Web scraping and social media scraping – introduction

Jacek Lewkowicz, Dorota Celińska-Kopczyńska

University of Warsaw

March 18, 2019

Motivation

- Tons of (potentially useful) information on the Web
- Instead of manual gathering the information needed, we employ computer programs to do it automatically
- We have to understand the structure of the webpage and find general scheme for data collection
- Web scraping may be useful and efficient here

Web scraping

- The process of taking unstructured information from the Web...
- ... and processing it into structured data for later analysis
- Commonly used not only by researchers: Web crawlers/web spiders/search bots used by search engines
- To *scrape* the web (=drag; *scra***p***ing*), not to *scrap* (=discard; *scra***pp***ing*)!

Bots, crawlers, spiders, scrapers...

- A **bot** is a software application that runs automated, usually repetitive tasks (scripts) over the Internet
- A **crawler** or **spider** is a bot which systematically browses the Internet, typically for the purpose of Web indexing. In other words, crawler surfs the web, following links – it does not have to extract data
- A **scraper** takes downloaded pages and attempts to extract data from them.
- However, there are no precise defitnions

Example areas of usage

- Download results of general elections
- Download demo MP3
- Download legal decisions
- Download comments and users' activity from social media

Alternatives to *traditional* scraping

- ScraperWiki
- Google Refine
- Outwit
- SocSciBot
- (arguably) APIs, already scraped collections of data available online (e.g., GHTorrent) or API mirrors

Is it legal?

- Technically... usually it is. However, **there are situations in which you are not allowed to gather the data from the given website**
- Look for disclaimers and Terms of (Fair) Use
- Scraping may mean copyright infringement
- Data privacy, sensitive information
- web crawling – difficult matter

Controversial cases

- eBay vs Bidder's Edge
  - One company crawls information from another
- Facebook vs Meltwater
  - Prior written permissions needed
- US vs Aaron Swartz
  - AS arrested in 2011 for having illegally downloaded millions of articles from the article archive JSTOR. The case was dismissed after Swartz' suicide in January 2013
- GHTorrent #issue32 incident
  - `http://speakerdeck.com/player/1c64fd1e7dfe4032aff246b2dd1195bf#`

Web scraping for social good or evil?

- People tend to be extremely paranoic about their data privacy, while being at the same time extremely careless about their data privacy
- Same people who cover their cameras in laptops, do not potentially care about the photos they upload to Facebook/other social media or the apps they are using
- *Celebgate* `http://en.wikipedia.org/wiki/ICloud_leaks_of_celebrity_photos`
- AI gone too far... `http://www.telegraph.co.uk/technology/2017/09/08/ai-can-tell-people-gay-straight-one-photo-face/`

Counteracting scraping – `robots.txt`

- Many web crawlers are hunting for content
- Web robots keep some indices actual
- `robots.txt` file and other methods are used for keep the server traffic in check

```
User-agent:  *
Disallow:  /private/
```

`robots.txt` – limitations

- Considered outdated standard due to its limitations
- The subsite can be still indexed if another site redirects to it
- Even if the subsite is not listed in `robots.txt`, it does not mean the owner allows to scrape it.
- Imagine a situation between the host and the guest: *Ok, feel at home, but do not take the money I put in the jar on the third shelf in the kitchen cupboard*. The guest should obey but will they? The same is with the `robots.txt`, which discloses the information that should stay hidden, e.g., the links to admin's login sites
- Read: `http://support.google.com/webmasters/answer/6062608`

Counteracting scraping – <meta> tags

- A modern approach to blocking bots from accessing the part of the website is using of `<meta> tags`
- `<meta name="robots" content="noindex">`
- Read: `https://support.google.com/webmasters/answer/93710`

What makes a crawler polite?

- A polite crawler respects `robots.txt`
- A polite crawler never degrades a website's performance
- A polite crawler identifies its creator with contact information
- A polite crawler does not drive system administrators crazy

Responsible crawling #1 – Terms of Use

- Make sure your crawler follows the rules contained in Terms of (Fair) Use
- **Do check `robots.txt` file**
- (*we know you have no time*) you may write a program, which parses `robots.txt` and enables to identify particular user-agents or corresponding rules of access!

Responsible crawling #2 – Identify yourself

- Include your company name and an email address or website in the request's User-Agent header
- For example, Google's crawler user agent is "Googlebot".

Responsible crawling #3 – Crawl delay

- Make sure your crawler does not hit a website too hard
- Respect the delay that crawlers should wait between requests by following the `robots.txt` Crawl-Delay directive
- If you do not... they will make you do

```
2012-12-21 05:04:36+0800 [working] DEBUG: Retrying
http://www.example.com/profile.php?id=1580> (failed 1
times):  503 Service Unavailable
```

Responsible crawling #4 – Take care of your data

- You should pay attention to what happens with the data you gathered
- Consider which part of the analyses and/or data sets can be displayed publicly
- Respect the sensitivity of the information

Good practices & ethics in a nutshell

- Respect the hosting site's wishes
  - `robots.txt` as a list of instruction to bots/scrapers
  - Terms of (Fair) Use – which elements of the website you should not scrap
- Respect the hosting site's bandwidth
  - Scraping may be costly or result in the site going down
- Respect the law
  - Terms & agreements
- Take care of the data
  - Consider which information can be displayed publicly or not
- If possible use APIs, APIs mirrors or collections of data already scraped for you

Requests library

- Python core libraries may be enough, but there are more appropriate packages
- `Requests` library is suitable for complicated HTTP requests, cookies, headers and other issues

Beautiful... what?

- Beautiful Soup is Python library
- Lewis Carroll's poem from *Alice's Adventures in Wonderland*
- The tool is for making sense of nonsensical

Beautiful Soup, so rich and green,
Waiting in a hot tureen!
Who for such dainties would not stoop?
Soup of the evening, beautiful Soup!

Installing Beautiful Soup

- `http://www.crummy.com/software/BeautifulSoup/bs4/doc/`
- `http://pypi.python.org/pypi/setuptools`
- Think about setting Python virtual environments
- Unixlike (Mac and GNU/Linux)
  - `sudo easy_install pip`
  - `pip install beautifulsoup4`
- Other environments
  - `python setup.py install`
  - `pip install beautifulsoup4`

Problems after installation

- Authors of the library suggest that usually known problems are related to Python 2 vs Python 3 issues...

Typical way of working

- You work with file containing python code and import the necessary libraries
- It requires some skills in programming

What is Scrapy?

- Scrapy is a **framework** which means a lot of issues related to scraping have already been solved for you...
- ... and you may use "automagical" tools to deal with them (with all the blessings and curses it provides)
- Requires minimal knowledge and skills in Python and programming
- Handles a lot of complexity of finding & evaluating links on a website, scraping whole domains or lists of domains

Scrapy – installation

- `http://scrapy.org/`
- `http://doc.scrapy.org/en/latest/intro/install.html`
- `http://pypi.python.org/pypi/setuptools`
- Think about setting Python virtual environments (recommended)
- Unixlike (Mac and GNU/Linux)
  - `sudo easy_install pip`
  - `pip install Scrapy`
- Other environments
  - Read the manual, recommended use of `anaconda`

Scrapy – dependencies

- If you encounter problems during the installation, probably you should check for the dependencies, e.g.,
  - `lxml` – efficient XML and HTML parser
  - `w3lib` – a multi-purpose helper for dealing with URLs and web page encodings
  - `cryptography` and `pyOpenSSL` – to deal with various network-level security needs

Scrapy – typical way of working

- You work with *projects* in a new directory
- Each Scrapy object stands for a single page on the website
- Each scrapper has a few core functions and structures
- Crawlers are run from the command line
- Troubleshooting usually within configuration files (`settings.py`)

Scrapy – responsible scraping

- `ROBOTSTXT_OBEY = True`
- `USER_AGENT = 'MyCompany-MyCrawler (bot@mycompany.com)'`
- `DOWNLOAD_DELAY`
- `CONCURRENT_REQUESTS_PER_DOMAIN`

```
2016-08-19 16:12:56 [scrapy] DEBUG: Forbidden by
robots.txt:  <GET http://website.com/login>
```

Homework

- **Install the libraries and frameworks on your hardware!**
- Read the **Introduction** and **Creating a project** sections in **Scrapy tutorial**:
  `http://doc.scrapy.org/en/latest/intro/tutorial.html`