# Big Data and Automated Content Analysis

Week 6 – Monday »Web scraping«

Damian Trilling

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

Afdeling Communicatiewetenschap Universiteit van Amsterdam

12 March 2018



# Today

- 1 A first intro to parsing webpages
- 2 OK, but this surely can be doe more elegantly? Yes!
- Scaling up
- 4 Next steps

Let's have a look of a webpage with comments and try to understand the underlying structure.



#### **REAGUURSELS**

Tsja, ik zou m ook niet in mn buurt willen hebben, die vreemde vogel.

Lepo | 05-05-14 | 11:13

Ţ

Volgend jaar stropdascontrole voor de heren en hoedjescheck voor de dames. De volledige lijst van goedgekeurde kleding kunt u vinden op Postbus51.nl.

Uiteraard bent u extra welkom als u Abercrombie & Fitch draagt.

rara | 05-05-14 | 11:15

Ţ.

Gewoon even de afdeling psychiatrie bellen, wie ze missen: Klaar!

Mazzeltov | 05-05-14 | 11:16

Ţ.

netjes opgelost toch?

--sql error-- | 05-05-14 | 11:16

\_\_\_\_

```
命
         i view-source:https://www.geenstijl.nl/3945571/das_toch_niet_normaal/
: 🔞 Aan de slag 🔞 Press This 👿 Atlassian Cloud
n class="divider"> </span>
n class="datetime">05-05-14 | 23:34</span>
lass="reportcomment" title="Deze reactie is in overtreding met de huisregels."></a>
er>
e>
rticle class="comment col-xs-12 no-v-padding"
ommentid="192631041"
92631041">
ss="anchor-pos" id="cid 192631041"></div>
lass="cmt-content">Een VZ800 Marauder, wat een giga kutmotor is dat. Na 10 km. heb je al
200
pan class="username">tiswat</span>
```

## Let's make a plan!

A first intro to parsing webpages

### Which elements from the page do we need?

- What do they mean?
- How are they represented in the source code?

### How should our output look like?

- What lists do we want?

And how can we achieve this?



 $\ \, \textbf{ D} \text{Ownload the page}$ 

- Download the page
  - Possibly taking measures to deal with cookie walls, being blocked, etc.

- Download the page
  - Possibly taking measures to deal with cookie walls, being blocked, etc.
- **2** Remove all line breaks (\n, but maybe also \n\r or \r) and TABs (\t): We want one long string



- Download the page
  - Possibly taking measures to deal with cookie walls, being blocked, etc.
- **2** Remove all line breaks (\n, but maybe also \n\r or \r) and TABs (\t): We want one long string
- Try to isolate the comments
  - Do we see any pattern in the source code? ⇒ last week: if we can see a pattern, we can describe it with a regular expression

```
import requests
    import re
2
3
    response = requests.get('http://kudtkoekiewet.nl/setcookie.php?t=http://
        www.geenstijl.nl/mt/archieven/2014/05/das_toch_niet_normaal.html')
5
    tekst=response.text.replace("\n"," ").replace("\t"," ")
6
7
    comments=re.findall(r'<div class="cmt-content">(.*?)</div>',tekst)
8
    print("There are",len(comments), "comments")
9
    print("These are the first two:")
10
    print(comments[:2])
11
```

#### Some remarks

### The regexp

.\*? instead of .\* means lazy matching. As .\* matches
everything, the part where the regexp should stop would not
be analyzed (greedy matching) – we would get the whole rest
of the document (or the line, but we removed all line breaks).

#### Some remarks

### The regexp

- .\*? instead of .\* means lazy matching. As .\* matches everything, the part where the regexp should stop would not be analyzed (greedy matching) we would get the whole rest of the document (or the line, but we removed all line breaks).
- The parentheses in (.\*?) make sure that the function only returns what's between them and not the surrounding stuff (like <div> and </div>)

#### Some remarks

### The regexp

- .\*? instead of .\* means lazy matching. As .\* matches
  everything, the part where the regexp should stop would not
  be analyzed (greedy matching) we would get the whole rest
  of the document (or the line, but we removed all line breaks).
- The parentheses in (.\*?) make sure that the function only returns what's between them and not the surrounding stuff (like <div> and </div>)

#### Optimization

- Parse usernames, date, time, ...
- Replace tags

# Further reading

### Doing this with other sites?

- It's basically puzzling with regular expressions.
- Look at the source code of the website to see how well-structured it is.

OK, but this surely can be doe more elegantly? Yes!

### Geenstijl-example

 Worked well (and we could do it with the knowledge we already had)

### Geenstijl-example

- Worked well (and we could do it with the knowledge we already had)
- But we can also use existing parsers (that can interpret the structure of the html page)



### Geenstijl-example

- Worked well (and we could do it with the knowledge we already had)
- But we can also use existing parsers (that can interpret the structure of the html page)
- especially when the structure of the site is more complex



### Geenstijl-example

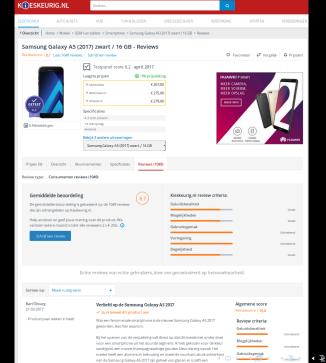
- Worked well (and we could do it with the knowledge we already had)
- But we can also use existing parsers (that can interpret the structure of the html page)
- especially when the structure of the site is more complex

The following example is based on http://www.chicagoreader.com/chicago/best-of-chicago-2011-food-drink/BestOf?oid=4106228. It uses the module 1xml



### What do we need?

- the URL (of course)
- the XPATH of the element we want to scrape (you'll see in a minute what this is)



#### Verliefd op de Samsung Galaxy A5 2017

✓ Ja, ik beveel dit product aan

Wat een fenomenale smartphone is de nieuwe Samsung Galaxy A5 2017 geworden, lees hier waarom.

Bij het openen van de verpakking valt direct op dat dit toestel niet onder doet voor een smartphone uit het duurste segment. Ik heb gekozen voor de kleur sand/gold, een mooie champagneachtige gouden kleur die erg opvalt. Het toestel heeft een aluminium behuizing en zowel de voorkant als de achterkant van de

Samsung Galaxy A5 2017 zijn geher vingerafdruksensor aanwezig. Dit a premium look and feel, prachtig! In accessoires die je nodig hebt zoals hoofdtelefoon én de bijzondere US USB-C kabel kan je dit toestel versn Copy
Select All
Search Google for "Wat een fenomen..."
View Selection Source
Inspect Element (Q)

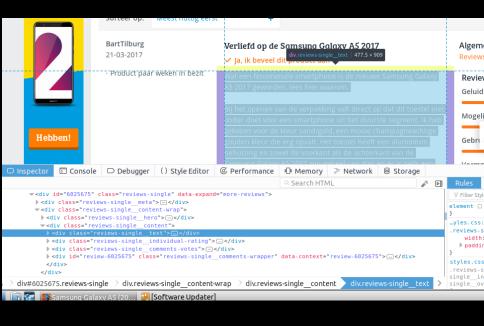
toestel heeft echt alles aan boord wat je mag verwachten en zelfs

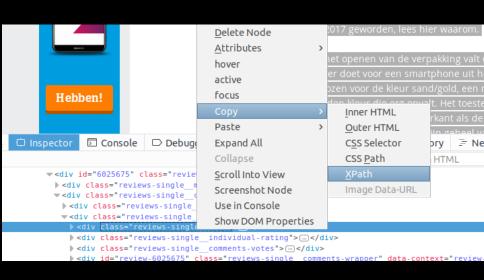
Algemene scor Reviewscore | 10

Review criteria

Mogelijkheden

Gebruiksgemak





There are multiple different XPATHs to address a specific element. Some things to play around with:

 // means 'arbitrary depth' (=may be nested in many higher levels)

There are multiple different XPATHs to address a specific element. Some things to play around with:

- // means 'arbitrary depth' (=may be nested in many higher levels)
- \* means 'anything'. (p[2] is the second paragraph, p[\*] are all

There are multiple different XPATHs to address a specific element. Some things to play around with:

- // means 'arbitrary depth' (=may be nested in many higher levels)
- \* means 'anything'. (p[2] is the second paragraph, p[\*] are all
- If you want to refer to a specific attribute of a HTML tag, you can use @. For example, every
  - \*[@id="reviews-container"] would grap a tag like <div id=reviews-container" class="'user-content'

There are multiple different XPATHs to address a specific element. Some things to play around with:

- // means 'arbitrary depth' (=may be nested in many higher levels)
- \* means 'anything'. (p[2] is the second paragraph, p[\*] are all
- If you want to refer to a specific attribute of a HTML tag, you can use @. For example, every
  - \*[@id="reviews-container"] would grap a tag like <div id=reviews-container" class="'user-content'
- Let the XPATH end with /text() to get all text

There are multiple different XPATHs to address a specific element. Some things to play around with:

- // means 'arbitrary depth' (=may be nested in many higher levels)
- \* means 'anything'. (p[2] is the second paragraph, p[\*] are all
- If you want to refer to a specific attribute of a HTML tag, you can use @. For example, every
  - \*[@id="reviews-container"] would grap a tag like <div id=reviews-container" class="'user-content'
- Let the XPATH end with /text() to get all text
- Have a look at the source code (via 'inspect elements') of the web page to think of other possible XPATHs!



### Let's test it!

https://www.kieskeurig.nl/smartphone/product/3518003-samsung-galaxy-a5-2017-zwart/reviews

# Let's scrape them!

```
from lxml import html
    import requests
3
    htmlsource = requests.get("https://www.kieskeurig.nl/smartphone/product
         /3518003-samsung-galaxy-a5-2017-zwart/reviews").text
    tree = html.fromstring(htmlsource)
6
    reviews = tree.xpath('//*[@class="reviews-single_ text"]/text()')
7
8
    # remove empty reviews
10
    reviews = [r.strip() for r in reviews if r.strip()!=""]
11
12
    print (len(reviews), "reviews scraped. Showing the first 60 characters of
         each:")
    i = 0
13
    for review in reviews:
14
       print("Review",i,":",review[:60])
15
       i+=1
16
```

# The output – perfect!

```
61 reviews scraped. Showing the first 60 characters of each:
Review 0 : Wat een fenomenale smartphone is de nieuwe Samsung Galaxy A5
```

Review 1 : Bij het openen van de verpakking valt direct op dat dit toes

Review 2: Tijd om deze badboy eens aan de tand te voelen dus direct he

Review 3: Na een volle week dit toestel gebruikt te hebben kan ik het

Review 4: ik ben verliefd op de Samsung Galaxy A5 2017!

Scaling up

## But this was on one page only, right?

Next step: Repeat for each relevant page.

### Possibility 1: Based on url schemes

```
If the url of one review page is https://www.hostelworld.com/hosteldetails.php/ClinkNOORD/Amsterdam/93919/reviews?page=2 ...then the next one is probably?
```

## But this was on one page only, right?

Next step: Repeat for each relevant page.

#### Possibility 1: Based on url schemes

```
If the url of one review page is https://www.hostelworld.com/hosteldetails.php/ClinkNOORD/Amsterdam/93919/reviews?page=2...then the next one is probably?
```

⇒ you can construct a list of all possible URLs:

# But this was on one page only, right?

Next step: Repeat for each relevant page.

### Possibility 2: Based on XPATHs

Use XPATH to get the url of the next page (i.e., to get the link that you would click to get the next review)

### Recap

#### General idea

- 1 Identify each element by its XPATH (look it up in your browser)
- Read the webpage into a (loooooong) string
- 3 Use the XPATH to extract the relevant text into a list (with a module like lxml)
- **4** Do something with the list (preprocess, analyze, save)
- 6 Repeat

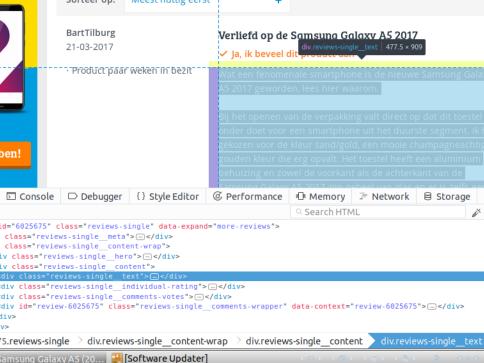
Alternatives: scrapy, beautifulsoup, regular expressions, ...



#### Last remarks

### There is often more than one way to specify an XPATH

- Sometimes, you might want to use a different suggestion to be able to generalize better (e.g., using the attributes rather than the tags)
- ② in that case, it makes sense to look deeper into the structure of the HTML code, for example with "Inspect Element" and use that information to play around with in the XPATH Checker



Next steps

#### From now on...

#### ... focus on individual projects!



## Wednesday

- Write a scraper for a website of your choice!
- Prepare a bit, so that you know where to start
- Suggestions: Review texts and grades from http://iens.nl, articles from a news site of your choice, ...

