2. Tokenisation

Contents

- Introduction
- Word tokenisation
- Sentence tokenisation

Introduction

Tokenisation is the process of dividing a text into smaller linguistic units, such as the words, the paragraph, or the sentences. It is a crucial pre-processing operation for many text analysis tasks.

Word tokenisation generally takes place on the basis of the spaces that occur in between the words. The individual words that are found in a text are called "**tokens**". When this full list is deduplicated, leaving only the unique words, the items in such lists are called "**types**".

This tutorial explains how you can tokenise a text using the `nltk` package, a toolkit that enables you you work with texts in natural languages.

To make use of this package, you firstly need to import it.

```
import nltk
```

The methods that are discussed in this tutorial make use of a number of additional resources which are not installed by default. If you have never used `nltk` before, you need to run the code below to install these resources.

```
```

```
import ssl

try:
    _create_unverified_https_context = ssl._create_unverified_context
except AttributeError:
    pass
else:
    ssl._create_default_https_context = _create_unverified_https_context

nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('tagsets')
nltk.download('wordnet')
nltk.download('sentiwordnet')
```

Word tokenisation

The `ntlk.tokenize` module has a method named `word_tokenize()`. This methods demand a string as input. This may be a sentence or a whole paragraph. When it is run, the method returns a Python list containing all the individual words found in the string that is provided. As mentioned, the individual words are identified using the spaces found in this string.

```
from nltk.tokenize import word_tokenize

quote = '''
It was the best of times,
it was the worst of times'''

words = word_tokenize(quote)

print(words)
```

When you look closely at the output of the code above, you can see that the `word_tokenize()` method treats punctuation marks as separate tokens. In the example above, the comma following the first occurrence of the word ‘times’ is actually a separate item in the list.

The function `remove_punctuation`, defined below, can be used to remove the tokens that consist of punctuation only. As input, the function demands a list of tokens. For each string in this list, the function tests whether it consists of alphanumeric characters, using the `isalnum()` method. The function returns those words which pass this test only.

```

```

```
def remove_punctuation(words):
    new_list= []
    for w in words:
        if w.isalnum():
            new_list.append( w )
    return new_list
```

This function can be used as follows:

```
words = remove_punctuation(words)
print(words)
```

Once you have created a list containing all the individual tokens, you can easily count the total number of token, using the `len()` function.

```
nr_tokens = len( words )

print( f'The text fragment contains {nr_tokens} tokens.' )
```

Sentence tokenisation

The method `sent_tokenize()` from the `nltk` package can be used to divide a text into its separate sentences. This type of tokenisation take place on the basis of full stops and upper case characters.

The cell below contains an illustration of how the `sent_tokenize()` method can be used.

```
import nltk
from nltk.tokenize import sent_tokenize

## First paragraph of "A Farewell to Arms"
quote = '''In the late summer of that year we lived in a house in a village
that looked across the river and the plain to the mountains. In the bed of t
river there were pebbles and boulders, dry and white in the sun, and the wat
was clear and swiftly moving and blue in the channels. Troops went by the ho
and down the road and the dust they raised powdered the leaves of the trees.
trunks of the trees too were dusty and the leaves fell early that year and
we saw the troops marching along the road and the dust rising and leaves, st
by the breeze, falling and the soldiers marching and afterward the road bare
white except for the leaves.'''

sentences = sent_tokenize(quote)

print( f'The fragment contains { len(sentences) } sentences.\n' )

for s in sentences:
    print(s + '\n')
```

Exercise 2.1.

The 'Corpus' folder that you have downloaded as part of this tutorial contains the full text of 'Pride and Prejudice'. Write code that can help you to answer the following questions about this text:

- How many characters are there in the novel?
- How many words does the novel contain?
- How often does the novel mention the word "marriage"?
- How long are the words in *Pride and Prejudice* on average (measured in number of characters)?

```
import os
from os.path import join
from tdmh import *
from nltk import word_tokenize


path = join( 'Corpus' , 'PrideandPrejudice.txt')

fh = open( path , encoding = 'utf-8' )
full_text = fh.read()
```

Exercise 2.2.

Can you print the first five sentences of *Pride and Prejudice*? How many sentences does the novel contain in total? Building on the results of the previous exercise, can you calculate the average length of sentences? In other words, how many tokens do the sentences contain on average?

```
from nltk import sent_tokenize


path = join( 'Corpus' , 'PrideandPrejudice.txt')

fh = open( path , encoding = 'utf-8' )
full_text = fh.read()
```

Exercise 2.3.

What is the longest word in the novel *Ullyses*?

```
from nltk import word_tokenize

import os
path = os.path.join('Corpus','Ullyses.txt')

file = open(path,encoding='utf-8')
full_text = file.read()
```