UE 803 - Data Science

Session 3: Webservices

Yannick Parmentier - Université de Lorraine / LORIA

Introduction

Webservice

> A service offered **by an electronic device to another electronic device**, *communicating with each other via the World Wide Web*.

Source: Wikipedia

Introduction

Webservice

> A service offered **by an electronic device to another electronic device**, *communicating with each other via the World Wide Web*.

Source: Wikipedia

That is, instead of conveying information to be processed by humans, with webservices **HTTP requests are used to deliver information to be processed by machines**

Introduction (continued)

- **Web-server**: Program serving web pages (HTML) (e.g. `Apache`)
- **Web-service** : Program serving data (e.g. XML, JSON) (e.g. `http://open-notify.org`)
- Web-services provide data through **endpoints** (URL) (e.g. `http://api.open-notify.org/iss-now.json`)
- Each endpoint provides functionalities documented in its so-called **Application Programming Interface**/API (e.g. `http://open-notify.org/Open-Notify-API/ISS-Location-Now/`)

Introduction (continued)

- Web-services may require **authentication** (e.g. sensitive data providers)
- Web-services complying with **RESTful** architectures are accessible via **HTTP queries** (e.g. `http://api.open-notify.org/iss-pass.json?lat=48.684&lon=6.1850`)
- RESTful web-services may be accessed **manually** (e.g. within a web-browser) or **programmatically**, e.g.:

```
parameters = {"lat": 48.684, "lon": 6.1850}
response = requests.get("http://api.open-notify.org/iss-pass.json",
     params=parameters)
data = response.json()
print(data)
```

Introduction (continued)


```
{
  "message": "success",
  "request": {
    "altitude": 100,
    "datetime": 1547320486,
    "latitude": 48.684,
    "longitude": 6.185,
    "passes": 5
  },
  "response": [
    {
      "duration": 592,
      "risetime": 1547331231
    },
    {
      "duration": 645,
      "risetime": 1547336976
    }
  ]
}
```

Outline

1. **Common data exchange formats**
   (introducing CSV, JSON, YAML)

2. **The Wikipedia API**
   (introducing wikipedia and wptools python wrappers)

3. **The Wikidata API**
   (introducing Open Linked Data)

Common data formats

Introducing CSV, JSON, YAML and more

Common data formats

Why using a common data format ?

Common data formats

Why using a common data format ?

→ To save time (no need to develop an ad-hoc language and parser for reading your data)!

Common data formats

Why using a common data format ?

→ To save time (no need to develop an ad-hoc language and parser for reading your data)!

→ To make your data readable by other programs (*aka* interoperability)!

Comma-Separated Values (CSV)

- **Plain text** format used to store datasets as **records**:
  - each data **record** is stored on a **single line**
  - **fields** of a record are **strings separated by a specific user-defined character** (default being comma)
- **Open** format (RFC 4180)
- **Lightweight** table-like format
- **Compatible** with spreadsheet processors / **native support** for many programming languages

Comma-Separated Values (CSV, continued)

- **Example** (beware of **encoding**, cf non-ASCII characters, especially newline, and of trailing **whitespaces**):

```
Firstname,Lastname,Year of birth
Alphonse,Dupont,1932
Béatrice,Durand,1964
Charlotte,Lamarque,1988
```

- *Pro*: **simple** format
  Cons: **poor** (flat, untyped) structure

JavaScript Object Notation (JSON)

- **Plain text** format used to store datasets as **objects** (*key-value pairs*) such that:
  - **keys** are unique and **ordered**
  - a **value** can be *null, a string, a number, a boolean, a list of values [...], or another object {...}*
    → **recursive structure**
- **Open** format (RFC 8259)
- **Expressive** dictionary-like format
- (Often natively) **Compatible** with webservices and programming languages

JavaScript Object Notation (JSON, continued)

- **Example**:

```
{  "menu": {
    "id": "file",
    "popup": {
        "menuitem": [
            { "value": "New", "onclick": "CreateNewDoc()" },
            { "value": "Close", "onclick": "CloseDoc()" } ] } }
}
```

- *Pro*: **rich** (recursive, typed) structure
  Cons: **lack of human readability**

Yet Another Markup Language (YAML)

- **Plain text** format used to store datasets as **values**:
  - values are separated with `---` (multi-rooted document)
  - a **value** can be a *scalar (null, string, number, boolean), a list of values, or key-value pairs* (i.e. a dictionary) → **recursive structure**
- **Open** format (Spec. 1.2)
- **Lightweight**, **expressive** and **readable** format
- **Compatible** with many programming languages (through dedicated libraries)

Yet Another Markup Language (YAML, continued)

- **Example**:

```yaml
menu:
  id: file
  popup:
      menuitem:
          - value: New
            onclick: CreateNewDoc()
          - value: Close
            onclick: CloseDoc()
```

- *Pro*: **readable**, **rich** structure
  Cons: **sensitivity to whitespaces**

More

- CSV, JSON and YAML all are **text-based** formats:
  - **Human-readable** 👍
  - Read / write requires **dedicated libraries** 👍 👎
  - Read / write may be **time consuming** 👎
  - **Expressivity is limited** 👎

More

- CSV, JSON and YAML all are **text-based** formats:
  - **Human-readable** 👍
  - Read / write requires **dedicated libraries** 👍 👎
  - Read / write may be **time consuming** 👎
  - **Expressivity is limited** 👎

- Need for more → **serialization**

More (continued)

- **Serialization** is the process of saving data in a **binary format** as a snapshot of the memory:
  - Provides **full expressivity** 👍
  - **No encoding / decoding** required 👍
  - Read / write is **lightning fast** 👍
  - Data is **no longer human-readable, neither interoperable directly** 👎
- Supported natively in Python via the `pickle` module:

```
>>> import pickle
>>> pickle.dump(data_out, file)    #write
>>> data_in = pikcle.loads(file)   #read
```

The Wikipedia API

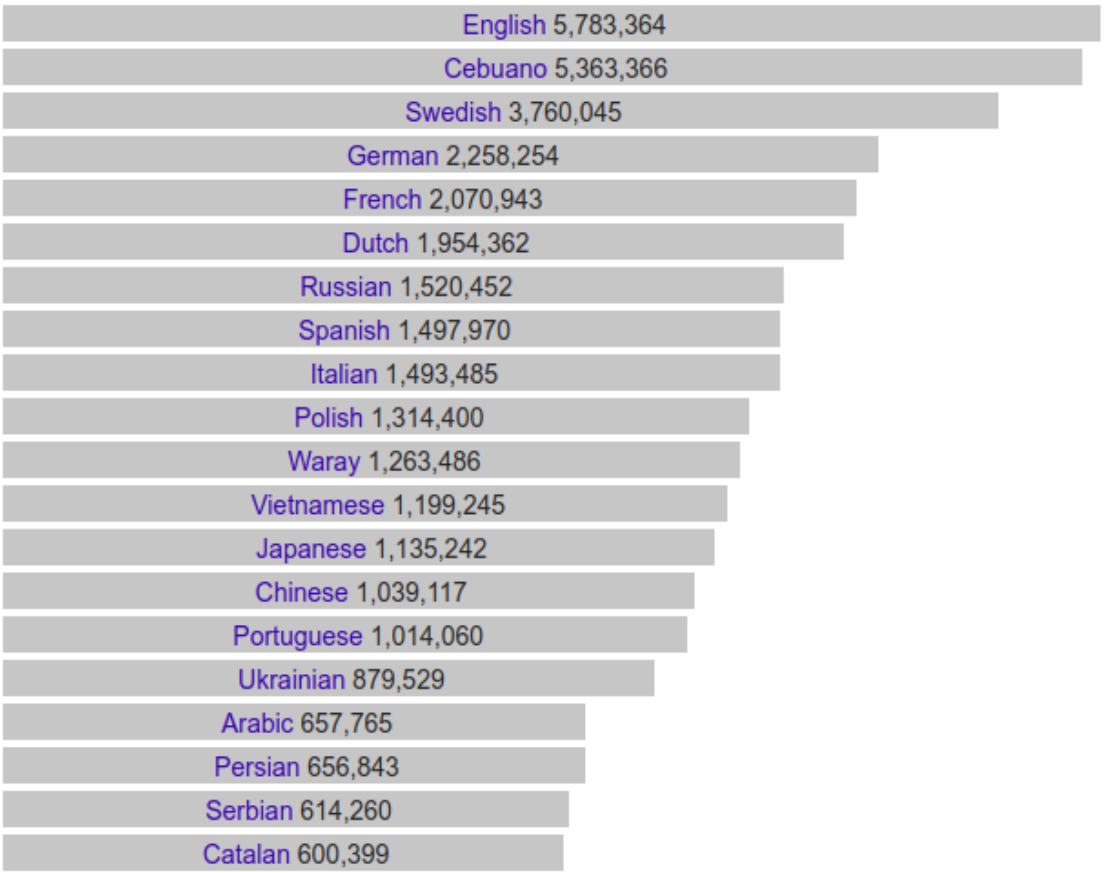
Introducing wikipedia and wptools

About Wikipedia

- **Wikipedia**: web-based, multilingual, collaborative, free encyclopedia created in 2001 (with 291 languages and more than 38 million articles in 2015)

- **Mediawiki**: open-source PHP wiki platform (dynamic and collaborative webpage edition) used by Wikipedia

About Wikipedia (continued)



Number of pages per language as of Jan. 2019

About Wikipedia (continued)

- Various **page types** (aka *namespaces*), including:
  - Main/Article
  - User page
  - Talk
  - File
  - MediaWiki
  - Template
  - Help
  - Category
  - Portal
  - Book

Wikipedia pages: general layout

- Articles have a **title** (bound to their URL)
- Usually consist of **paragraphs** and **images** (perhaps with other types of audiovisual **media**)
- May also be stand-alone **lists or tables**
- Contain *references* and **links**
- May contain *summary* tables (aka **infoboxes**)

Wikipedia pages: Infoboxes

```
{{Infobox prepared food
| name             = Crostata
| image            = Crostata limone e zenzero 3.jpg
| imagesize        =
| caption          = Crostata with lemon ginger filling
| alternate_name   =
| country          = [[Italy]]
| region           = [[Lombardia]]
| creator          =
| course           = [[Dessert]]
| type             = [[Tart]]
| served           =
| main_ingredient  = Pastry crust, [[jam]] or [[ricotta]], fruit
| variations       = "Crostata di frutta", "crostata di ricotta"...
| calories         =
| other            =
}}
```

Crostata

Crostata with lemon ginger filling

Type	Tart
Course	Dessert
Place of origin	Italy
Region or state	Lombardia
Main ingredients	Pastry crust, jam or ricotta, fruit
Variations	*Crostata di frutta*, *crostata di ricotta*, many other sweet or savoury variations

Cookbook: Crostata Media: Crostata

Related projects

- **Wikipedia dumps**: off-line access to the encyclopedia (in various formats)
- **DBpedia**: *structured content* (database) semi-automatically extracted from Wikipedia
- **Wikidata**: *knowledge-base* underlying (among others) the Wikipedia encyclopedia

→ Resources used in many research projects
(e.g. Universal Dependencies)

The Wikipedia APIs: `wikipedia` and `wptools`

- **Python wrappers** for the Mediawiki API
- These offer an **easy access** to Wikipedia data / webservice (e.g. no URL handling needed)
- **Functionalities** include:
  - searching Wikipedia
  - getting article summaries
  - getting links
  - getting images
  - etc.

`wikipedia`: basic search

- Installation:

```
conda install -c conda-forge wikipedia
```

- Usage:

```
>>> import wikipedia

>>> wikipedia.search("Barack")
['Barack Obama', 'Barack Obama Sr.', 'Family of Barack Obama', ...]

>>> wikipedia.search("Ford", results=3)
['Ford Motor Company', 'Gerald Ford', 'Henry Ford']

>>> wikipedia.summary("Apple III", sentences=1)
'The Apple III (often rendered as Apple ///) is a business-oriented
```

- Link to `wikipedia` documentation 🔗

`wikipedia`: ambiguous search

- In case of ambiguous search, an error is raised:

```
>>> wikipedia.summary("Mercury")
Traceback (most recent call last):
...
wikipedia.exceptions.DisambiguationError: "Mercury" may refer to:
Mercury (element)
Mercury (mythology)
Mercury (planet)

>>> try:
...     mercury = wikipedia.summary("Mercury")
... except wikipedia.exceptions.DisambiguationError as e:
...     print e.options
...
['Mercury (element)', 'Mercury (mythology)', 'Mercury (planet)', ..
```

`wikipedia`: information retrieval

- Some available pieces of information:

```
>>> ny = wikipedia.page("New York City")
>>> ny.title
'New York City'
>>> ny.url
'http://en.wikipedia.org/wiki/New_York_City'
>>> ny.content
'The City of New York, often called New York City (NYC) or simply N
>>> ny.images[0]
'https://upload.wikimedia.org/wikipedia/commons/3/3d/10_mile_panora
>>> ny.links[0]
'10 Hudson Yards'
```

- To define a specific target language:

```
>>> wikipedia.set_lang("fr")
>>> print(wikipedia.summary('Francois Hollande'))
'François Hollande [fʁɑ̃swa ʔɔlɑ̃d] , né le 12 août 1954 à Rouen, est
```

`wikipedia`: introducing `WikipediaPage` objects

```
class wikipedia.WikipediaPage(title=None, \
   pageid=None, redirect=True, preload=False, \
   original_title='')

   self.content
   self.coordinates
   self.images
   self.links
   self.references
   self.section(section_title)
   self.sections
   self.summary
   self.revision_id
   self.html()
```

(see documentation 🔗)

`wptools`: basic search

- Installation:

```
conda remove zeromq #cf https://github.com/jupyter/notebook/issues/
conda install zeromq
conda install conda-build
conda install pycurl

pip3 install wptools
```

- Usage:

```
>>> import wptools
>>> page = wptools.page('crostata', lang='it') #Preparing query

>>> page.get_query() #Fetch information from Mediawiki:Query API
>>> page.get_parse() #OR fetch info from Mediawiki:Parse API
>>> page.get()       #OR fetch Wikipedia and wikidata info
>>> page.get_more()  #OR fetch more (expensive) info
```

- Link to `wptools` documentation 🔗

`wptools`: basic search (continued)

- Output of `get_query()`: `data` object

```
it.wikipedia.org (query) crostata
it.wikipedia.org (imageinfo) File:Crostata limone e zenzero 3.jpg
Crostata (it) data
{
  ...
  extext: <str(273)> La **crostata** è un dolce tipico italiano, a.
  ...
}
<wptools.page.WPToolsPage object at 0x7fb961236fd0>
```

- Dictionary giving access to (among others) the following page elements :
  - exthtml (content as html) → `data['exthtml']`
  - extext (content as text) → `data['extext']`
  - infobox → `data['infobox']`

`wptools`: accessing infoboxes

- Note the infobox itself is a dictionary:

```
>>> page = wptools.page('crostata') #default language is English
>>> page.get_parse()

en.wikipedia.org (parse) 4795498
en.wikipedia.org (imageinfo) File:Crostata limone e zenzero 3.jpg
Crostata (en) data
{
  ...
  infobox:  name, image, image_size, caption, country, r...
}
<wptools.page.WPToolsPage object at 0x7f05d90f7c50>

>>> page.data['infobox']['caption']

'Crostata with lemon ginger filling'
```

- See full documentation 🔗

Wikipedia and Open Linked Data

Introducing Wikidata

Wikidata (official presentation)

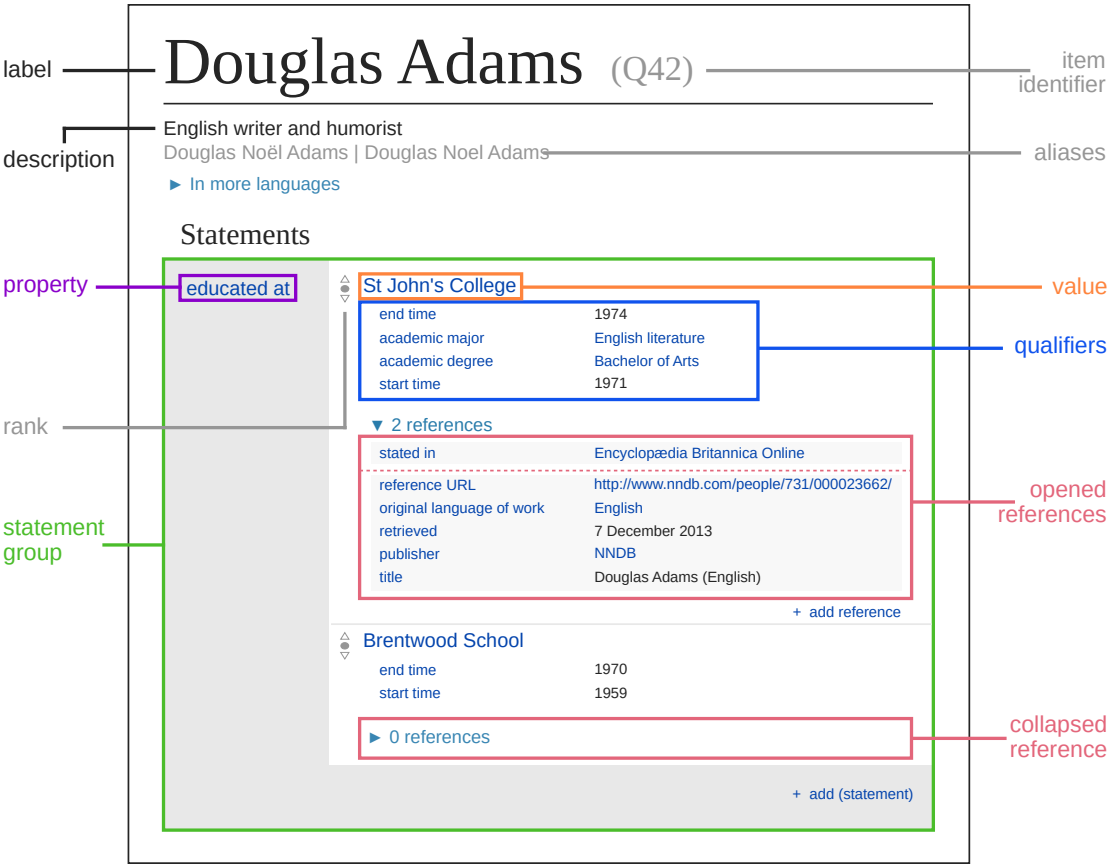
- Free and open **knowledge base that** can be read and edited by both humans and machines.
- **Central storage** for the **structured data** of its Wikimedia sister projects (e.g. Wikipedia, Wikivoyage, Wikibooks, Wikinews, etc.).
- **Support to many other services** beyond just Wikimedia projects!
- Content available under a **free license**, exported using **standard formats**, and **interlinkable** to other open data sets.

Wikidata (continued)

- Collection of **Entities** of two types:
  - **Items** (whose identifiers are prefixed with Q) ;
  - **Properties** (whose identifiers are prefixed with P)
- Items and properties have a **persistent Uniform Resource Identifier** (URI) obtained by appending its identifier to the **Wikidata concept namespace** (https://www.wikidata.org/entity/)
- Wikidata URIs can be used as **Uniform Resource Locators** (URL)
- Several output **formats** available, e.g. https://www.wikidata.org/wiki/Special:EntityData/Q42.json

Wikidata (continued)

Entities are *linked* using **statements** containing triples of the form (Item, Property, Value)



Wikidata (continued)

- Searching wikidata can be done with `wptools`:

```
>>> page = wptools.page(wikibase='Q3698703')
>>> page.get_wikidata()
```

```
www.wikidata.org (wikidata) Q3698703
www.wikidata.org (labels) Q2095|P373|P18|P279|P495|P646|Q38|Q133602
Note: Wikidata item Q3698703 missing 'instance of' (P31)
en.wikipedia.org (imageinfo) File:Crostata limone e zenzero 3.jpg
Crostata (en) data
{
  claims: <dict(5)> P373, P646, P279, P495, P18
  image: <list(1)> {'file': 'File:Crostata limone e zenzero 3.jpg'.
  label: Crostata
  labels: <dict(8)> Q2095, P373, P18, P279, P495, P646, Q38, Q13360
  wikibase: Q3698703
  wikidata: <dict(5)> Commons category (P373), Freebase ID (P646),.
  wikidata_pageid: 3524796
  wikidata_url: https://www.wikidata.org/wiki/Q3698703
}
```

Wikidata (continued)

- The **labels** field corresponds to items' descriptions:

```
>>> page.data['labels']
{'Q2095': 'food', 'P373': 'Commons category', 'P18': 'image', 'P279

>>> page.data['labels']['Q2095']
'food'
```

- The **claims** field corresponds to statements :

```
>>> page.data['claims']
{'P373': ['Crostata'], 'P646': ['/m/0cnp1y'], 'P279': ['Q13360264',
```

- See full list of wikidata fields 🔗
- See `wptools` examples on-line 🔗

Exercise sheet #4

Using APIs

Thank you!

Slideshow created using remark.