DataSci 400

lesson 6: data structures

Seth Mottaghinejad

today's agenda

- data structures
- "flattening" hierarchical data
- working with semi-structured data in Python
  - JSON
  - YAML
  - XML
  - HTML
- web-scraping with Python

overview of data structures

- different structures or **representation** for data
  - **tabular (structured)**: relational tables (`SQL`), matrices, `DataFrame` , ...
  - **semi-structured**: `JSON` , `XML` , `Mongo DB` , graph datatabases, ...
  - **unstructured**: raw text, images, sound, video, ...
- most ML algorithms only work with tabular data
- once data is made tabular, we still need to do a lot of pre-processing prior to ML

lab time

- here's an example of a single record in semi-structured data:

```
{
  "author" : ["Walter", "Habib"],
  "title" : "How to drink water",
  "language" : "Eng",
  "year_pulished" : ["2008", "2012"]
}
```

- represent it using a **tabular structure** (there is more than one way)
- propose extra columns that can be **extracted** from the above data

break time

lab time

- in the next page, you will see two tabular representation of the semi-structured data below, in **wide format** and **long format**

```
{
  "author" : ["Walter", "Habib"],
  "title" : "How to drink water",
  "language" : "Eng",
  "year_pulished" : ["2008", "2012"]
}
```

- which format do you prefer? why? (think about what happens as more and more books are added to each table)

book title	lang	auth_1	auth_2	pub_1	pub_2
How to drink water	Eng	Walter	Habib	2008	2012
⋮	⋮	⋮	⋮	⋮	⋮

book title	lang	author	published
How to drink water	Eng	Walter	2008
How to drink water	Eng	Walter	2012
How to drink water	Eng	Habib	2008
How to drink water	Eng	Habib	2012
⋮	⋮	⋮	⋮

working with semi-structured data

- there are many packages in Python for dealing with different data formats: `yaml`, `json`, `lxml`, `bs4`, etc.
- they are used **read**, **query**, **modify**, and **write** to each format
- querying can be done using what we already know about navigating Python objects like **dictionaries and lists**
- these packages also **offer helper** functions to make it easy to run query against each data format
- alternatively, each format has one or serveral **query languages**, such as **XPath**, **JSONPath**, **YAQL** and more

web-scraping

- knowledge of semi-structured data can be very handy if we want to learn how to **scrape the web**
- information on the web looks very disorganized and it can be, but getting behind the scenes we can often find **structure**
- once you understand the structure, you know how to query it
- but in practice, this is a difficult topic and many books and articles are written on the subject
- you can really improve your Python skills by trying this out!

references

- YAML
- JSON
- XML

the end