DATA ACQUISITION

project with the latest technology

Lee Hae Joon

Motivation

- Data acquisition has been understood as the process of **gathering, filtering, and cleaning data** before the data is put in a data warehouse or any other storage solution. The acquisition of big data is most commonly governed by four of the Vs: **volume, velocity, variety, and value**

- Data Acquisition is the step where you get the data in one or more of the below ways
  - either free found or by buying data,
  - either using a specialist web scraper technology or by simple copy pasting,
  - either from internal sources (**sales reports**) or external (**trade journals, third party web sites**)

Interesting Article

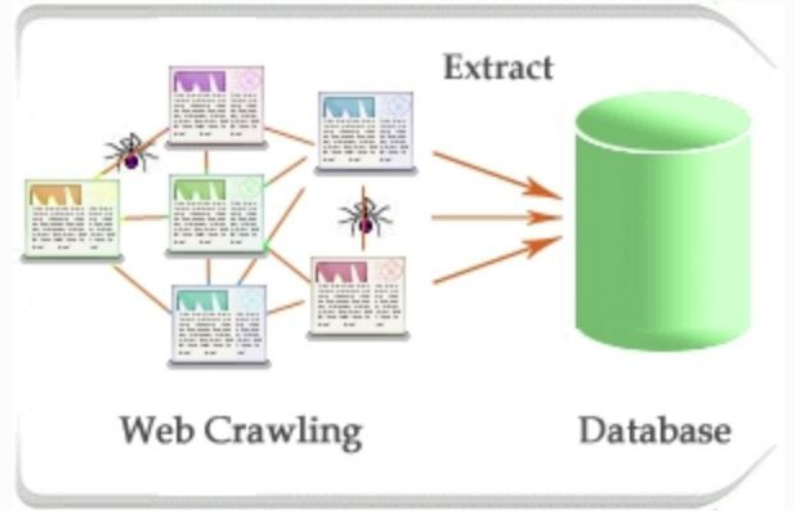
- https://link.springer.com/chapter/10.1007/978-3-319-21569-3_4
- https://www.promptcloud.com/blog/Why-web-data-acquisition-is-one-of-the-biggest-pain-points-in-the-data-industry
- https://www.promptcloud.com/data-scraping-vs-data-crawling/

Contents

- **Crawling**
- Scraping

Crawling

Crawling usually refers to dealing with large datasets where you develop your own crawlers (or bots) which crawl to the deepest of the web pages.



Crawling

Urlopen function can be used to retrieve data from the Website

```
import urllib.request

url = "http://uta.pw/shodou/img/28/214.png"
savename = "test.png"

mem = urllib.request.urlopen(url).read()

with open(savename, mode="wb") as f:
    f.write(mem)
    print("saved")
```

Crawling

Urlopen function can be used to retrieve data from the Website

```
import urllib.request

url = "http://api.aoikujira.com/ip/ini"
res = urllib.request.urlopen(url)
data = res.read()

text = data.decode("utf-8") # binary to string by decoding
print(text)
```

Crawling

Dynamically crawling data from 기상청 RSS

http://www.weather.go.kr/weather/lifenindustry/sevice_rss.jsp

Crawling

Dynamically crawling data from 기상청 RSS

```python
#!/usr/bin/env python3
import sys
import urllib.request as req
import urllib.parse as parse

if len(sys.argv) <= 1:
    print("USAGE: download-forecast-argv <Region Number>")
    sys.exit()
regionNumber = sys.argv[1]

API = "http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp"
values = {
    'stnId': regionNumber
}
params = parse.urlencode(values)
url = API + "?" + params
print("url=", url)

data = req.urlopen(url).read()
text = data.decode("utf-8")
print(text)
```

Crawling

URL 형식에 맞게 Parsing https://en.wikipedia.org/wiki/URL
- http://ce.khu.ac.kr?key1=v1&key2=v2

> Every HTTP URL conforms to the syntax of a generic URI. A generic URI is of the form:

```
scheme:[//[user[:password]@]host[:port]][/path][?query][#fragment]
```

shebang? https://en.wikipedia.org/wiki/Shebang_(Unix)
- #!/usr/bin/env python3
- 실행권한이 있을때

Scrapping

Scraping data does not necessarily involve the web. Data scraping could refer to **extracting information from a local machine, a database, or even if it is from the internet,** a mere “Save as” link on the page is also a subset of the data scraping universe.

Scrapping, BeautilfulSoup

Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of **navigating, searching, and modifying the parse tree**. It commonly saves programmers hours or days of work.

>> pip3 install beautifulsoup4

https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Scrapping, BeautilfulSoup

Parsing DOM (Document Object Model)

```
from bs4 import BeautifulSoup

html = """
<html><body>
<div id="project">
  <h1 id='title'>BIG DATA PROGRAMMING</h1>
  <p id='body'>DATA ANLYSIS AND SCIENCE</p>
  <p>DATA ACQUISTION PART1</p>
  </ul>
  <ul class="items">
    <li>CRAWLING</li>
    <li>SCRAPPING</li>
    <li>HYBRID WAY</li>
</div>
</body></html>
"""
```

< html data from github >

```
soup = BeautifulSoup(html, 'html.parser')

h1 = soup.html.body.h1
p1 = soup.html.body.p
p2 = p1.next_sibling.next_sibling

print("h1 = " + h1.string)
print("p  = " + p1.string)
print("p  = " + p2.string)

title = soup.find(id="title")
body  = soup.find(id="body")

print("#title=" + title.string)
print("#body="  + body.string)

h1 = soup.select_one("div#project > h1").string
print("h1 =", h1)

li_list = soup.select("div#project > ul.items > li")
for li in li_list:
  print("li =", li.string)
```

Scrapping, BeautilfulSoup

Practice

url = "http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp"

1. access this url (e.g., res = req.urlopen(url)
2. use BeautifulSoup to extract DOM (e.g., title, wf)

Scrapping, BeautilfulSoup

Explore "http://info.finance.naver.com/marketindex/"

Scrapping, BeautilfulSoup

Practice

url = "http://info.finance.naver.com/marketindex/"

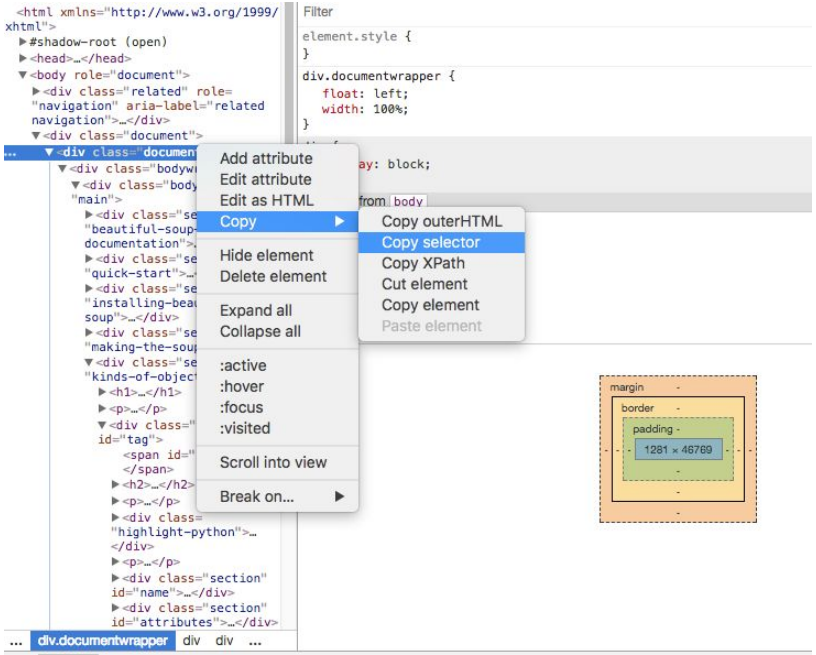
1. access this url (e.g., res = req.urlopen(url)
2. use BeautifulSoup ‘select_one()’ to extract **PRICE**

Find only the first tag that matches a selector:

```
soup.select_one(".sister")
# <a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>
```

Scrapping, BeautilfulSoup

Tips - CSS Selector (Chrome -> Right Button -> Copy -> Copy Selector)



Scrapping, BeautilfulSoup

```
html = """
<ul>
  <li><a href="hoge.html">hoge</li>
  <li><a href="https://example.com/fuga">fuga*</li>
  <li><a href="https://example.com/foo">foo*</li>
  <li><a href="http://example.com/aaa">aaa</li>
</ul>
"""
```

Practice

1. access this **html**

2. use BeautifulSoup to parse by **regular expression**
   - href=re.compile(r"^https://")
   - https://www.crummy.com/software/BeautifulSoup/bs4/doc/

3. print all attributes
   - for e in li: print(e.attrs['href'])