Big Data and Automated Content Analysis

Week 5 – Monday

»Automated content analysis with NLP and regular expressions«

Damian Trilling

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

Afdeling Communicatiewetenschap
Universiteit van Amsterdam

5 March 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?

1. `[Pp]ython`
2. `[A-Z]+`
3. `RT ?:?  @[a-zA-Z0-9]*`

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 occurances 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 occurances 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 all occurances 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]
             uthorit)',artikel))
13       count2 = len(re.findall('[Pp]alest',artikel))
14
15       count1_list.append(count1)
16       count2_list.append(count2)
17
18 output=zip(filename_list,count1_list, count2_list)
19 with open("results.csv", mode='w',encoding="utf-8") as fo:
20     writer = csv.writer(fo)
21     writer.writerows(output)
```

Example 2: Which number has this Lexis Nexis article?

```
                                All Rights Reserved

                                 2 of 200 DOCUMENTS

                                    De Telegraaf

                                 21 maart 2014 vrijdag

Brussel bereikt akkoord aanpak probleembanken;
ECB krijgt meer in melk te brokkelen

SECTION: Finance; Blz. 24
LENGTH: 660 woorden

BRUSSEL  Europa heeft gisteren op de valreep een akkoord bereikt
over een saneringsfonds voor banken. Daarmee staat de laatste
```

Example 2: Check the number of a lexis nexis article

```
                                All Rights Reserved

                                 2 of 200 DOCUMENTS

                                    De Telegraaf

                                 21 maart 2014 vrijdag

Brussel bereikt akkoord aanpak probleembanken;
ECB krijgt meer in melk te brokkelen

SECTION: Finance; Blz. 24
LENGTH: 660 woorden

BRUSSEL  Europa heeft gisteren op de valreep een akkoord bereikt
over een saneringsfonds voor banken. Daarmee staat de laatste
```

```
for line in tekst:
    matchObj=re.match(r" +([0-9]+) of ([0-9]+) DOCUMENTS",line)
    if matchObj:
        numberofarticle= int(matchObj.group(1))
        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-occurance graph, irrelevant information will dominate the picture

Stopword removal: How

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

What do we get if we do:

```
print (testonuovo)
```

Can you explain the algorithm?

We get:

```
>>> print (testonuovo)
'He gives beer cigarette. '
```

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 stopword 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 stopword removal - let's combine them!

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

Now, `print(frasenuevo)` returns:

```
run generous greet neighbor
```

Perfect!

Stemming and stopword removal - let's combine them!

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

Now, `print(frasenuevo)` returns:

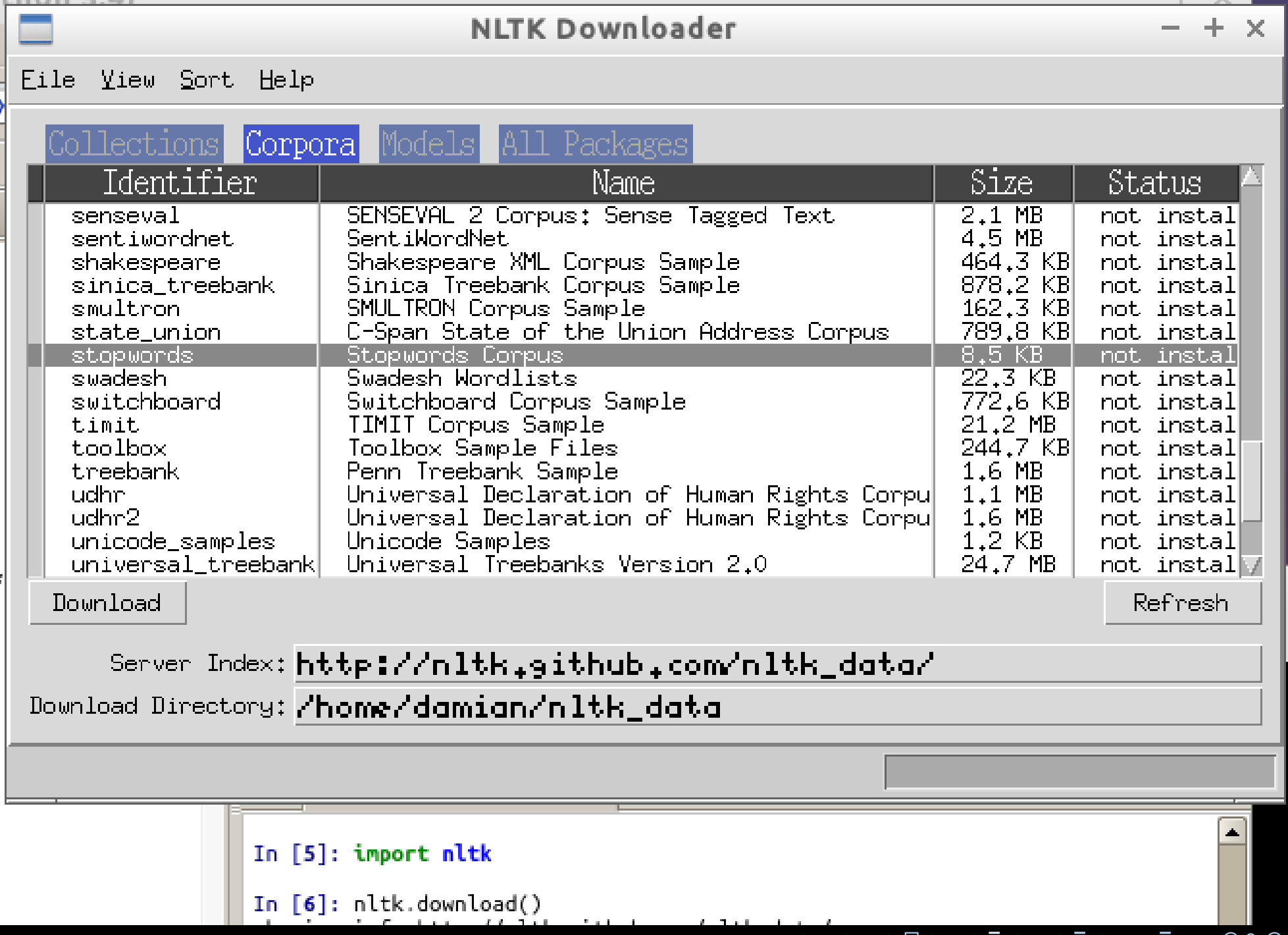
```
run generous greet neighbor
```

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.




```
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:

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

gives you the following:

```
[('At', 'IN'), ('eight', 'CD'), ("o'clock", 'JJ'), ('on', 'IN'),
('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

Wednesday

Statistics with pandas and Jupyter Notebook
Take-home exam