A Practical Introduction to Machine Learning in Python Day 2 - Tuesday Afternoon »From text to features: Advanced NLP and Regular Expression «

Rupert Kiddle
Marieke van Hoof

r.t.kiddle@vu.nl, @rptkiddle
m.vanhoof@uva.nl, @marieke_vh

September 17, 2024

Today

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:

```
tokens = nltk.word_tokenize(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')]
```

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:

```
tokens = nltk.word_tokenize(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')]
```

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

Named Entity Recognition with spacy

Terminal:

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

Python:

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

returns:

```
Zeist LOC
Rabobank ORG
```

Why POS-tag or NER?

- As part of your preprocessing pipeline (e.g., filter nouns)
- Enrich your feature set (e.g., number of people in the text carries a lot of information?)

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

regexp quizz

Which words would be matched?

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

regexp quizz

Which words would be matched?

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

regexp 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 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"

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 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"

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:

```
line="            2 of 25 points"
result=re.match(" +([0-9]+) of ([0-9]+) points",line)
if result:
    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

```python
import re, csv
from glob import glob
count1_list=[]
count2_list=[]
filename_list = glob("/home/damian/articles/*.txt")

for fn in filename_list:
    with open(fn) as fi:
        artikel = fi.read()
        artikel = artikel.replace('\n',' ')

    count1 = len(re.findall('Israel.*(minister|politician.*|[Aa]uthorit)
        ',artikel))
    count2 = len(re.findall('[Pp]alest',artikel))

    count1_list.append(count1)
    count2_list.append(count2)

output=zip(filename_list,count1_list, count2_list)
with open("results.csv", mode='w',encoding="utf-8") as fo:
    writer = csv.writer(fo)
    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))
```

Practice yourself!

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

- extracts URLS form 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/