

# Big Data and Automated Content Analysis

Week 5 – Friday

Automated content analysis with NLP and  
regular expressions

Damian Trilling

d.c.trilling@uva.nl

@damian0604

[www.damiantrilling.net](http://www.damiantrilling.net)

Afdeling Communicatiewetenschap  
Universiteit van Amsterdam

4 May 2018

# Today

- 1 ACA using regular expressions
  - Bottom-up vs. top-down
  - What is a regexp?
  - Using a regexp in Python
- 2 Natural Language Processing
  - Stopword removal
  - Stemming
  - Parsing sentences
- 3 Take-home message & next steps

# Automated content analysis using regular expressions

# Bottom-up vs. top-down

## Bottom-up

- Count most frequently occurring words (Week 2)
- Maybe better: Count combinations of words  $\Rightarrow$  Which words co-occur together? (Chapter 9, not obligatory)

We *don't* specify what to look for in advance

# Bottom-up vs. top-down

## Bottom-up

- Count most frequently occurring words (Week 2)
- Maybe better: Count combinations of words  $\Rightarrow$  Which words co-occur together? (Chapter 9, not obligatory)

We *don't* specify what to look for in advance

## Top-down

- Count frequencies of pre-defined words (like in BOW-sentiment analysis)
- Maybe better: patterns instead of words (**regular expressions, today!**)

We *do* specify what to look for in advance

# Regular Expressions: What and why?

## What is a regexp?

- a *very* widespread way to describe patterns in strings

# Regular Expressions: What and why?

## What is a regexp?

- a *very* widespread way to describe patterns in strings
- Think of wildcards like `*` or operators like OR, AND or NOT in search strings: a regexp does the same, but is *much* more powerful

# Regular Expressions: What and why?

## What is a regexp?

- a *very* widespread way to describe patterns in strings
- Think of wildcards like `*` or operators like OR, AND or NOT in search strings: a regexp does the same, but is *much* more powerful
- You can use them in many editors (!), in the Terminal, in STATA ... and in Python



# An example

## From last week's task

- We wanted to remove everything but words from a tweet
- We did so by calling the `.replace()` method
- We could do this with a regular expression as well:  
    `[^a-zA-Z]` would match anything that is not a letter

# Basic regexp elements

## Alternatives

`[TtFf]` matches either T or t or F or f

`Twitter|Facebook` matches either Twitter or Facebook

`.` matches any character

# Basic regexp elements

## Alternatives

`[TtFf]` matches either T or t or F or f

`Twitter|Facebook` matches either Twitter or Facebook

`.` matches any character

## Repetition

`*` the expression before occurs 0 or more times

`+` the expression before occurs 1 or more times

# regexp quizz

Which words would be matched?

① [Pp]ython

# regexp quizz

Which words would be matched?

- ① [Pp]ython
- ② [A-Z] +

# regexp quizz

Which words would be matched?

- ❶ [Pp]ython
- ❷ [A-Z]+
- ❸ RT ? : ? @ [a-zA-Z0-9]\*

## What is a regexp?

# What else is possible?

If you google regexp or regular expression, you'll get a bunch of useful overviews. The wikipedia page is not too bad, either.

# How to use regular expressions in Python

## The module re

`re.findall("[Tt]witter|[Ff]acebook", testo)` returns a list with all occurrences of Twitter or Facebook in the string called `testo`

`re.findall("[0-9]+[a-zA-Z]+", testo)` returns a list with all words that start with one or more numbers followed by one or more letters in the string called `testo`



# How to use regular expressions in Python

## The module `re`

`re.findall("[Tt]witter|[Ff]acebook", testo)` returns a list with all occurrences of Twitter or Facebook in the string called `testo`

`re.findall("[0-9]+[a-zA-Z]+", testo)` returns a list with all words that start with one or more numbers followed by one or more letters in the string called `testo`

`re.sub("[Tt]witter|[Ff]acebook", "a social medium", testo)` returns a string in which all occurrences of Twitter or Facebook are replaced by "a social medium"

# How to use regular expressions in Python

## The module re

`re.match(" +([0-9]+) of ([0-9]+) points",line)` returns `None` unless it *exactly* matches the string `line`. If it does, you can access the part between `()` with the `.group()` method.

### Example:

```
1 line="                2 of 25 points"
2 result=re.match(" +([0-9]+) of ([0-9]+) points",line)
3 if result:
4     print ("Your points:",result.group(1))
5     print ("Maximum points:",result.group(2))
```

Your points: 2

Maximum points: 25

# Possible applications

## Data preprocessing

- Remove unwanted characters, words, ...
- Identify *meaningful* bits of text: usernames, headlines, where an article starts, ...
- filter (distinguish relevant from irrelevant cases)

# Possible applications

## Data analysis: Automated coding

- Actors
- Brands
- links or other markers that follow a regular pattern
- Numbers (!)

## Example 1: Counting actors

```
1 import re, csv
2 from glob import glob
3 count1_list=[]
4 count2_list=[]
5 filename_list = glob("/home/damian/articles/*.txt")
6
7 for fn in filename_list:
8     with open(fn) as fi:
9         artikel = fi.read()
10        artikel = artikel.replace('\n',' ')
11
12        count1 = len(re.findall('Israel.*(minister|politician.*|[Aa]
13                               uthorit)',artikel))
14
15        count2 = len(re.findall('[Pp]alest',artikel))
16
17        count1_list.append(count1)
18        count2_list.append(count2)
19
20 output=zip(filename_list,count1_list, count2_list)
21 with open("results.csv", mode='w',encoding="utf-8") as fo:
22     writer = csv.writer(fo)
23     writer.writerows(output)
```

## Example 2: Which number has this Lexis Nexis article?

```
1           All Rights Reserved
2
3           2 of 200 DOCUMENTS
4
5           De Telegraaf
6
7           21 maart 2014 vrijdag
8
9 Brussel bereikt akkoord aanpak probleembanken;
10 ECB krijgt meer in melk te brokkelen
11
12 SECTION: Finance; Blz. 24
13 LENGTH: 660 woorden
14
15 BRUSSEL Europa heeft gisteren op de valreep een akkoord bereikt
16 over een saneringsfonds voor banken. Daarmee staat de laatste
```

## Example 2: Check the number of a lexis nexis article

```
1           All Rights Reserved
2
3           2 of 200 DOCUMENTS
4
5           De Telegraaf
6
7           21 maart 2014 vrijdag
8
9 Brussel bereikt akkoord aanpak probleembanken;
10 ECB krijgt meer in melk te brokkelen
11
12 SECTION: Finance; Blz. 24
13 LENGTH: 660 woorden
14
15 BRUSSEL Europa heeft gisteren op de valreep een akkoord bereikt
16 over een saneringsfonds voor banken. Daarmee staat de laatste
```

```
1 for line in tekst:
2     matchObj=re.match(r" +([0-9]+) of ([0-9]+) DOCUMENTS",line)
3     if matchObj:
4         numberofarticle= int(matchObj.group(1))
5         totalnumberofarticles= int(matchObj.group(2))
```

# Practice yourself!

`http://www.pyregex.com/`



# Natural Language Processing

# NLP: What and why?

## What can we do?

- remove stopwords

# NLP: What and why?

## What can we do?

- remove stopwords
- stemming

# NLP: What and why?

## What can we do?

- remove stopwords
- stemming
- parse sentences (advanced)

# Natural Language Processing:

## Stopword removal

## Natural Language Processing: **Stopword removal**

*Have a look back at last week! The logic of the algorithm is very much related to the one of our first simple sentiment analysis!*

# Stopword removal: What and why?

## Why remove stopwords?

- If we want to identify key terms (e.g., by means of a word count), we are not interested in them
- If we want to calculate document similarity, it might be inflated
- If we want to make a word co-occurrence graph, irrelevant information will dominate the picture

# Stopword removal: How

```
1 testo='He gives her a beer and a cigarette.'  
2 testonuovo=""  
3 stopwords=['and','the','a','or','he','she','him','her']  
4 for verbo in testo.split():  
5     if verbo not in stopwords:  
6         testonuovo=testonuovo+verbo+" "
```

What do we get if we do:

```
1 print (testonuovo)
```

Can you explain the algorithm?



# We get:

```
1 >>> print (testonuovo)
2 'He gives beer cigarettes. '
```

Why is "He" still in there?  
How can we fix this?

# Stopword removal

```
1 testo='He gives her a beer and a cigarette.'  
2 testonuovo=""  
3 stopwords=['and','the','a','or','he','she','him','her']  
4 for verbo in testo.split():  
5     if verbo.lower() not in stopwords:  
6         testonuovo=testonuovo+verbo+" "
```

# NLP: What and why?

## Why do stemming?

- Because we do not want to distinguish between smoke, smoked, smoking, . . .
- Typical preprocessing step (like stopwords removal)

# Stemming

(with NLTK, see Bird, S., Loper, E., & Klein, E. (2009). *Natural language processing with Python*. Sebastopol, CA: O'Reilly.)

```
1 from nltk.stem.snowball import SnowballStemmer
2 stemmer=SnowballStemmer("english")
3 frase="I am running while generously greeting my neighbors"
4 frasenuevo=""
5 for palabra in frase.split():
6     frasenuevo=frasenuevo + stemmer.stem(palabra) + " "
```

If we now did `print(frasenuevo)`, it would return:

```
1 i am run while generous greet my neighbor
```

# Stemming and stopwords removal - let's combine them!

```
1 from nltk.stem.snowball import SnowballStemmer
2 from nltk.corpus import stopwords
3 stemmer=SnowballStemmer("english")
4 stopwords = stopwords.words("english")
5 frase="I am running while generously greeting my neighbors"
6 frasenuevo=""
7 for palabra in frase.lower().split():
8     if palabra not in stopwords:
9         frasenuevo=frasenuevo + stemmer.stem(palabra) + " "
```

Now, `print(frasenuevo)` returns:

```
1 run generous greet neighbor
```

Perfect!

# Stemming and stopwords removal - let's combine them!

```
1 from nltk.stem.snowball import SnowballStemmer
2 from nltk.corpus import stopwords
3 stemmer=SnowballStemmer("english")
4 stopwords = stopwords.words("english")
5 frase="I am running while generously greeting my neighbors"
6 frasenuevo=""
7 for palabra in frase.lower().split():
8     if palabra not in stopwords:
9         frasenuevo=frasenuevo + stemmer.stem(palabra) + " "
```

Now, `print(frasenuevo)` returns:

```
1 run generous greet neighbor
```

## Perfect!

In order to use `nltk.corpus.stopwords`, you have to download that module once. You can do so by typing the following in the Python console and selecting the appropriate package from the menu that pops up:

```
import nltk
nltk.download()
```

NB: Don't download everything, that's several GB.

## NLTK Downloader

File View Sort Help

Collections Corpora Models All Packages

| Identifier         | Name  | Size     | Status     |
|--------------------|---|----------|------------|
| senseval           | SENSEVAL 2 Corpus: Sense Tagged Text        | 2.1 MB   | not instal |
| sentiwordnet       | SentiWordNet                                | 4.5 MB   | not instal |
| shakespeare        | Shakespeare XML Corpus Sample               | 464.3 KB | not instal |
| sinica_treebank    | Sinica Treebank Corpus Sample               | 878.2 KB | not instal |
| smultron           | SMULTRON Corpus Sample                      | 162.3 KB | not instal |
| state_union        | C-Span State of the Union Address Corpus    | 789.8 KB | not instal |
| stopwords          | Stopwords Corpus                            | 8.5 KB   | not instal |
| swadesh            | Swadesh Wordlists                           | 22.3 KB  | not instal |
| switchboard        | Switchboard Corpus Sample                   | 772.6 KB | not instal |
| timit              | TIMIT Corpus Sample                         | 21.2 MB  | not instal |
| toolbox            | Toolbox Sample Files                        | 244.7 KB | not instal |
| treebank           | Penn Treebank Sample                        | 1.6 MB   | not instal |
| udhr               | Universal Declaration of Human Rights Corpu | 1.1 MB   | not instal |
| udhr2              | Universal Declaration of Human Rights Corpu | 1.6 MB   | not instal |
| unicode_samples    | Unicode Samples                             | 1.2 KB   | not instal |
| universal_treebank | Universal Treebanks Version 2.0             | 24.7 MB  | not instal |

Download

Refresh

Server Index: [http://nltk.github.com/nltk\\_data/](http://nltk.github.com/nltk_data/)Download Directory: [/home/damian/nltk\\_data](/home/damian/nltk_data)

In [5]: import nltk

In [6]: nltk.download()

# NLP: What and why?

## Why parse sentences?

- To find out what grammatical function words have
- and to get closer to the meaning.



# Parsing a sentence

```
1 import nltk
2 sentence = "At eight o'clock on Thursday morning, Arthur didn't feel
   very good."
3 tokens = nltk.word_tokenize(sentence)
4 print (tokens)
```

`nltk.word_tokenize(sentence)` is similar to `sentence.split()`, but compare handling of punctuation and the `didn't` in the output:

```
1 ['At', 'eight', "o'clock", 'on', 'Thursday', 'morning', 'Arthur', 'did',
   "n't", 'feel', 'very', 'good', '.']
```

# Parsing a sentence

Now, as the next step, you can “tag” the tokenized sentence:

```
1 tagged = nltk.pos_tag(tokens)
2 print (tagged[0:6])
```

gives you the following:

```
1 [('At', 'IN'), ('eight', 'CD'), ("o'clock", 'JJ'), ('on', 'IN'),
2  ('Thursday', 'NNP'), ('morning', 'NN')]
```

# Parsing a sentence

Now, as the next step, you can “tag” the tokenized sentence:

```
1 tagged = nltk.pos_tag(tokens)
2 print (tagged[0:6])
```

gives you the following:

```
1 [('At', 'IN'), ('eight', 'CD'), ("o'clock", 'JJ'), ('on', 'IN'),
2  ('Thursday', 'NNP'), ('morning', 'NN')]
```

And you could get the word type of "morning" with  
tagged[5][1]!

## More NLP

Look at <http://nltk.org>

Take-home message  
Take-home exam  
Next meetings

# Take-home messages

## What you should be familiar with:

- Possible steps to preprocess the data
- Regular expressions
- Word counts

# Next meetings

## TAKE HOME EXAM

Deadline: Tuesday, 23.59

## Wednesday

Statistics with pandas and Jupyter Notebook