Introduction to tokenization

LaTeX, Puteaux, 2020, 2021

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

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

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

1 Introduction to tokenization

1.1 What is tokenization?

- It turns a string or document into tokens (smaller chunks).
- It's one step in preparing a text for NLP.
- It has many different theories and rules.
- Users can create their own rules using regular expressions.
- There are some examples:
  - *breaking out words or sentences*
  - *separating punctuation*
  - *separating all hashtags in a tweet*

1.2 What is the NLTK library?

- NLTK: Natural Language Toolkit

1.3 Code of the NLTK library:

[1]:
```
from nltk.tokenize import word_tokenize

word_tokenize("Hi there!")
```

[1]:
```
['Hi', 'there', '!']
```

1.4 Why tokenize?

- Easier to map part of speech.
- To match common words.
- To remove unwanted tokens.
- E.g.,

```
>>> word_tokenize("I don't like Sam's shoes.")
['I', 'do', "n't", 'like', 'Sam', "'s", 'shoes', '.']
```

1.5 What are the other NLTK tokenizers?

- `sent_tokenize`: tokenize a document into sentences.
- `regexp_tokenize`: tokenize a string or document based on a regular expression pattern.
- `TweetTokenizer`: special class just for tweet tokenization, allowing separate hashtags, mentions, and lots of exclamation points, such as '!!!'.

1.6 Code of regex practice (the difference between `re.search()` and `re.match()`):

[2]:
```
import re

re.match('abc', 'abcde')
```

[2]:
```
<re.Match object; span=(0, 3), match='abc'>
```

[3]:
```
re.search('abc', 'abcde')
```

[3]:
```
<re.Match object; span=(0, 3), match='abc'>
```

[4]:
```
re.match('cd', 'abcde')
```

[5]:
```
re.search('cd', 'abcde')
```

[5]:
```
<re.Match object; span=(2, 4), match='cd'>
```

1.7 Practice exercises for introduction to tokenization:

▶ Data pre-loading:

[6]:
```
scene_one = open("ref2. Scene 1 of Monty Python and the Holy Grail.txt").read()
```

▶ NLTK word tokenization with practice:

[7]:
```
# Import necessary modules
from nltk.tokenize import sent_tokenize
from nltk.tokenize import word_tokenize
```

```
# Split scene_one into sentences: sentences
sentences = sent_tokenize(scene_one)

# Use word_tokenize to tokenize the fourth sentence: tokenized_sent
tokenized_sent = word_tokenize(sentences[3])

# Make a set of unique tokens in the entire scene: unique_tokens
unique_tokens = set(word_tokenize(scene_one))

# Print the unique tokens result
print(unique_tokens)
```

```
{'right', 'Uther', 'Listen', 'master', 'to', 'since', 'King', 'Yes', 'them',
'order', "'em", 'why', '1', 'a', 'plover', 'strand', 'are', 'simple', 'will',
'kingdom', 'sovereign', 'martin', 'tell', "'ve", 'go', 'this', 'grips',
'creeper', 'wind', 'mean', 'KING', 'weight', 'it', 'south', 'ratios', 'bird',
'I', 'Ridden', 'bangin', '?', 'me', 'ask', 'Camelot', 'European', 'court',
'must', 'We', "'", 'needs', 'swallows', 'do', 'dorsal', 'No', 'African', 'our',
'from', 'goes', 'anyway', 'got', 'sun', 'ounce', 'he', 'if', 'husk', 'defeator',
'!', 'servant', 'maybe', 'Pendragon', 'suggesting', 'Please', 'times', 'you',
'But', 'join', 'A', 'castle', 'migrate', "'m", 'land', 'Wait', '…', 'Court',
'line', 'Arthur', 'house', 'yet', 'on', 'grip', 'five', '2', 'horse', 'may',
'carry', 'in', 'found', 'these', 'England', 'beat', 'an', 'wants', 'fly',
'Whoa', 'climes', 'that', 'where', ']', 'Are', 'tropical', 'You', 'coconut',
'Oh', 'air-speed', 'maintain', 'back', 'minute', 'SOLDIER', 'It', 'son', '.',
'lord', 'So', 'they', 'yeah', 'am', 'velocity', 'Not', 'all', 'carrying',
'Where', 'empty', 'but', '#', 'bring', 'swallow', 'is', "'d", 'snows', 'Saxons',
':', 'Well', 'be', 'and', 'using', 'or', 'by', 'Supposing', 'have', 'with',
'the', 'seek', 'knights', 'strangers', 'Halt', 'coconuts', 'Am', 'could',
'forty-three', 'speak', 'course', 'just', 'Found', 'does', 'ridden', 'SCENE',
',', 'your', 'Britons', 'get', 'Who', 'pound', 'other', 'ARTHUR', 'halves',
'held', 'its', 'wings', 'second', 'search', 'trusty', 'breadth', 'question',
'every', 'one', 'winter', 'who', 'Pull', "'re", 'covered', "n't", 'point',
'clop', 'The', 'In', 'length', 'of', 'carried', '[', 'under', 'temperate',
'That', '--', 'What', 'non-migratory', 'guiding', 'Will', 'Mercea', 'my',
'agree', 'here', 'then', 'feathers', 'They', 'through', 'there', 'Patsy',
'together', 'warmer', 'not', 'two', 'zone', 'interested', 'at', "'s", 'matter',
'use'}
```

▶ Package pre-loading:

[8]:
```
import re
```

▶ Regex (`re.search()`) practice:

[9]:
```
# Search for the first occurrence of "coconuts" in scene_one: match
match = re.search("coconuts", scene_one)
```

```
# Print the start and end indexes of match
print(match.start(), match.end())
```

```
580 588
```

[10]:
```
# Write a regular expression to search for anything in square brackets: pattern1
pattern1 = r"\[.*\]"

# Use re.search to find the first text in square brackets
print(re.search(pattern1, scene_one))
```

```
<re.Match object; span=(9, 32), match='[wind] [clop clop clop]'>
```

[11]:
```
# Find the script notation at the beginning of the fourth sentence and print it
pattern2 = r"[\w\s#]+:"
print(re.match(pattern2, sentences[3]))
```

```
<re.Match object; span=(0, 7), match='ARTHUR:'>
```

1.8 Version checking:

[12]:
```
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.
```