in4080_2022_Ex_2_solution

September 4, 2022

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import nltk
from nltk.corpus import brown
```

[2]:
```
##################################
##
##  Exercise set 2, IN4080, 2019
##
##################################
```

[3]:
```
print("\nExercise 1 a")
print('******************')

with open("crisis.txt", 'r') as f:
    raw = f.read()

crisis_tokens = nltk.word_tokenize(raw)
print("There are {} tokens and {}  different types in the text.".format(
       len(crisis_tokens), len(set(crisis_tokens)))
      )
```

```
Exercise 1 a
******************
There are 1093 tokens and 448  different types in the text.
```

[4]:
```
print("\nExercise 1b")
print('******************')

words = []
numbers = []
puncts = []
others = []

for t in crisis_tokens:
    if t.isalpha(): words.append(t)
    elif t.isnumeric(): numbers.append(t)
```

```
    elif t in ".,;:?!'-()" or t in ['"', "''", "``"]: puncts.append(t)
    else: others.append(t)

print("Number of words: ", len(words))
print("Number of numbers: ", len(numbers))
print("Number of punctuation signs: ", len(puncts))
print("Number of other tokens: ", len(others))
```

```
Exercise 1b
******************
Number of words:  961
Number of numbers:  4
Number of punctuation signs:  110
Number of other tokens:  18
```

0.0.1 Exercise 1c

We inspect the non-word classes

[5]:
```
set(puncts)
```

[5]:
```
{"''", '(', ')', ',', '.', ':', '``'}
```

[6]:
```
set(others)
```

[6]:
```
{"'d",
 "'s",
 '1.4',
 '50/50',
 'climate-friendly',
 'hold-out',
 'long-time',
 "n't",
 'so-called'}
```

[7]:
```
numbers
```

[7]:
```
['66', '300', '250', '2013']
```

We see that there are some forms we will include by the words.

[8]:
```
import re

print("\nExercise 1c")
print('******************')

words = []
```

```
numbers = []
puncts = []
others = []

for t in crisis_tokens:
    if t.isalpha(): words.append(t)
    elif re.search('\w-\w',t): words.append(t)
    elif t.isnumeric(): numbers.append(t)
    elif t in ".,;:?!'-()" or t in ['"', "'", "``"]: puncts.append(t)
    else: others.append(t)

print("Number of words: ", len(words))
print("Number of numbers: ", len(numbers))
print("Number of punctuation signs: ", len(puncts))
print("Number of other tokens: ", len(others))
```

```
Exercise 1c
******************
Number of words:  965
Number of numbers:  4
Number of punctuation signs:  110
Number of other tokens:  14
```

[9]:
```
set(others)
```

[9]:
```
{"'d", "'s", '1.4', '50/50', "n't"}
```

0.1 Exercise 2

[10]:
```
print("\nExercise 2a")
print('******************')

sents = nltk.sent_tokenize(raw)
print("There are {} many sentences.".format(len(sents)))
```

```
Exercise 2a
******************
There are 37 many sentences.
```

0.1.1 Exercise 2b and 2d

[11]:
```
print("\nExercise 2b and d")
print('******************')

tokenized = [nltk.word_tokenize(s) for s in sents]
```

```
tokens = [[w for w in s ] for s in tokenized]
print("The number of tags before cleaning {}".format(sum(len(s) for s in␣
 ↪tokens)))

cleaned = [[w for w in s if not(w in ".,;:?!'-" or w in '"')]
                      for s in tokenized]

cleaned2 = [[w for w in s if not(w in ".,;:?!'-" or w  in ['"', "'", "``"])]
                      for s in tokenized]

number_of_words = sum(len(s) for s in cleaned)
ave_length = number_of_words/len(cleaned)

print("The number of tokens which are not punctuation marks {}".
 ↪format(number_of_words))
print("The number of tokens which are not punctuation marks, take 2 {}".
 ↪format(sum(len(s) for s in cleaned2)))
print("The average sentence length is {}".format(round(ave_length, 3)))
```

```
Exercise 2b and d
******************
The number of tags before cleaning 1093
The number of tokens which are not punctuation marks 1013
The number of tokens which are not punctuation marks, take 2 991
The average sentence length is 27.378
```

0.1.2 Exercise 4

[12]:
```
print("\nExercise 4a")
print('******************')

uni_tag_words = [x for x in brown.tagged_words(tagset='universal')]
# x is a pair of word and tag
uni_tag_freq = nltk.FreqDist([t for w,t in uni_tag_words])
# count the number of occurrences of each tag.
for t in uni_tag_freq:
    print("{:7}{:10}".format(t, uni_tag_freq[t]))
```

```
Exercise 4a
******************
NOUN       275558
VERB       182750
.          147565
ADP        144766
```

```
DET         137019
ADJ          83721
ADV          56239
PRON         49334
CONJ         38151
PRT          29829
NUM          14874
X             1386
```

[13]:
```python
print("\nExercise 4b")
print('******************')

uni_distr = nltk.ConditionalFreqDist(uni_tag_words)
# For each word this gives a FreqDist over the tags assigned to the word.

number_of_tags = {w : len(uni_distr[w]) for w in uni_distr}
# A FreqDist which to each word gives its number of different tags.

freq_freqs = nltk.FreqDist([number_of_tags[w] for w in number_of_tags])
# The frequency of frequensis of tags assiged to a word.

for numb in sorted(freq_freqs):
    print("{:10} words have {} different tags".format(freq_freqs[numb],numb))

m = max(freq_freqs)
print("\nThe following words occur with {} different tags:".format(m))
for w in number_of_tags:
    if number_of_tags[w] == m: print(w)
```

```
Exercise 4b
******************
     52461 words have 1 different tags
      3331 words have 2 different tags
       227 words have 3 different tags
        32 words have 4 different tags
         6 words have 5 different tags

The following words occur with 5 different tags:
that
to
well
down
round
damn
```