OCTOBER 8, 2017 BY AS9736

Web Scraping YouTube Videos in Python

Web crawling and web scraping are two sides of the same coin. Web scraping is simply extracting information from the internet in an automated fashion. Web crawling is about indexing information on webpages and – normally – using it to access other webpages where the thing you actually want to scrape is located.

YouTube is a brilliant source of online information, not only does it host a ton of video and audio data, but it also lets people comment on and "like" those data. It effectively allows you to perform opinion mining about dataset while also providing the datasets themselves. Here I'm going to look at how to download videos.

To extract those data from YouTube requires a bit of web crawling and web scraping – and it's surprisingly straight forward to do in Python. Especially since there are lots of libraries out there to help you. Two of the most popular Python libraries for web scraping are BeautifulSoup and ScraPy. Here I'm going to pick randomly and use BeautifulSoup.

Beautiful Soup

BeautifulSoup is pip installable:

```
1 pip install beautifulsoup
```

However this will install v3 of beautifulsoup and it's likely that you'll probably want v4. To get it use:

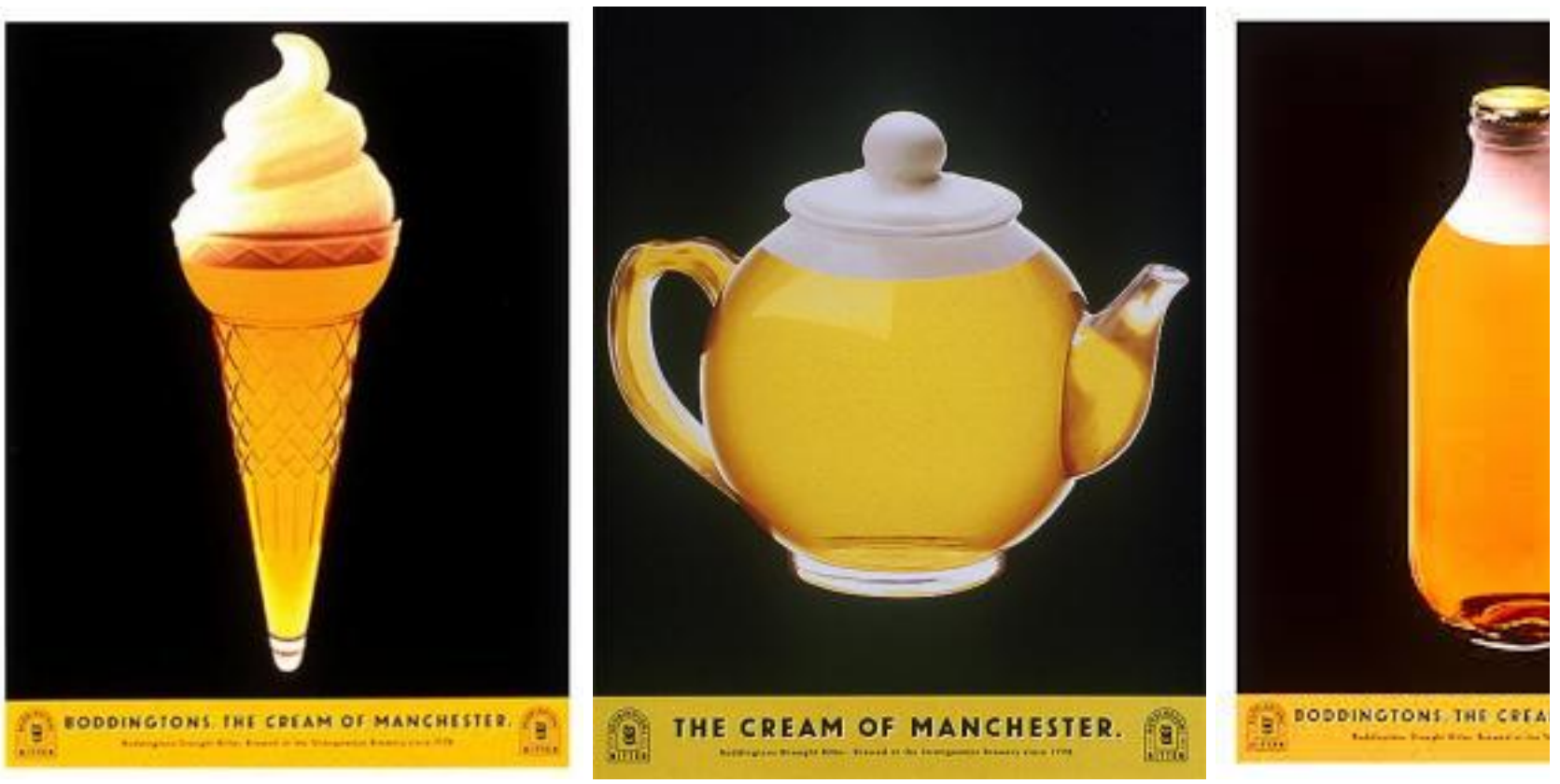
```
1 pip install -U bs4
```

I'm also going to use the `http` requests library:

```
pip install requests
```

First off, we need to decide what we want to download from YouTube. YouTube has a reputation as being difficult to scrape because it's a maze of information. So for this simple example I'm going in with a clear idea of what I want and if I don't get 100% exactly that, I won't be devastated.

I'm using adverts as an example. Let's say that I want a copy of all the adverts for Boddington's beer (remember those?).



We'll need to import our libraries:

```
from bs4 import BeautifulSoup as bs
import requests
```

I'm going to query YouTube for a particular search string:

```
base = "https://www.youtube.com/results?search_query="
qstring = "boddingtons+advert"

r = requests.get(base+qstring)
```

We can then extract the `html` of the search results page using BeautifulSoup:

```
page = r.text
```

```
soup=bs(page,'html.parser')
```

Once you've used BeautifulSoup to parse the html, you can extract all kinds of things. For this example I'm just looking at the search results.

Each of the search result items is a tile on the page, which links to one of the search results. In `html` terms these are <a> tags because they define hyper-links. Within each <a> tag there will be a parameter called `href`, which is the link to the individual page of each search result.

We need to extract the links to the individual videos, so we start by making a list of all the <a> tags for each tile. There are a bunch of different <a> tags on the whole page, we just want the ones that are classified as YouTube User Interface XML Tile Links: `yt-uix-tile-link`.

```
vids = soup.findAll('a',attrs={'class':'yt-uix-tile-link'})
```

[In fact you can even leave out the `'a'` in the search and just use `soup.findAll(attrs={'class':'yt-uix-tile-link'})` if you want.]

We can then extract the web links for each tile using the `href` property. I'm going to append them into a list.

```
videolist=[]
for v in vids:
    tmp = 'https://www.youtube.com' + v['href']
    videolist.append(tmp)
```

PyTube

Up until this point we've basically been crawling the YouTube webpages to index the information. Now we've done that we can scrape the bits we want.

The other library I'll be using allows us to extract/download video material from YouTube. It's called PyTube and you can also get it from pip:

```
pip install pytube
```

We can import the YouTube tools:

```
from pytube import YouTube
```

Then loop over all the web-links in our list of hyper-refs and download the video for each one:

```
count=0
for item in videolist:

    # increment counter:
    count+=1

    # initiate the class:
    yt = YouTube(item)

    # have a look at the different formats available:
    #formats = yt.get_videos()

    # grab the video:
    video = yt.get('mp4', '360p')

    # set the output file name:
    yt.set_filename('Video_'+str(count))

    # download the video:
    video.download('./')
```

...and that's it.

However, to end on a cautionary note: the YouTube ToS do state that,

> ***5.1 H.*** *you agree not to use or launch any automated system (including, without limitation, any robot, spider or offline reader) that accesses the Service in a manner that sends more request messages to the YouTube servers in a given period of time than a human can reasonably produce in the same period by using a publicly available, standard (i.e. not modified) web browser;*

Then for the blog this.

DATA ACQUISITION, WEB CRAWLING, WEB SCRAPING

DATAMINING, MACHINE LEARNING, PYTHON, SOCIALDATA

One Reply to “Web Scraping YouTube Videos in Python”

Pingback: WebCrawling: YouTube Pagination in Python – ALL YOUR BASE ARE BELONG TO US