

# Big Data and Automated Content Analysis

Week 3 – Monday

» Data harvesting and storage «

Anne Kroon

a.c.kroon@uva.nl  
@annekroon

Afdeling Communicatiewetenschap  
Universiteit van Amsterdam

April 15, 2019

# 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
  - Databases
- ④ Next meetings

## **Last week's exercise**

Discussing the code

## Working with a subset of the data

What to do if you want to work with a smaller subset of the data?  
Taking a random sample of a dict (to practice with a smaller subset of the data)

```
1 import json
2 import random
3
4 with open("/home/damian/pornexercise/xhamster.json") as fi:
5     data=json.load(fi)
6
7 short_data = dict(random.sample(data.items(), 10))
```

NB: This will return a random sample of 10 items in the dictionary

NB: Obviously, you can select samples in any size you want:

`dict(random.sample(data.items(), 357))` allows you to get 357 items

# Working with a subset of the data

What to do if you want to work with a smaller subset of the data?

Taking the first 10 elements in a list:

```
1 mylist_short = mylist[:10]
```

# 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`

# What to do if you do not know the structure of the dataset?

Inspecting your data: use the functions `type()` and `len()` and/or the dictionary method `.keys()`

```
1 len(data)
2 type(data)
3 data.keys()
```

`len()` returns the number of items of an object; `type()` returns the type object; `.keys()` returns a list of all available keys in the dictionary

# What to do if you do not know the structure of the dataset?

Inspecting your data: use the module pprint

```
1 from pprint import pprint
2 pprint(data)
```

(but better do this on a smaller subset of the data!)



```

'42197': {'channels': ['Amateur'],
'description': 'My wife 20 years ago....',
'id': 42197,
'nb_comments': 10,
'nb_views': 45296,
'nb_votes': 190,
'runtime': 207,
'title': 'my wife 20 years ago',
'upload_date': '2008-06-28',
'uploader': 'NA'},
'541406': {'channels': ['Amateur', 'Masturbation', 'Webcams'],
'description': 'NA',
'id': 541406,
'nb_comments': 2,
'nb_views': 3006,
'nb_votes': 11,
'runtime': 115,
'title': 'a little camshow',
'upload_date': '2011-01-29',
```



## 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?**

# 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.extend(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

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))
```

## Extending (merging) vs appending

Merging:

```
1 l1 = [1,2,3]
2 l2 = [4,5,6]
3 # either:
4 l1 = l1 + l2
5 # or:
6 l1.extend(l2)
7 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

```
1 allwords=[]
2 for k,v in data.items():
3     allwords+=v["description"].split()
4 c2=Counter(allwords)
5 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!!!**

Suggestion: Exercise in Appendix B (Analyze your own WhatsApp history!)

## **Data harvesting and storage**

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

## Collecting data: APIs

# APIs



# 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, ...

## 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](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 (JSON!)
- Easy to process automatically
- Can be directly 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!**

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.



Collecting data:  
RSS feeds

# 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 feed

```
<rss xmlns:atom="http://www.w3.org/2005/Atom" version="2.0" xmlns:media="http://search.yahoo.com/mr
<link>https://www.nu.nl/</link><description>Het laatste nieuws het eerst op NU.nl</description><ato
nl</language><copyright>Copyright (c) 2018, NU</copyright><lastBuildDate>Sun, 18 Feb 2018 13:34:17
<atom:logo>https://www.nu.nl/static/img/atoma/images/logos/rss-logo-250x40.png</atom:logo><item><ti
<link>https://www.nu.nl/olympische-winterspelen-pyeongchang-2018/5139494/kodaira-wint-olympische-50
olympisch goud veroverd op de 500 meter. De Nederlandse vrouwen grepen zoals verwacht naast een med
</description><pubDate>Sun, 18 Feb 2018 13:32:27 +0100</pubDate><guid isPermaLink="false">https://w
url="https://media.nu.nl/m/jllxny7aub8u_sqr256.jpg/kodaira-wint-olympische-500-meter-mors-knap-zesd
<category>Winterspelen</category><dc:creator>NU.nl</dc:creator><dc:rights>copyright photo: ANP</dc:
2018/5139461/verweij-vreest-ondanks-matig-optreden-niet-plek-in-ploegachtervolging.html" type="text
in ploegachtervolging"></atom:link><atom:link href="http://nu.nl/olympische-winterspelen-pyeongchan
type="text/html" rel="related" title="Nederlandse achtervolgingsploeg met enige moeite naar halve f
pyeongchang-2018/5139386/herstelde-verbij-heeft-geen-idee-hij-kan-olympische-500-meter.html" type="
olympische 500 meter"></atom:link><atom:link href="http://nu.nl/olympische-winterspelen-pyeongchang
lorentzen.html" type="text/html" rel="related" title="Verbij als laatste Nederlander in actie op 50
krijgt tegen Heracles eerste basisplek sinds terugkeer bij Feyenoord</title><link>https://www.nu.nl
feyenoord.html</link><description>Robin van Persie heeft zondag in de thuiswedstrijd tegen Heracles
</description><pubDate>Sun, 18 Feb 2018 13:26:20 +0100</pubDate><guid isPermaLink="false">https://w
url="https://media.nu.nl/m/9ixxvr8atmaf_sqr256.jpg/van-persie-krijgt-heracles-eerste-basisplek-bind
<category>Algemeen</category><category>Voetbal</category><dc:creator>NU.nl</dc:creator><dc:rights>c
eerste toernooizege in anderhalf jaar</title><link>https://www.nu.nl/sport-overig/5139473/luiten-bo
Luiten heeft zondag de Oman Open gewonnen. Voor de Nederlandse golfer is het zijn eerste toernooize
18 Feb 2018 13:27:52 +0100</pubDate><guid isPermaLink="false">https://www.nu.nl/-/5139473/</guid><e
eerste-toernooizege-in-anderhalf-jaar.jpg" length="0" type="image/jpeg"></enclosure><category>Sport
<dc:rights>copyright photo: Getty Images</dc:rights></item><item><title>Dode gevonden in woning Zwo
<link>https://www.nu.nl/binnenland/5139464/dode-gevonden-in-woning-zwolle-politie-gaat-van-misdrijf
overschot aangetroffen. De politie gaat ervan uit dat de dode slachtoffer is geworden van een misdr
vermist, is nog onduidelijk.</description><pubDate>Sun, 18 Feb 2018 13:08:55 +0100</pubDate><guid i
url="https://media.nu.nl/m/i9nxbu6ag2e9_sqr256.jpg/dode-gevonden-in-woning-zwolle-politie-gaat-van-
<category>Algemeen</category><category>Binnenland</category><dc:creator>ANP</dc:creator><dc:rights>
&#x2013;astronauten-tv&#x2013; uit jaren &#x2013;70</title><link>https://www.nu.nl/tech/5139467/brit-speelt-modern
spelcomputers werkend gekregen op de Videosphere. De Videosphere komt uit de jaren &#x2013;70 en ziet er
verschillende kabels gebruiken. De Brit speelde onder meer Mario Kart 8, Forza 7 en Horizon Zero Da
isPermaLink="false">https://www.nu.nl/-/5139467/</guid><enclosure url="https://media.nu.nl/m/wmqxt8
length="0" type="image/jpeg"></enclosure><category>Tech</category><dc:creator>NU.nl</dc:creator></i
ploegachtervolging</title><link>https://www.nu.nl/olympische-winterspelen-pyeongchang-2018/5139461/
</description>Koen Verweij kende zondag mede door problemen met zijn linkerschaats geen goed optrede
Hollander is voorlopig niet bang dat hij in de halve finales wordt vervangen door Patrick Roest.</d
isPermaLink="false">https://www.nu.nl/-/5139461/</guid><enclosure url="https://media.nu.nl/m/co2x9m
ploegachtervolging.jpg" length="0" type="image/jpeg"></enclosure><category>Winterspelen</category><
```

# Parsing RSS feeds

```
1 import feedparser
2 d = feedparser.parse(rss_body)
3     for post in d.entries:
4         teaser=re.sub(r"\n|\r\\|\\t"," ",post.description)
5         link = post.link
6     ...
7     ...
```

(abbreviated Python example of how to parse an RSS feed)

# RSS feeds

## Pro

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

## Con

- Full text often not included, you have to download the link separately (⇒ 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

- 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:](https://github.com/anthonydb/python-get-started/blob/master/5-web-scraping.py)

[//github.com/anthonydb/python-get-started/blob/master/5-web-scraping.py](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

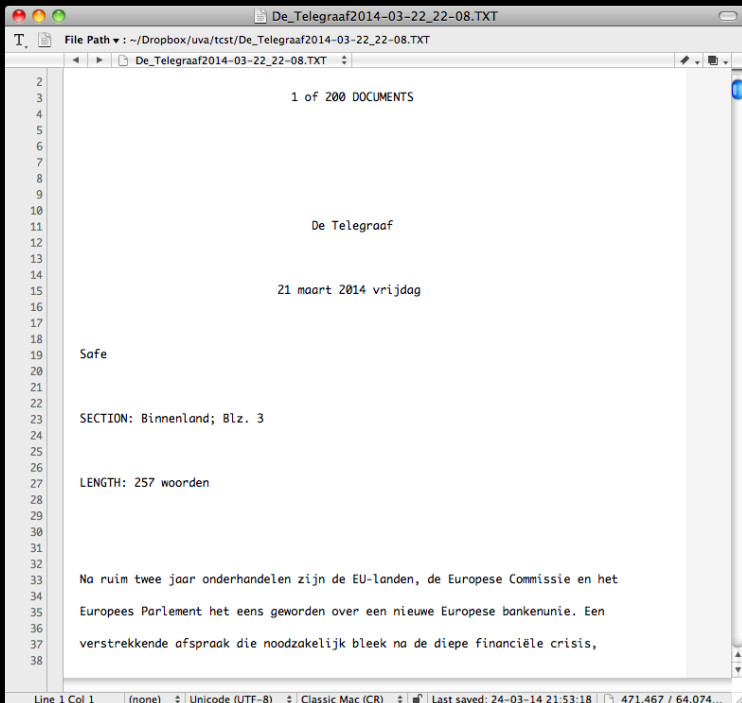
Guiding question: Can we identify some kind of pattern?

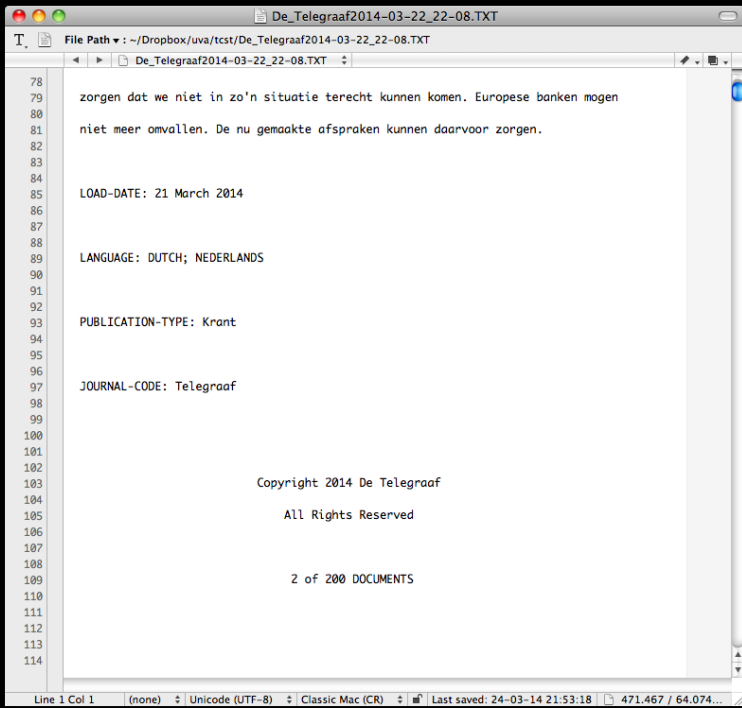
## Examples

- 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.

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





```
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 regarding 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 resultaten 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

- 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

```
1  ...
2  <item>
3  <title>Agema doet aangifte tegen Samsom en Spekman</title>
4  <link>http://www.nu.nl/politiek/3743441/agema-doet-aangifte-samsom-en-
    spekman.html</link>
5  <guid>http://www.nu.nl/politiek/3743441/index.html</guid>
6  <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>
7  <pubDate>Thu, 03 Apr 2014 21:58:48 +0200</pubDate>
8  <category>Algemeen</category>
9  <enclosure url="http://bin.snmmnd.nl/m/m1mxwpka6nn2_sqr256.jpg" type="
    image/jpeg" />
10 <copyrightPhoto>nu.nl</copyrightPhoto>
11 </item>
12 ...
```

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

The screenshot shows the Spyder Python IDE interface. On the left, a window titled "data1000 - Dictionary (1000 elements)" displays a table of data. The table has columns for Key, Type, Size, and Value. The values are lists of strings representing different categories. On the right, a window titled "1001532 - Dictionary (10 elements)" provides a detailed view of a specific entry. This window also has columns for Key, Type, Size, and Value, showing fields like channels, description, id, nb\_comments, nb\_views, nb\_votes, runtime, and title. Below the detailed view, a code editor shows the following Python code:

```
In [1]: import json
In [2]: data1000 = json.load(open('pornexercise/xhamster1000.json'))
In [3]:
```

# When it gets bigger: Databases

Out of scope of this course, but if it get's bigger, you could use a database, e.g.

- For (relational) tables: SQL, e.g. mysql
- For dumping dicts, messy content, and text: NoSQL, e.g. MongoDB or ElasticSearch

They are easy to install on Linux and can be addressed from Python!

Günther, E., Trilling, D. C., & van de Velde, R. N. (2018). But how do we store it? Data architecture in the social-scientific research process. In C. M. Stuetzer, M. Welker, & M. Egger (Eds.), *Computational social science in the age of Big Data. Concepts, methodologies, tools, and applications* (pp. 161-187). Cologne: Herbert von Halem.

## Next meetings

# Next meetings

Thursday, 18–2

Writing some first data collection scripts