GETTING DATA FROM THE WEB

27-11-2012
Philo van Kemenade
CAST Sandbox
Goldsmiths College

HELLO

Philo van Kemenade
BSc Artificial Intelligence, University of Amsterdam
MSc Cognitive Computing, Goldsmiths College

I use Python for:
tracking video propagation in over 100gb of tweets
building statistical models for natural language processing
analysing experimental data from a database
talking to robots :)

You can find these slides at:
tinyurl.com/datafromtheweb

THIS SESSION

- Recap Python
- Web Information Extraction
- Web Scraping
- Exercise 1
- Using the Twitter API
- Exercise 2
- Mini Project
- Useful References

RECAP

Codecademy?

Interactive vs Scripting

Variables and Data Types

Collections of Data

Functions & Methods

Flow of Control

Libraries

INTRO: Setting up

Install '**setuptools**' from http://pypi.python.org/pypi/setuptools

This let's you install Python packages super easy!

In terminal / cmd window:

```
phillchill@16:20:22: easy_install beautifulsoup4
Searching for beautifulsoup4
Reading http://pypi.python.org/simple/beautifulsoup4/
Reading http://www.crummy.com/software/BeautifulSoup/bs4/
Reading http://www.crummy.com/software/BeautifulSoup/bs4/download/
Best match: beautifulsoup4 4.1.3
Downloading http://www.crummy.com/software/BeautifulSoup/bs4/download/
beautifulsoup4-4.1.3.tar.gz
Processing beautifulsoup4-4.1.3.tar.gz
Running beautifulsoup4-4.1.3/setup.py -q bdist_egg --dist-dir /var/folders/Sj/
SjSaOugoFY4ESIgBR4CMBE+++TI/-Tmp-/easy_install-lXGWke/beautifulsoup4-4.1.3/egg-dist-tmp-
aTtESG
zip_safe flag not set; analyzing archive contents...
Adding beautifulsoup4 4.1.3 to easy-install.pth file

Installed /Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/site-packages/
beautifulsoup4-4.1.3-py2.7.egg
Processing dependencies for beautifulsoup4
Finished processing dependencies for beautifulsoup4
```

INFORMATION EXTRACTION

Data on the web:

Semi-structured on a web page

- different structure for every page
- available in your browser
- extractable by *scraping*

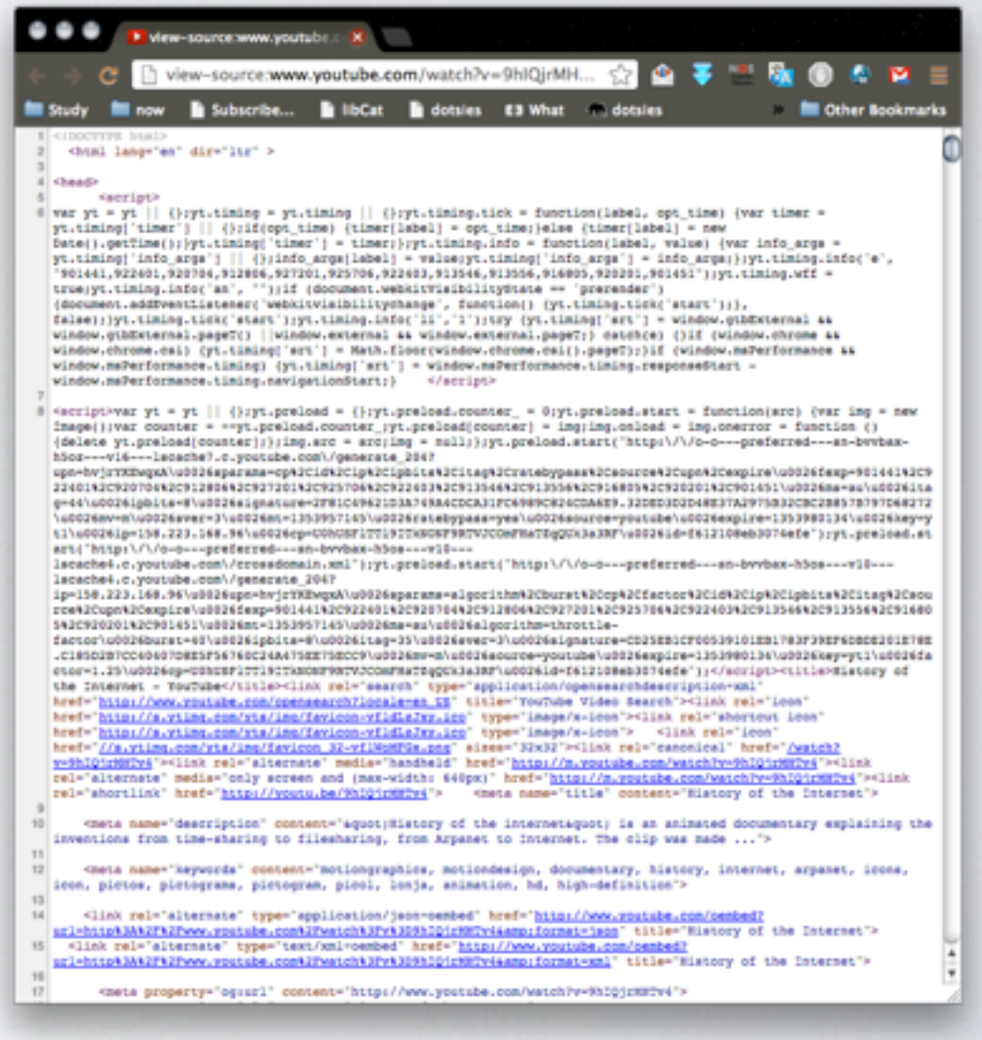
Structured in databases

- structured and indexed
- hidden on the server-side of a web platform
- may be accessible by *Application Programming Interface* (API)

WEB SCRAPING

Extracting data from a web page's source

For example: http://www.youtube.com/watch?v=9hlQjrMHTv4



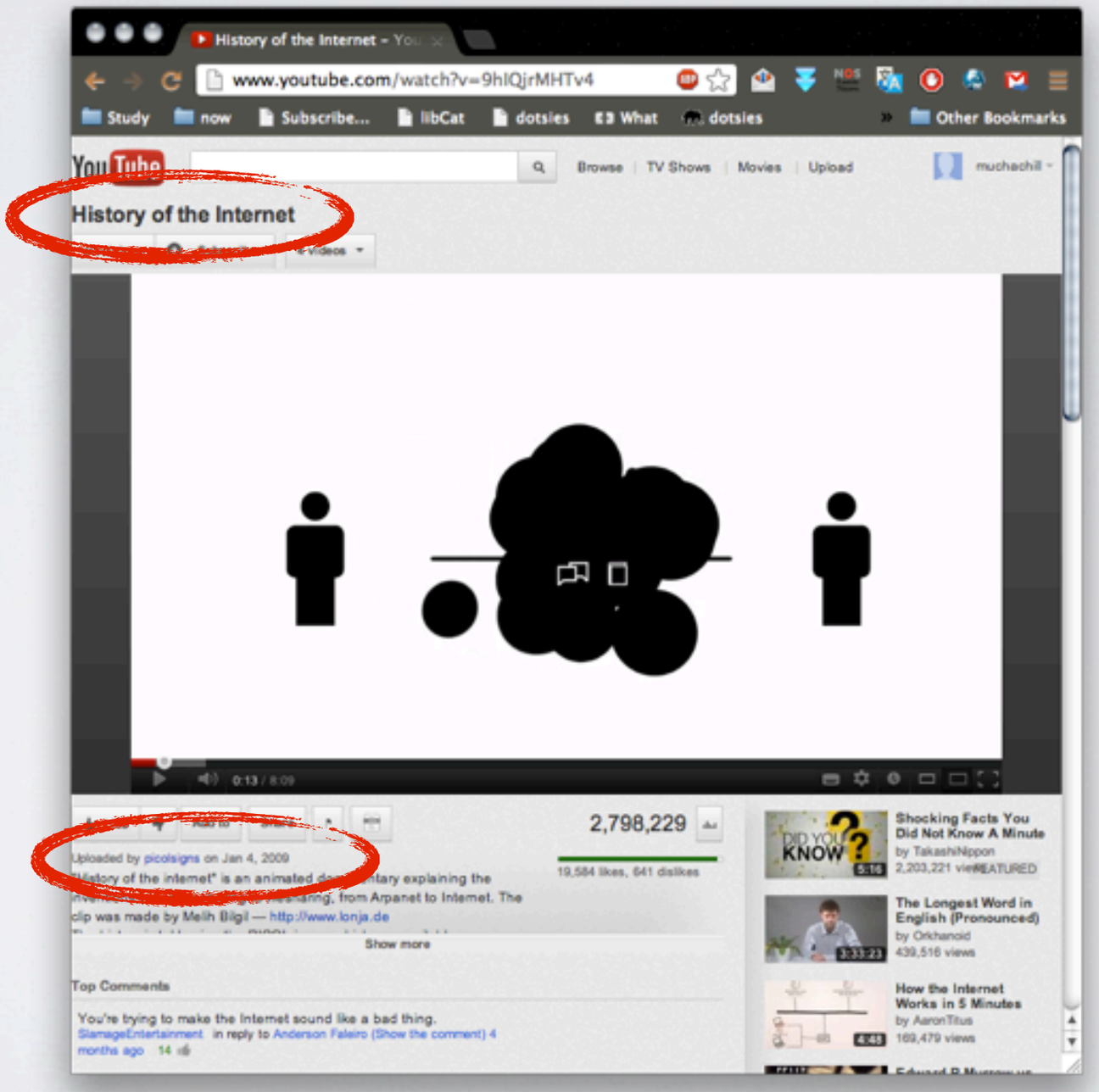
WEB SCRAPING

Pseudo code:

1. Determine what data we are looking for
2. Obtain semi-structured document
3. Find out where data is in semi-structured format
4. Parse semi-structured document into something more structured
5. Extract what we're looking for
6. Do stuff with our extracted data

WEB SCRAPING

1. Determine what data you are looking for



WEB SCRAPING

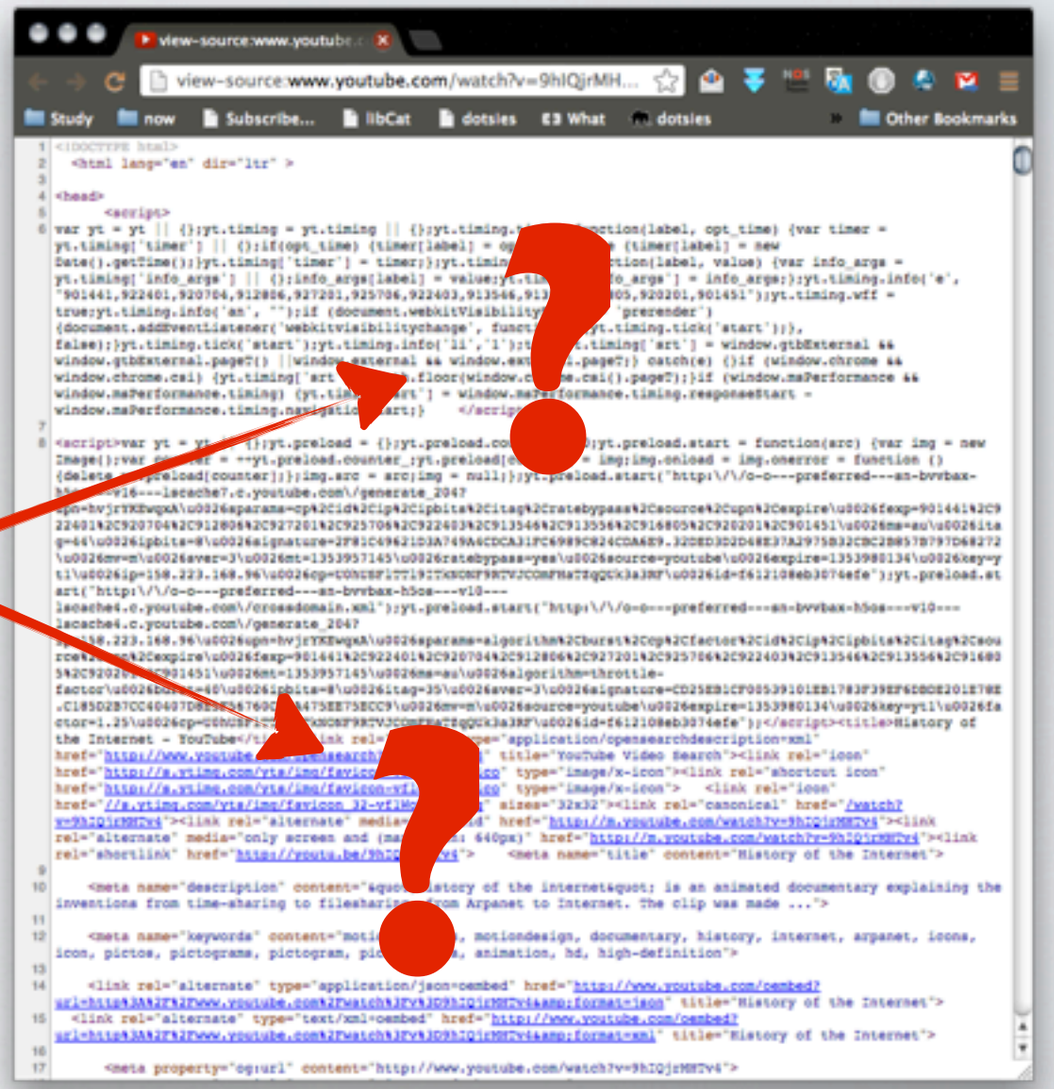
2. Obtain semi-structured document

```
# import library
from urllib import urlopen

# open webpage
webpage = urlopen('http://www.youtube.com/watch?v=9hIQjrMHTv4').read()
```

WEB SCRAPING

3. Find out where data is in semi-structured format



WEB SCRAPING

4. Parse semi-structured document into something more structured

***Beautiful Soup*:**

"Beautiful Soup provides a few simple methods [...] for navigating, searching, and modifying a parse tree: a toolkit for **dissecting a document** and **extracting what you need**.
It **doesn't take much code** to write an application"

(easy_install BeautifulSoup4)

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

WEB SCRAPING

4. Parse semi-structured document into something more structured

```
# import library
from bs4 import BeautifulSoup

# turn html into beautiful soup
ytSoup = BeautifulSoup(webpage)
```

Now we can do easy stuff like:

```
>>> ytSoup.title
<title>History of the Internet - YouTube</title>
```

WEB SCRAPING

5. Extract what we're looking for

For example the uploader's user name:

```
# extract info from soup
uploader = ytSoup.find('a', class_='author').string
```

WEB SCRAPING

6. Do stuff with our extracted data

For example print to screen:

```
# print info to screen
print 'uploader: ', uploader
```

Or save in .csv file
http://docs.python.org/2/library/csv.html
http://www.doughellmann.com/PyMOTW/csv/

WEB SCRAPING

All together now

```
'''A simple script that extracts and prints info from a Youtube video page'''

# import libraries
from urllib import urlopen
from bs4 import BeautifulSoup

# open webpage
webpage = urlopen('http://www.youtube.com/watch?v=9hIQjrMHTv4').read()

# turn html into beautiful soup
ytSoup = BeautifulSoup(webpage)

# extract info from soup
title = ytSoup.find('span', id='eow-title').string
uploader = ytSoup.find('a', class_='author').string
date = ytSoup.find('span', id='eow-date').string

# print info to screen
print 'title: ', title
print 'uploader: ', uploader
print 'date: ', date
```

WEB SCRAPING

looping over RSS

```
[...truncated...]
# open rss page
rssPage = urlopen("http://gdata.youtube.com/feeds/base/users/
TEDtalksDirector/uploads?alt=rss&v=2&orderby=published").read()
# turn page into soup
rssSoup = BeautifulSoup(rssPage)
# extract links
ytLinks = rssSoup.find_all('link')

# loop over videos
for link in ytLinks:
    if ("watch" in link.string):
        print "now scraping: "
        print link.string
        # call scrape function on yt video url
        scrape_yt(link.string)
        print "\n"
```

EXERCISE 1

Choose another Youtube video page to scrape from
Adapt the script on the previous slide to also scrape the video category

Bonus: Also scrape the number of views

(Cheating is encouraged)

QUESTIONS

USING AN API

Accessing online data via an ***Application Programming Interface***

For example:

the Twitter API

entities

tweets

users

places

You can easy_install several packages to connect to the twitter API

For example: `easy_install twitter`

(https://github.com/sixohsix/twitter)

USING AN API

Twitter actually has three APIs:

- Search API; searching tweets
- REST API; accessing core twitter concepts
- Streaming API; for real-time & data-intensive use

https://dev.twitter.com/docs

USING AN API

Making HTTP requests to the REST API

For example **show user info**:

http://api.twitter.com/1/users/show.json?screen_name=castlondon

API resource

parameter(s)

API version

```
import twitter
import json
# construct twitter API object
twitterAPI = twitter.Twitter(domain="api.twitter.com", api_version='1')
# Get user info
screenName = "castlondon"
response = twitterAPI.users.show(screen_name=screenName)
# print json formatted response to screen
print json.dumps(response, indent=2)
```

https://dev.twitter.com/docs/api/1/get/users/show

USING AN API

get trends:

http://api.twitter.com/1/trends/1.json

API resource

parameter(s)

API version

```
import twitter
import json
# construct twitter API object
twitterAPI = twitter.Twitter(domain="api.twitter.com", api_version='1')
# Get trends
response = twitterAPI.trends(id='1')
# print json formatted response to screen
print json.dumps(response, indent=2)
```

https://dev.twitter.com/docs/api/1/get/trends/%3Awoeid

USING AN API

search tweets:

http://search.twitter.com/search.json?q=cast&rpp=5

parameter(s)

API

API resource

```
import twitter
import json
# construct twitter API object
searchApi = twitter.Twitter(domain="search.twitter.com")
# Get trends
query = "#cast"
response = searchApi.search(q=query, rpp=100)

# print json formatted response to screen
```

https://dev.twitter.com/docs/api/1/get/search

USING AN API

turning *JSON* into Python *Dictionary*

```
...
>>> response = searchApi.search(q='#cast', rpp=100)
>>> type(response)
<class 'twitter.api.WrappedTwitterResponse'>
>>> responseString = json.dumps(response)
>>> responseDict = json.loads(responseString)
>>> type(responseDict)
<type 'dict'>
>>> responseDict.keys()
[u'next_page', u'completed_in', u'max_id_str', u'since_id_str',
u'refresh_url', u'results', u'since_id', u'results_per_page', u'query',
u'max_id', u'page']
```

EXERCISE 2

Have a look at the twitter API documentation
https://dev.twitter.com/docs/api/1

Find out how to get the user ids of all the followees (“friends”) of someone you know

Bonus: format the response into a dictionary and count the number of followees

(Cheating is encouraged)

QUESTIONS

MINI PROJECT

Create a web scraping program that extracts article information from your favorite online news source (blog, newspaper site, reddit, ...)

Extract at least:

- article text
- author
- date

Bonus: automate the process for all the articles in a RSS feed

USEFUL RESOURCES

Matthew A. Russel - Mining the Social Web; Practical book covering social data mining with Python

www.google.com; "python" + your problem / question

www.python.org/doc/; official python documentation, useful to find which functions are available

http://docs.python.org/tutorial/; official tutorial if you want to explore more detail

www.stackoverflow.com; huge gamified forum with discussions on all sorts of programming questions, answers are ranked by community

http://tinyurl.com/datafromtheweb; these slides :)