Big Data and Automated Content Analysis

Week 3 – Monday

»Data harvesting and storage«

Damian Trilling

d.c.trilling@uva.nl
@damian0604
www.damiantrilling.net

Afdeling Communicatiewetenschap
Universiteit van Amsterdam

19 February 2018

Today

1. Last week's exercise
   - Step by step
   - Concluding remarks
2. Data harvesting and storage
   - APIs
   - RSS feeds
   - Scraping and crawling
   - Parsing text files
3. Storing data
   - CSV tables
   - JSON and XML
4. Next meetings

Last week's exercise

Discussing the code

Reading a JSON file into a dict, looping over the dict

Task 1: Print all titles of all videos

```
1  import json
2
3  with open("/home/damian/pornexercise/xhamster.json") as fi:
4      data=json.load(fi)
5
6  for k,v in data.items()):
7      print (v["title"])
```

Reading a JSON file into a dict, looping over the dict

Task 1: Print all titles of all videos

```
1  import json
2
3  with open("/home/damian/pornexercise/xhamster.json") as fi:
4      data=json.load(fi)
5
6  for k,v in data.items()):
7      print (v["title"])
```

NB: You have to know (e.g., by reading the documentation of the dataset) that the key is called `title`

Reading a JSON file into a dict, looping over the dict

Task 1: Print all titles of all videos

```
1  import json
2
3  with open("/home/damian/pornexercise/xhamster.json") as fi:
4      data=json.load(fi)
5
6  for k,v in data.items()):
7      print (v["title"])
```

NB: You have to know (e.g., by reading the documentation of the dataset) that the key is called `title`

NB: `data` is in fact a dict of dicts, such that each value `v` is another dict.

For each of these dicts, we retrieve the value that corresponds to the key `title`

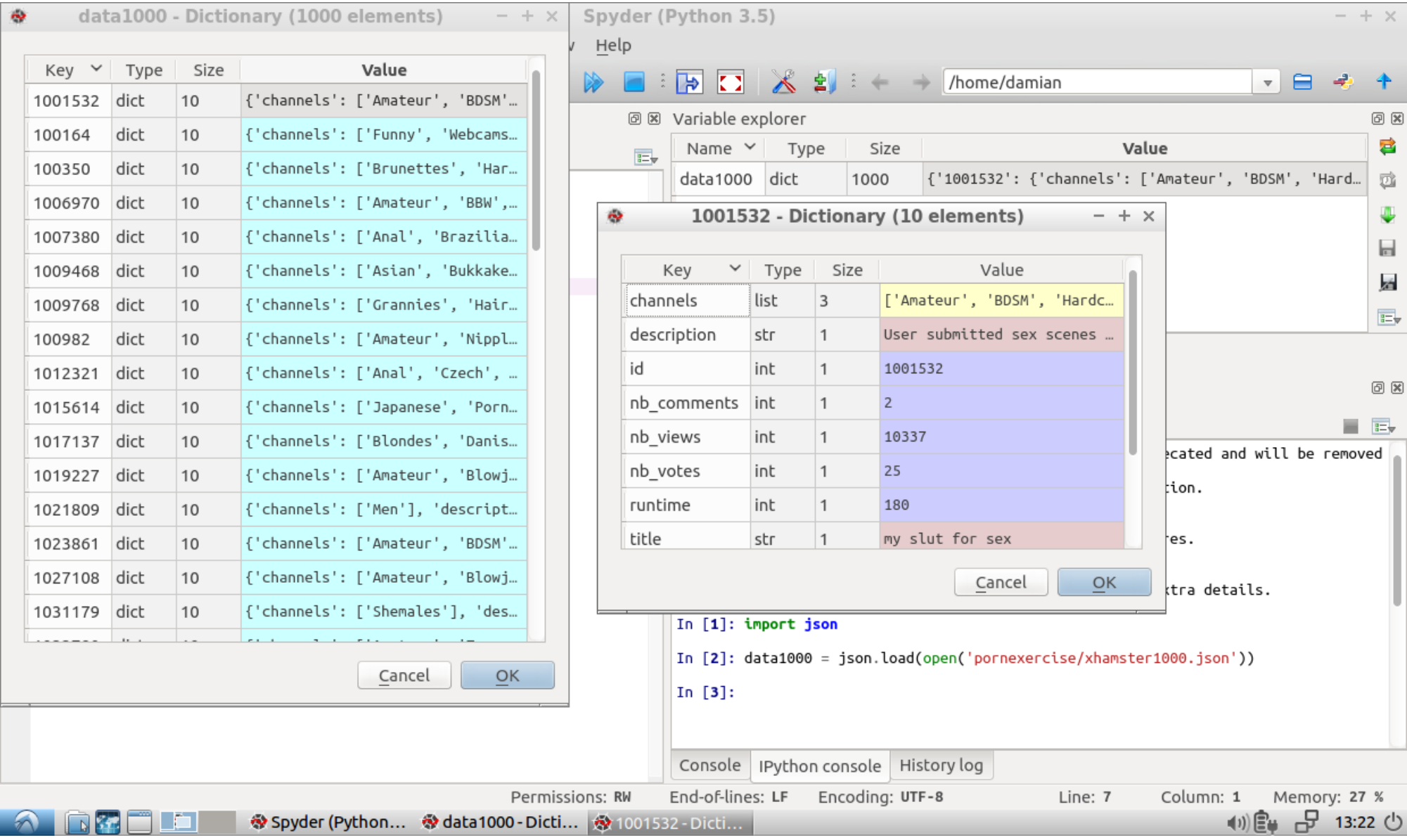
For the sake of completeness. . .

`.items()` returns a key-value *pair*, that's why we need to assign *two* variables in the for statement.
These alternatives would also work:

```
1 for v in data.values()):
2         print(v["title"])
```

```
1 for k in data:    #or: for k in data.keys():
2         print(data[k]["title"])
```

Do you see (dis-)advantages?


data1000 - Dictionary (1000 elements)
Key
Type
Size
Value
1001532 dict 10 {'channels': ['Amateur', 'BDSM'…
100164 dict 10 {'channels': ['Funny', 'Webcams…
100350 dict 10 {'channels': ['Brunettes', 'Har…
1006970 dict 10 {'channels': ['Amateur', 'BBW',…
1007380 dict 10 {'channels': ['Anal', 'Brazilia…
1009468 dict 10 {'channels': ['Asian', 'Bukkake…
1009768 dict 10 {'channels': ['Grannies', 'Hair…
100982 dict 10 {'channels': ['Amateur', 'Nippl…
1012321 dict 10 {'channels': ['Anal', 'Czech', …
1015614 dict 10 {'channels': ['Japanese', 'Porn…
1017137 dict 10 {'channels': ['Blondes', 'Danis…
1019227 dict 10 {'channels': ['Amateur', 'Blowj…
1021809 dict 10 {'channels': ['Men'], 'descript…
1023861 dict 10 {'channels': ['Amateur', 'BDSM'…
1027108 dict 10 {'channels': ['Amateur', 'Blowj…
1031179 dict 10 {'channels': ['Shemales'], 'des…
Cancel
OK
Spyder (Python 3.5)
Help
/home/damian
Variable explorer
Name
Type
Size
Value
data1000 dict 1000 {'1001532': {'channels': ['Amateur', 'BDSM', 'Hard…
1001532 - Dictionary (10 elements)
Key
Type
Size
Value
channels list 3 ['Amateur', 'BDSM', 'Hardc…
description str 1 User submitted sex scenes …
id int 1 1001532
nb_comments int 1 2
nb_views int 1 10337
nb_votes int 1 25
runtime int 1 180
title str 1 my slut for sex
Cancel
OK
ecated and will be removed
tion.
es.
tra details.
In [1]: import json
In [2]: data1000 = json.load(open('pornexercise/xhamster1000.json'))
In [3]:
Console
IPython console
History log
Permissions: RW
End-of-lines: LF
Encoding: UTF-8
Line: 7
Column: 1
Memory: 27 %
Spyder (Python…
data1000 - Dicti…
1001532 - Dicti…
13:22

Initializing variables, merging two lists, using a counter

Task 2: Average tags per video and most frequently used tags

```
1  from collections import Counter
2
3  alltags=[]
4  i=0
5  for k,v in data.items():
6      i+=1
7      alltags+=v["channels"]
8
9  print(len(alltags),"tags are describing",i,"different videos")
10 print("Thus, we have an average of",len(alltags)/i,"tags per video")
11
12 c=Counter(alltags)
13 print (c.most_common(100))
```

(there are other, more efficient ways of doing this)

Nesting blocks, using a defaultdict to count, error handling

Task 3: What porn category is most frequently commented on?

```
1  from collections import defaultdict
2
3  commentspercat=defaultdict(int)
4  for k,v in data.items():
5          for tag in v["channels"]:
6              try:
7                  commentspercat[tag]+=int(v["nb_comments"])
8              except:
9                  pass
10 print(commentspercat)
11 # if you want to print in a fancy way, you can do it like this:
12 for tag in sorted(commentspercat, key=commentspercat.get, reverse=True):
13     print( tag,"\t", commentspercat[tag])
```

A defaultdict is a normal dict, with the difference that the type of each value is pre-defined and it doesn't give an error if you look up a non-existing key

Nesting blocks, using a defaultdict to count, error handling

Task 3: What porn category is most frequently commented on?

```
1  from collections import defaultdict
2
3  commentspercat=defaultdict(int)
4  for k,v in data.items():
5          for tag in v["channels"]:
6              try:
7                  commentspercat[tag]+=int(v["nb_comments"])
8              except:
9                  pass
10 print(commentspercat)
11 # if you want to print in a fancy way, you can do it like this:
12 for tag in sorted(commentspercat, key=commentspercat.get, reverse=True):
13     print( tag,"\t", commentspercat[tag])
```

A defaultdict is a normal dict, with the difference that the type of each value is pre-defined and it doesn't give an error if you look up a non-existing key

NB: In line 7, we assume the value to be an int, but the datasets sometimes contains the string "NA" instead of a string representing an int. That's why we need the try/except construction

Adding elements to a list, sum() and len()

Task 4: Average length of descriptions

```
1  length=[]
2  for k,v in data.items():
3      length.append(len(v["description"]))
4
5  print ("Average length",sum(length)/len(length))
```

Merging vs appending

Merging:

```
1 l1 = [1,2,3]
2 l2 = [4,5,6]
3 l1 = l1 + l2
4 print(l1)
```

gives `[1,2,3,4,5,6]`

Appending:

```
1 l1 = [1,2,3]
2 l2 = [4,5,6]
3 l1.append(l2)
4 print(l1)
```

gives `[1,2,3,[4,5,6]]`

`l2` is seen as *one* element to append to `l1`

Tokenizing with .split()

Task 5: Most frequently used words

```
allwords=[]
for k,v in data.items():
    allwords+=v["description"].split()
c2=Counter(allwords)
print(c2.most_common(100))
```

`.split()` changes a string to a list of words.
`"This is cool".split()`
results in
`["This", "is", "cool"]`

Concluding remarks

Make sure you fully understand the code!

Re-read the corresponding chapters

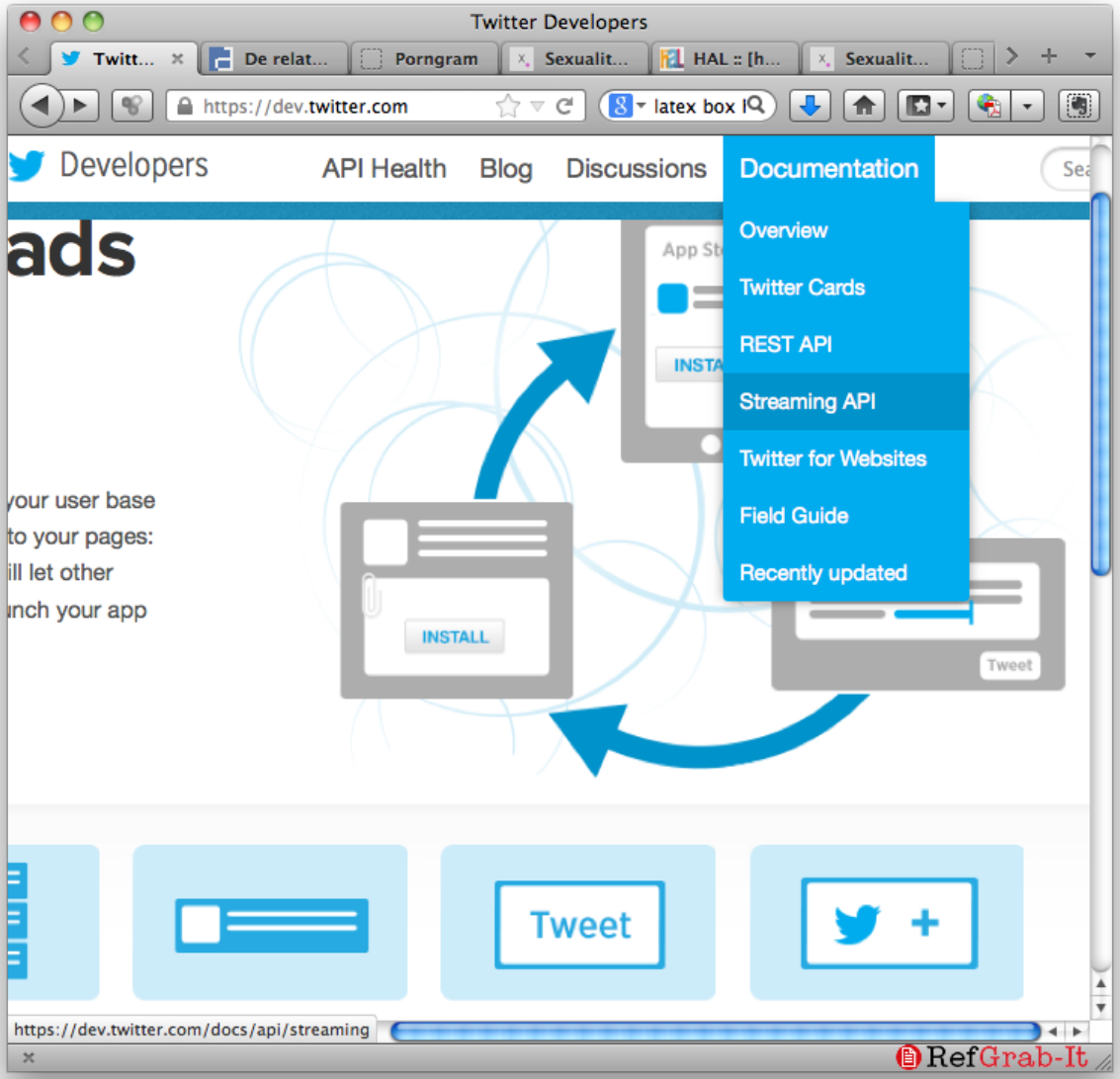
PLAY AROUND!!!

Data harvesting and storage

An overview of APIs, scrapers, crawlers, RSS-feeds, and different file formats

Collecting data: APIs

APIs


Twitter Developers
https://dev.twitter.com
Developers
API Health
Blog
Discussions
Documentation
Overview
Twitter Cards
REST API
Streaming API
Twitter for Websites
Field Guide
Recently updated
ads
your user base
to your pages:
ill let other
nch your app
INSTALL
Tweet
Tweet
https://dev.twitter.com/docs/api/streaming
RefGrab-It

Querying an API

```
1  # contact the Twitter API
2  auth = OAuth(access_key, access_secret, consumer_key, consumer_secret)
3  twitter = Twitter(auth = auth)
4
5  # get all info about the user 'username')
6  tweepinfo=twitter.users.show(screen_name=username)
7
8  # save his bio statement to the variable bio
9  bio=tweepinfo["description"])
10
11 # save his location to the variable location
12 location=tweepinfo["location"]
```

(abbreviated Python example of how to query the Twitter REST API)

Who offers APIs?

The usual suspects: Twitter, Facebook, Google – but also Reddit, Youtube, . . .

Who offers APIs?

The usual suspects: Twitter, Facebook, Google – but also Reddit, Youtube, . . .

If you ever leave your bag on a bus on Chicago

Who offers APIs?

The usual suspects: Twitter, Facebook, Google – but also Reddit, Youtube, ...

If you ever leave your bag on a bus on Chicago

...but do have Python on your laptop, watch this:
`https://www.youtube.com/watch?v=RrPZza_vZ3w`.
That guy queries the Chicago bus company's API to calculate when *exactly the vehicle* with his bag arrives the next time at the bus stop in front of his office.

(Yes, he tried calling the help desk before, but they didn't know. He got his bag back.)

APIs

Pro

- Structured data
- Easy to process automatically
- Can be directy embedded in your script

Morstatter, F., Pfeffer, J., Liu, H., & Carley, K. M. (2013). Is the sample good enough? Comparing data from Twitter's Streaming API with Twitter's Firehose. *International AAAI Conference on Weblogs and Social Media*.

APIs

Pro

- Structured data
- Easy to process automatically
- Can be directy embedded in your script

Con

- Often limitations (requests per minute, sampling, . . .)
- You have to trust the provider that he delivers the right content (⇒ Morstatter e.a., 2013)
- **Some APIs won't allow you to go back in time!**

Morstatter, F., Pfeffer, J., Liu, H., & Carley, K. M. (2013). Is the sample good enough? Comparing data from Twitter's Streaming API with Twitter's Firehose. *International AAAI Conference on Weblogs and Social Media*.

So we have learned that we can access an API directly.
But what if we have to do so 24/7?

So we have learned that we can access an API directly.
But what if we have to do so 24/7?

Collecting tweets with a tool running on a server that *does* query the API 24/7.

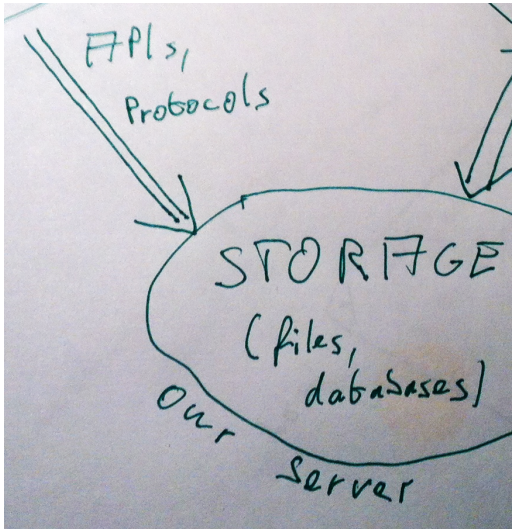
So we have learned that we can access an API directly.
But what if we have to do so 24/7?

Collecting tweets with a tool running on a server that *does* query the API 24/7.

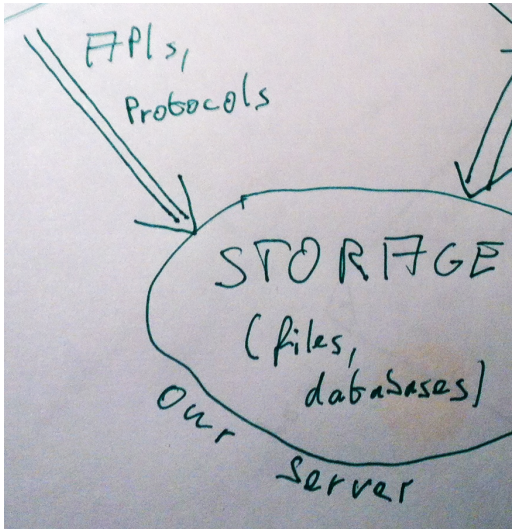
For example, we are running DMI-TCAT (Borra & Rieder, 2014) on a cloud computing platform.

Borra, E., & Rieder, B. (2014). Programmed method: Developing a toolset for capturing and analyzing tweets. *Aslib Journal of Information Management, 66*(3), 262–278. doi:10.1108/AJIM-09-2013-0094

DMI-TCAT



DMI-TCAT

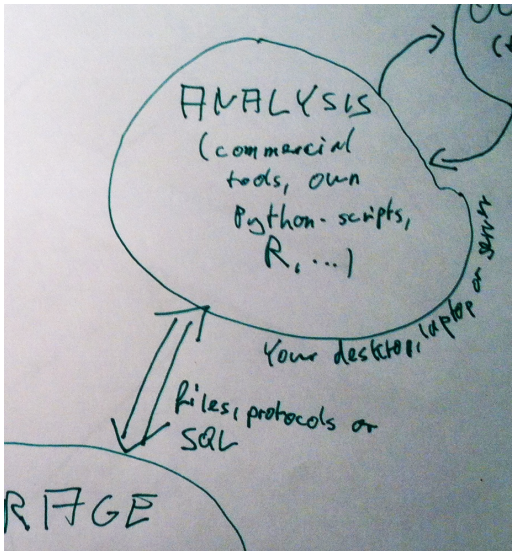


It queries the API and stores the result

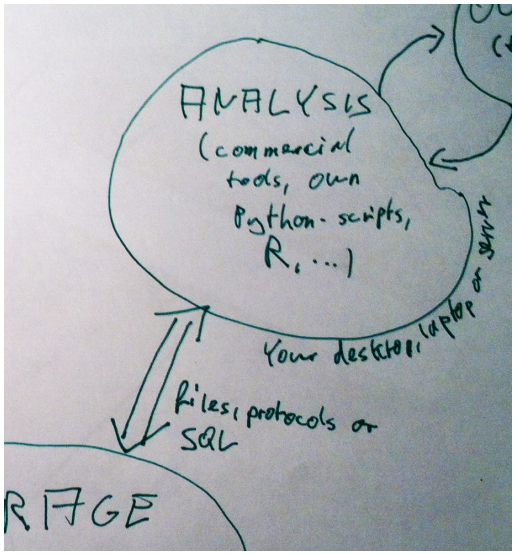
It continuosly calls the Twitter-API and saves all tweets containing specific hashtags to a MySQL-database.

You tell it once which data to collect – and wait some months.

DMI-TCAT


ANALYSIS
(commercial
tools, own
Python-scripts,
R, ...)
Your desktop, laptop or server
files, protocols or
SQL
RANGE

DMI-TCAT



Retrieving the data for analysis

You *could* access the MySQL-database directly.

But that's not neccessary, as DMI-TCAT has a nice interface that allows you to save the data as a CSV file.
(and offers tools for direct analysis as well)

Collecting data: RSS feeds

RSS feeds

What's that?

- A structured (XML) format in which for example news sites and blogs offer their content

RSS feeds

What's that?

- A structured (XML) format in which for example news sites and blogs offer their content
- You get only the new news items (and that's great!)

RSS feeds

What's that?

- A structured (XML) format in which for example news sites and blogs offer their content
- You get only the new news items (and that's great!)
- Title, teaser (or full text), date and time, link

RSS feeds

What's that?

- A structured (XML) format in which for example news sites and blogs offer their content
- You get only the new news items (and that's great!)
- Title, teaser (or full text), date and time, link

`http://www.nu.nl/rss`

RSS feeds

Pro

- *One* protocol for all services
- Easy to use

Con

- Full text often not included, you have to download the link separately ($\Rightarrow$ Problems associated with scraping)
- **You can't go back in time!** But we have archived a lot of RSS feeds

Collecting data: Scraping and crawling

Scraping and crawling

If you have no chance of getting already structured data via one of the approaches above

Scraping and crawling

If you have no chance of getting already structured data via one of the approaches above

- Download web pages, try to identify the structure yourself
- You have to *parse* the data

Scraping and crawling

If you have no chance of getting already structured data via one of the approaches above

- Download web pages, try to identify the structure yourself
- You have to *parse* the data
- Can get very complicated (depending on the specific task), especially if the structure of the web pages changes

Further reading:
`http://scrapy.org`
`https://github.com/anthonydb/python-get-started/blob/master/5-web-scraping.py`

Collecting data: Parsing text files

For messy input data or for semi-structured data

Guiding question: Can we identify some kind of pattern?

For messy input data or for semi-structured data

Guiding question: Can we identify some kind of pattern?

Examples

- Lewis, Zamith, & Hermida (2013) had a corrupt CSV-file

Lewis, S. C., Zamith, R., & Hermida, A. (2013). Content analysis in an era of Big Data: A hybrid approach to computational and manual methods. *Journal of Broadcasting & Electronic Media, 57*(1), 34–52. doi:10.1080/08838151.2012.761702

For messy input data or for semi-structured data

Guiding question: Can we identify some kind of pattern?

Examples

- Lewis, Zamith, & Hermida (2013) had a corrupt CSV-file
- LexisNexis gives you a chunk of text (rather than, e.g., a structured JSON or XML object)

For messy input data or for semi-structured data

Guiding question: Can we identify some kind of pattern?

Examples

- Lewis, Zamith, & Hermida (2013) had a corrupt CSV-file
- LexisNexis gives you a chunk of text (rather than, e.g., a structured JSON or XML object)

But in both cases, as long as you can find any pattern or structure in it, you can try to write a Python script to *parse* the data.

1 of 200 DOCUMENTS

De Telegraaf

21 maart 2014 vrijdag

Safe

SECTION: Binnenland; Blz. 3

LENGTH: 257 woorden

Na ruim twee jaar onderhandelen zijn de EU-landen, de Europese Commissie en het Europees Parlement het eens geworden over een nieuwe Europese bankenunie. Een verstrekkende afspraak die noodzakelijk bleek na de diepe financiële crisis,

zorgen dat we niet in zo'n situatie terecht kunnen komen. Europese banken mogen niet meer omvallen. De nu gemaakte afspraken kunnen daarvoor zorgen.

LOAD-DATE: 21 March 2014

LANGUAGE: DUTCH; NEDERLANDS

PUBLICATION-TYPE: Krant

JOURNAL-CODE: Telegraaf

2 of 200 DOCUMENTS

```
tekst={}
section={}
length={}
...
...
with open(bestandsnaam) as f:
  for line in f:
    line=line.replace("\r","")
    if line=="\n":
      continue
    matchObj=re.match(r"\s+(\d+) of (\d+) DOCUMENTS",line)
    if matchObj:
      artikelnr= int(matchObj.group(1))
      tekst[artikelnr]=""
      continue
    if line.startswith("SECTION"):
      section[artikelnr]=line.replace("SECTION: ","").rstrip("\n")
    elif line.startswith("LENGTH"):
      length[artikelnr]=line.replace("LENGTH: ","").rstrip("\n")
...
...
...

    else:
      tekst[artikelnr]=tekst[artikelnr]+line
```

Storing data: CSV tables

CSV-files

Always a good choice

- *All* programs can read it
- Even human-readable in a simple text editor:
- Plain text, with a comma (or a semicolon) denoting column breaks
- No limits regarging the size
- But: several dialects (e.g., , vs. ; as delimiter)

A CSV-file with tweets

```
1 text,to_user_id,from_user,id,from_user_id,iso_language_code,source,
      profile_image_url,geo_type,geo_coordinates_0,geo_coordinates_1,
      created_at,time
2 :-) #Lectrr #wereldleiders #uitspraken #Wikileaks #klimaattop http://t.
      co/Udjpk48EIB,,henklbr,407085917011079169,118374840,nl,web,http://
      pbs.twimg.com/profile_images/378800000673845195/
      b47785b1595e6a1c63b93e463f3d0ccc_normal.jpeg,,0,0,Sun Dec 01
      09:57:00 +0000 2013,1385891820
3 Wat zijn de resulaten vd #klimaattop in #Warschau waard? @EP_Environment
       ontmoet voorzitter klimaattop @MarcinKorolec http://t.co/4
      Lmiaopf60,,Europarl_NL,406058792573730816,37623918,en,<a href="http
      ://www.hootsuite.com" rel="nofollow">HootSuite</a>,http://pbs.twimg
      .com/profile_images/2943831271/
      b6631b23a86502fae808ca3efde23d0d_normal.png,,0,0,Thu Nov 28
      13:55:35 +0000 2013,1385646935
```

Storing data: JSON and XML

JSON and XML

Great if we have a nested data structure

JSON and XML

Great if we have a nested data structure

- Items within feeds

JSON and XML

Great if we have a nested data structure

- Items within feeds
- Personal data within authors within books

JSON and XML

Great if we have a nested data structure

- Items within feeds
- Personal data within authors within books
- Tweets within followers within users

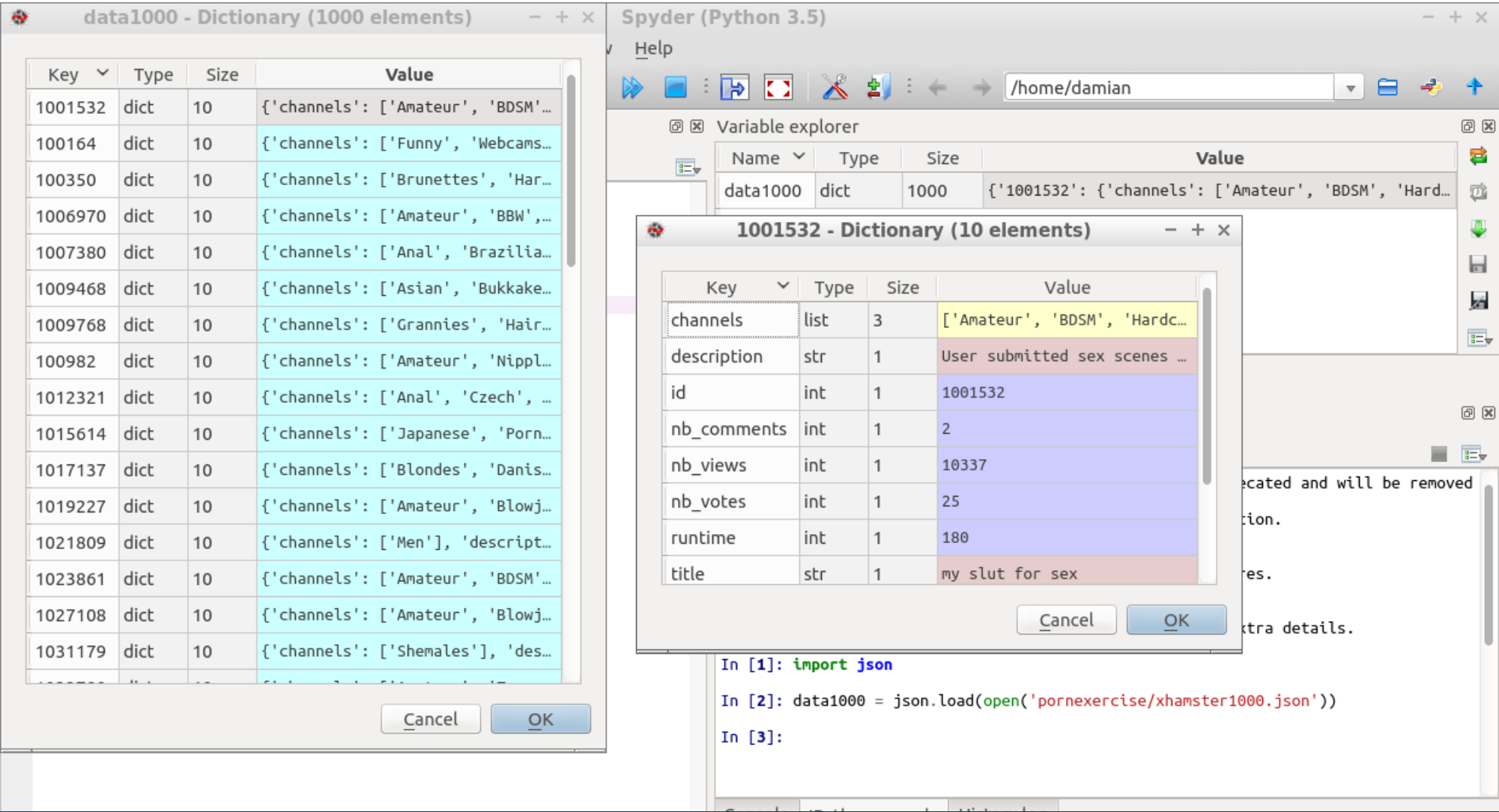
A JSON object containing GoogleBooks data

```
1 {'totalItems': 574, 'items': [{'kind': 'books#volume', 'volumeInfo': {'
      publisher': '"O\'Reilly Media, Inc."', 'description': u'Get a
      comprehensive, in-depth introduction to the core Python language
      with this hands-on book. Based on author Mark Lutz\u2019s popular
      training course, this updated fifth edition will help you quickly
      write efficient, high-quality code with Python. It\u2019s an ideal
      way to begin, whether you\u2019re new to programming or a
      professional developer versed in other languages. Complete with
      quizzes, exercises, and helpful illustrations, this easy-to-follow,
       self-paced tutorial gets you started with both Python 2.7 and 3.3\
      u2014 the
2 ...
3 ...
4 'kind': 'books#volumes'}
```

An XML object containing an RSS feed

```
...
<item>
        <title>Agema doet aangifte tegen Samsom en Spekman</title>
        <link>http://www.nu.nl/politiek/3743441/agema-doet-aangifte-
            samsom-en-spekman.html</link>
         <guid>http://www.nu.nl/politiek/3743441/index.html</guid>
         <description>PVV-Kamerlid Fleur Agema gaat vrijdag aangifte doen
              tegen PvdA-leider Diederik Samsom en PvdA-voorzitter Hans
             Spekman wegens uitspraken die zij hebben gedaan over
             Marokkanen. </description>
         <pubDate>Thu, 03 Apr 2014 21:58:48 +0200</pubDate>
        <category>Algemeen</category>
         <enclosure url="http://bin.snmmd.nl/m/m1mxwpka6nn2_sqr256.jpg"
             type="image/jpeg" />
         <copyrightPhoto>nu.nl</copyrightPhoto>
</item>
...
```

It's the same as our "dict of dicts"/"dict of lists"/... data model!



Next meetings

Next meetings

Wednesday, 23–3

Writing some first data collection scripts