

# Teach-the-teacher: Python

## Day 4: » Advanced NLP & Regular Expressions «

---

Anne Kroon

a.c.kroon@uva.nl

December 4, 2023

UvA RM Communication Science

# Today

Advanced NLP

Parsing sentences

ACA using regular expressions

What is a regexp?

Using a regexp in Python

# Advanced NLP

---

# Advanced NLP

---

Parsing sentences

# 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 using NLTK

Tokenize a sentence, and “tag” the tokenized sentence:

```
1 tokens = nltk.word_tokenize(sentence)
2 tagged = nltk.pos_tag(tokens)
3 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]!

# Parsing a sentence using NLTK

Tokenize a sentence, and “tag” the tokenized sentence:

```
1 tokens = nltk.word_tokenize(sentence)
2 tagged = nltk.pos_tag(tokens)
3 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]`!

# Named Entity Recognition with spacy

## Terminal:

```
1 sudo pip3 install spacy
2 sudo python3 -m spacy download nl # or en, de, fr ....
```

## Python:

```
1 import spacy
2 nlp = spacy.load('nl')
3 doc = nlp('Een 38-jarige vrouw uit Zeist en twee mannen moeten 24
    maanden de cel in voor de gecoördineerde oplichting van Rabobank-
    klanten.')
4 for ent in doc.ents:
5     print(ent.text, ent.label_)
```

## returns:

```
1 Zeist LOC
2 Rabobank ORG
```



# More NLP

<http://nlp.stanford.edu> <http://spacy.io> <http://nltk.org>

<https://www.clips.uantwerpen.be/pattern>

# Main takeaway

- Preprocessing matters, be able to make informed choices.
- Keep this in mind when moving to Machine Learning.

# Regular expressions

---

## Automated content analysis using regular expressions

# Regular expressions

---

What is a regexp?

# 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

# 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

# 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

## Regex example

- Let's say we wanted to remove everything but words from a tweet
- We could do 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

### Repetition

`*` the expression before occurs 0 or more times

`+` the expression before occurs 1 or more times

## 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

## regex quiz

Which words would be matched?

1. [Pp]ython

2. [A-Z]+

3. RT ? : ? @ [a-zA-Z0-9]\*

## regex quiz

Which words would be matched?

1. [Pp]ython

2. [A-Z]+

3. RT ? : ? @ [a-zA-Z0-9]\*

# regex quizz

Which words would be matched?

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

## What else is possible?

See the table in the book!

# Regular expressions

---

Using a regexp in Python



# 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"

Use the less-known but more powerful module `regex` instead to support all dialects used in the book

# 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"

Use the less-known but more powerful module `regex` instead to support all dialects used in the book

# 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 (f"Your points: {result.group(1)}, 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)
13        ',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 probleebanken;
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 probleebanken;

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     numberofarticles= int(matchObj.group(1))
```



## Practice yourself!

Let's take some time to write some regular expressions. Write a script that

- extracts URLs from a list of strings
- removes everything that is not a letter or number from a list of strings

(first develop it for a single string, then scale up)

More tips: <http://www.pyregex.com/>