COMSM0085

Overview of Software Tools

Software Tools: Part 2

(COMS10012 / COMSM0085)

Welcome back – two weeks to go:

Week 9: Web scraping

Week 10: Encryption

Why are we learning this?

Previously: writing and serving websites, presenting data in a website.

Now: collecting webpages, extracting data from them.

The web is a massive data resource – why would we we only access it manually?

Swarms of robots

Approximately half of web traffic is not human.

Robots visit websites for a variety of reasons:

- Indexing websites for search or archive purposes.
- Carrying out specific tasks for their owners (e.g., newsreader-like tools).
- Probing for security vulnerabilities.
- Pretending to be humans for nefarious purposes (e.g., in order to advertise something on social media).
- Generating false 'traffic' so website owners can scam ad networks.

Crawlers

Depending on the usage, also a 'spider' – because it traverses the web:

- Spiders are indexing content, usually for search purposes
- Other 'crawlers' may have other uses for the content.

A crawler is a HTTP client, like a browser, but automated. Two approaches:

- write your own purpose-specific software
- use an existing tool

Remember `wget <url>` ?

wget is a crawler

'do one thing well'

So far you've only used the bare minimum of `wget` 's capacity

Downloading webpages for offline consumption:

```
wget https://news.ycombinator.com/news
```

vs

```
wget -p https://news.ycombinator.com/news
```

Resources required to display the page correctly.

robots.txt

Standard for websites to use to tell HTTP or FTP crawlers which parts of a site can be accessed.

List of rules for which parts of the site a crawler can access:

```
User-Agent: foobot
Allow: /example/page/
Disallow: /example/page/disallowed.gif

User-Agent: bazbot
Disallow: /example/page.html
```

Crawlers are meant to check for their own user-agent string in the file (always placed in the webroot) and follow the rules.

robots.txt issues

- Doesn't actually enforce access restrictions.
- Need real security mechanisms to do that (e.g., authentication).
- By listing site components, makes content findable.

More of an 'honour system' to enable good bots to respect the wishes of website owners.

`wget` is a good bot.

You should also write good bots!

More wget

```
elinks danluu.com
```

In the labs you'll practice *recursive* downloading and true web mirroring using `wget`.

Key use of this: make a personal offline backup of a website you like.

What else can we do with copies of websites?

Other limits on crawling

You have a generally unrestricted right to *access* public web content.

This doesn't mean you can do anything you want with it.

- copyright limits on republishing content
- ethical limits, especially for authenticated content

The means of accessing content can also be important – aggressive downloading of large sites can pose a burden on servers. Servers sometimes respond by blacklisting clients.

Generally 'polite' to introduce small delays between requests, even if not asked for.

API endpoints designed for automated access may impose different rate limits.

What else *can* we do with it?

Webpages often present structured information which we would like to work with.

However, page structures can be complex, and are very site-specific.

Need a system for accessing page content programmatically.

Javascript has methods for this *within* the browser, but often we want our code to run on its own.

BeautifulSoup

Python library for extracting data from HTML files.

- Not a HTML parser itself, but can use many parsers.
- Provides a flexible interface for navigating, searching and altering HTML programmatically.

Current version is BeautifulSoup 4 (`bs4`).

HN demo

```
from bs4 import BeautifulSoup
```

```
filename = "news.html"
handle = open(filename, 'r')
text = handle.read()
soup = BeautifulSoup(text)
```

Not a Python course

We don't make Python itself a focus for this unit.

Dicts

```
d = {}
d['this'] = 'that'
d['other'] = 'value'
```

```
d = {'a': 1, 'b': 2}
```

Lists

```
l = ['a']
l.append('b')
l.append('c')
```

This week

- Using `wget` and understanding how to control its behaviour.
- Writing simple Python programs using BS4.
- Support for Javascript lab.