
g>>> item.labels['de']
g   u'Douglas Adams'
g>>> target_item = item.claims['P21'][0].target
g>>> _ = target_item.get()
g>>> target_item.labels['ro']
g   u'bărbat'
```

This is for the branch presently called `core`. 
**Wikidata tools**

Using Magnus Manske’s tool to get Danish political parties with a Twitter account

```python
>>> import requests
>>> url_base = "https://wdq.wmflabs.org/api?q="
>>> items = requests.get(url_base + query).json()['items']
>>> items
[25785, 212101, 217321, 478180, 507170, 615603, 902619, 916161]
```

These numbers are Wikidata identifiers for the Danish political parties, e.g., [https://www.wikidata.org/wiki/Q25785](https://www.wikidata.org/wiki/Q25785) is the Red-Green Alliance.

Query to be read: instance of political party and country Denmark and website account on Twitter
url_base = ('http://wikidata.org/w/api.php?'
    'action=wbgetentities&format=json&ids=Q')
for item in items:
    party = requests.get(url_base + str(item)).json()['entities'].values()[0]
    label = party['labels']['en']['value']
    account = ''
    for claim in party['claims']['P553']:
        if claim['mainsnak']['datavalue']['value']['numeric-id'] == 918:
            # Twitter == 918
            try:
                account = claim['qualifiers']['P554'][0]['datavalue']['value']
            except IndexError, KeyError:
                pass
            break
    print('{}: https://twitter.com/{}'.format(label, account))

It gets the parties from the ‘ordinary’ API and produces the output:

Red-Green Alliance: https://twitter.com/Enhedslisten
Social Democrats: https://twitter.com/Spolitik
Venstre: https://twitter.com/Venstredk
...

Finn Årup Nielsen
More information and features

Book about Semantic Web and rdflib: (Segaran et al., 2009)

rdflib can read N3 and RDF file formats

rdflib can handle namespaces.

There are dedicated triple store databases, e.g., Virtuoso.
Summary

You can get large amount of background information from the Semantic Web & Co.