Advanced tokenization with NLTK and regex

LaTeX, Puteaux, 2020, 2021

```
##############################################
##                                          ##
##  Natural Language Processing in Python   ##
##                                          ##
##############################################
```

```
§1 Introduction to Natural Language Processing in Python
```

```
§1.1 Regular expressions & word tokenization
```

1 Advanced tokenization with NLTK and regex

1.1 How to make regex groups and how to indicate "OR"?

- "OR" is represented using `|`.
- It is possible to define a group using `()`.
- It is possible to define explicit character ranges using `[]`.

1.2 Code of regex groups and the indication of "OR":

[1]:
```
import re

match_digits_and_words = ('(\d+|\w+)')
re.findall(match_digits_and_words, 'He has 11 cats.')
```

[1]:
```
['He', 'has', '11', 'cats']
```

1.3 What are regex ranges and groups?

pattern	matches	example
[A-Za-z]+	upper and lowercase English alphabet	'ABCDEFghijk'
[0-9]	numbers from 0 to 9	9
[A-Za-z\-\.]+	upper and lowercase English alphabet, - and .	'My-Website.com'
(a-z)	a, - and z	'a-z'
(\s+|,)	spaces or a comma	', '

1.4 Code of character range with `re.match()`:

[2]:
```
my_str = 'match lowercase spaces nums like 12, but no commas'
re.match('[a-z0-9 ]+', my_str)
```

[2]:
```
<re.Match object; span=(0, 35), match='match lowercase spaces nums like 12'>
```

1.5 Practice question for choosing a tokenizer:

- Given the following string, which of the below patterns is the best tokenizer? It is better to retain sentence punctuation as separate tokens if possible but have `'#1'` remain a single token.

  `my_string = "SOLDIER #1: Found them? In Mercea? The coconut's tropical!"`

  ☐ `r"(\w+|\?|!)"`.

  ☒ `r"(\w+|#\d|\?|!)"`.

  ☐ `r"(#\d\w+\?!)"`.

  ☐ `r"\s+"`.

▶ Package pre-loading:

[3]:
```
from nltk.tokenize import regexp_tokenize
```

▶ Data pre-loading:

[4]:
```
my_string = "SOLDIER #1: Found them? In Mercea? The coconut's tropical!"
string = my_string

pattern1 = r"(\w+|\?|!)"
pattern2 = r"(\w+|#\d|\?|!)"
pattern3 = r"(#\d\w+\?!)"
pattern4 = r"\s+"
```

▶ Question-solving method:

[5]:
```
pattern = pattern1
print(regexp_tokenize(string, pattern))
```

```
['SOLDIER', '1', 'Found', 'them', '?', 'In', 'Mercea', '?', 'The', 'coconut',
's', 'tropical', '!']
```

[6]:
```
pattern = pattern2
print(regexp_tokenize(string, pattern))
```

```
['SOLDIER', '#1', 'Found', 'them', '?', 'In', 'Mercea', '?', 'The', 'coconut',
's', 'tropical', '!']
```

[7]:
```
pattern = pattern3
print(regexp_tokenize(string, pattern))
```

```
[]
```

[8]:
```
pattern = pattern4
print(regexp_tokenize(string, pattern))
```

```
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
```

1.6 Practice exercises for advanced tokenization with NLTK and regex:

▶ **Data pre-loading:**

[9]:
```
tweets = [
    'This is the best #nlp exercise ive found online! #python',
    '#NLP is super fun! <3 #learning', 'Thanks @datacamp :) #nlp #python'
]
```

▶ **NLTK regex tokenization practice:**

[10]:
```
# Import the necessary modules
from nltk.tokenize import TweetTokenizer
from nltk.tokenize import regexp_tokenize
```

[11]:
```
# Import the necessary modules
from nltk.tokenize import regexp_tokenize
from nltk.tokenize import TweetTokenizer
# Define a regex pattern to find hashtags: pattern1
pattern1 = r"#\w+"
# Use the pattern on the first tweet in the tweets list
hashtags = regexp_tokenize(tweets[0], pattern1)
print(hashtags)
```

```
['#nlp', '#python']
```

[12]:
```
# Import the necessary modules
from nltk.tokenize import regexp_tokenize
from nltk.tokenize import TweetTokenizer
# Write a pattern that matches both mentions (@) and hashtags
pattern2 = r"([@#]\w+)"
# Use the pattern on the last tweet in the tweets list
mentions_hashtags = regexp_tokenize(tweets[-1], pattern2)
print(mentions_hashtags)
```

```
['@datacamp', '#nlp', '#python']
```

[13]:
```
# Import the necessary modules
from nltk.tokenize import regexp_tokenize
from nltk.tokenize import TweetTokenizer
# Use the TweetTokenizer to tokenize all tweets into one list
tknzr = TweetTokenizer()
all_tokens = [tknzr.tokenize(t) for t in tweets]
print(all_tokens)
```

```
[['This', 'is', 'the', 'best', '#nlp', 'exercise', 'ive', 'found', 'online',
'!', '#python'], ['#NLP', 'is', 'super', 'fun', '!', '<3', '#learning'],
['Thanks', '@datacamp', ':)', '#nlp', '#python']]
```

▶ Package pre-loading:

[14]:
```
from nltk.tokenize import word_tokenize
```

▶ Data re-pre-loading:

[15]:
```
german_text = 'Wann gehen wir Pizza essen?  Und fährst du mit Über? '
```

▶ Non-ASCII tokenization practice:

[16]:
```
# Tokenize and print all words in german_text
all_words = word_tokenize(german_text)
print(all_words)

# Tokenize and print only capital words
capital_words = r"[A-ZÜ]\w+"
print(regexp_tokenize(german_text, capital_words))

# Tokenize and print only emoji
emoji = "['\U0001F300-\U0001F5FF'|'\U0001F600-\U0001F64F'|\
'\U0001F680-\U0001F6FF'|'\u2600-\u26FF\u2700-\u27BF']"

print(regexp_tokenize(german_text, emoji))
```

```
['Wann', 'gehen', 'wir', 'Pizza', 'essen', '?', ' ', 'Und', 'fährst', 'du',
'mit', 'Über', '?', ' ']
```

```
['Wann', 'Pizza', 'Und', 'Über']
[' ', ' ']
```

1.7 Version checking:

[17]:
```
import sys
import nltk

print('The Python version is {}.'.format(sys.version.split()[0]))
print('The NLTK version is {}.'.format(nltk.__version__))
```

```
The Python version is 3.7.9.
The NLTK version is 3.5.
```