Laboratorio 1

November 19, 2022

0.1 Laboratorio 1: NLTK y Contar Palabras Corpora

0.1.1 3: Usando NLTK

[1]:
```
from nltk.book import *
```

```
*** Introductory Examples for the NLTK Book ***
Loading text1, …, text9 and sent1, …, sent9
Type the name of the text or sentence to view it.
Type: 'texts()' or 'sents()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G . K . Chesterton 1908
```

La lista muestra 9 textos o corpora, algunos de ellos son novelas inglesas que quizás reconozcas. El corpus de artículos de Wall Street Journal, tiene un total de 1 millon de palabras de los diarios del Wall Streen Journal. Para listar las palabras de un texto en particular, puede usar los métodos de tokens. Por ejemplo, los primeros 5 tokens en el corpus de "Moby Dick" (text1) pueden ser obtenidos escribiendo:

[2]:
```
text1.tokens[0:5]
```

[2]:
```
['[', 'Moby', 'Dick', 'by', 'Herman']
```

Para hacer lo mismo en el Wall Street Journal corpus, tipeamos:

[3]:
```
text7.tokens[0:5]
```

[3]:
```
['Pierre', 'Vinken', ',', '61', 'years']
```

Notese que puede ver la lista de corpora otra vez, tipeando:

[4]:
```
texts()
```

```
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
```

```
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G . K . Chesterton 1908
```

También puede contar el número de ocurrencias de una palabra en particular, usando el método "count" y un parámetro con la palabra:

[5]:
```
text1.count("Moby")
```

[5]:
```
84
```

0.1.2 4: Strings y expresiones regulares

Antes de trabajar con un corpora grande, vamos a tomar una oración simple y mirar a los problemas que deberían tenerse en cuenta cuando procesamos strings. Trate de ejecutar lo siguiente:

[6]:
```
mysent = "The cat sat on the mat."
```

[7]:
```
from nltk import word_tokenize
```

[8]:
```
mysent_tokens = word_tokenize(mysent)
```

[9]:
```
mysent_tokens
```

[9]:
```
['The', 'cat', 'sat', 'on', 'the', 'mat', '.']
```

[10]:
```
len(mysent_tokens)
```

[10]:
```
7
```

word_tokenize es un método built-in que "tokeniza" una cadena de lenguaje natural. Notese que la tokenización no está basada solamente en separar por espacios. len(mysent_tokens) retorna el tamaño de la lista de tokens dado como argumento

A veces no queremos considerar las puntuaciones (punctuations) de la misma manera que los otros tokens (por ejemplo cuando queremos saber el total de palabras que hay en un texto).

Para eliminar signos de puntuación, puede usar expresiones regulares. Una expresión regular le permite hacer coincidir una cadena con un patrón. Por ejemplo, puede verificar si la cadena completa coincide con algunos caracteres iniciales y finales. Es decir, la cadena comienza con 'a' o 'd' y termina con una 'k'?

O puede encontrar si una parte de la cadena coincide con algún patrón. Un ejemplo de este segundo caso sería encontrar si una cadena 'x' es una subcadena de la cadena

'y'. Hay algunos patrones de cadena que no pueden especificarse mediante expresiones regulares. No entraremos en detalles aquí, pero puede leer más al respecto.

Una forma común de usar expresiones regulares en Python es a través del método re.search. El siguiente fragmento muestra cómo buscar si leng es una subcadena del texto procesamiento del lenguaje natural:

[11]:
```
from nltk import re
```

[12]:
```
m = re.search("leng", "Procesamiento del lenguaje natural")
```

[13]:
```
print(m)
```

```
<re.Match object; span=(18, 22), match='leng'>
```

[14]:
```
m.start()
```

[14]: 18

[15]:
```
m.end()
```

[15]: 22

El método re.search devuelve un objeto de coincidencia si hay una coincidencia exitosa. Al llamar al inicio, los métodos de finalización en el objeto de coincidencia devuelven el índice de inicio y finalización de la cadena de consulta en la cadena dada. Mira lo que pasa en este caso:

[16]:
```
n = re.search("procesa", "procesamiento del lenguaje natural")
```

[17]:
```
n
```

[17]: <re.Match object; span=(0, 7), match='procesa'>

Mientras que en los casos anteriores, el patrón era un valor de cadena específico, generalmente las expresiones regulares involucran rangos y caracteres comodín, pero también hay formas abreviadas incorporadas para las expresiones de uso frecuente. Ahora supongamos que consideramos como palabras, solo aquellos tokens que son alfanuméricos, es decir. solo contienen alfabetos, números o el hiper símbolos. La expresión regular que puede usar para este patrón es " w".

[18]:
```
mysent_tokens_nopunct = [word for word in word_tokenize(mysent)\
                         if re.search("\w", word)]
```

[19]:
```
mysent_tokens_nopunct
```

[19]: ['The', 'cat', 'sat', 'on', 'the', 'mat']

[20]:
```
len(mysent_tokens_nopunct)
```

[20]: 6

El método set crea un set. Un set por definición, solo tendrá objetos únicos (sin repetición). Por lo tanto, cada palabra aparecerá una vez. Intente:

[21]:
```
set(mysent_tokens_nopunct)
```

[21]: {'The', 'cat', 'mat', 'on', 'sat', 'the'}

[22]:
```
len(set(mysent_tokens_nopunct))
```

[22]: 6

El resultado se ve como lo que esperaba? (Pista: qué tamaño esperaba que tenga el set?)

set es una función case sensitive, por lo que considera a "the" y "The" como cadenas distintas y no elimina las repeticiones. Por lo tanto en este caso la longitud es la misma

Busque la lista de métodos para strings de Python y encuentre un método para solucionar este problema (https://docs.python.org/2/library/stdtypes.html#string-methods). (Pista: es más fácil aplicar el método antes de crear los tokens de mysent). Una vez creados los nuevos mysent_tokens, use el set otra vez para asegurarse de que está conforme con el resultado. Llamando al método len en este set, encontrará el número de palabras únicas (también conocido como "types").

Una posible solución es aplicar la función lower al string, que retorna una copia con todos los caracteres en minúsculas. De forma que al crear el set se eliminan todas las repeticiones.

[23]:
```
mysent_tokens = [word for word in word_tokenize(mysent.lower())\
                 if re.search("\w", word)]
```

[24]:
```
mysent_tokens
```

[24]: ['the', 'cat', 'sat', 'on', 'the', 'mat']

[25]:
```
set(mysent_tokens)
```

[25]: {'cat', 'mat', 'on', 'sat', 'the'}

[26]:
```
len(set(mysent_tokens))
```

[26]: 5

Ahora si, estamos listos para aplicar estas técnicas a un corpora más grande, de NLKT.

[27]:
```
moby_dick_tokens = text1.tokens
```

[28]:
```
moby_dick_tokens_nopunct = [word.lower() for word in moby_dick_tokens \
                            if re.search("\w", word)]
```

[29]:
```
moby_dick_tokens_nopunct[0:5]
```

[29]:
```
['moby', 'dick', 'by', 'herman', 'melville']
```

0.1.3 5: Probando sus conocimientos

Basado en lo que aprendió en las secciones previas, responda las siguientes preguntas:

1. ¿Cual es el numero de tokens en Moby Dick?

[30]:
```
len(moby_dick_tokens_nopunct)
```

[30]:
```
218621
```

En Moby Dick hay 218621 tokens

2. ¿Cuál es el número de types en Moby Dick?

[31]:
```
len(set(moby_dick_tokens_nopunct))
```

[31]:
```
17140
```

En Moby Dick hay 17140 types

3. Moby Dick type-token ratio

[32]:
```
len(set(moby_dick_tokens_nopunct)) / len(moby_dick_tokens_nopunct)
```

[32]:
```
0.07840051962071347
```

Moby Dick type-token ratio = 0.07840051962071347

4. WSJ type-token ratio

[33]:
```
wsj_tokens = text7.tokens
```

[34]:
```
wsj_tokens_nopunct = [word.lower() for word in wsj_tokens \
                      if re.search("\w", word)]
```

[35]:
```
len(set(wsj_tokens_nopunct)) / len(wsj_tokens_nopunct)
```

[35]:
```
0.129748424801388
```

WSJ type-ratio = 0.129748424801388

5. ¿Cuál de los dos tiene más diversidad léxica?

El texto de WSJ tiene mayor diversidad léxica

6. ¿Puede pensar una razón por la cual ese corpus es más diverso que el otro?

Un motivo podría ser que el corpus de WSJ toma noticias del diario Wall Street Journal, que posee artículos tratando una amplia variedad de temas. Mientras que Moby Dick es una novela de ficción centrada únicamente en la vida naval y la caza de ballenas, por lo que se usa un vocabulario más limitado a estos temas.

7. Cual es el "Maximum Likelikhood Estimate (MLE)" de la palabra "whale" (ballena) en Moby Dick?

[36]:
```
recuento_whale = moby_dick_tokens_nopunct.count("whale")
```

[37]:
```
recuento_whale
```

[37]: 1226

[38]:
```
len(moby_dick_tokens_nopunct)
```

[38]: 218621

$$P_{mobydick}("whale") = \frac{recuento("whale")}{totalcorpus} = \frac{1226}{218624} = 0.005607878474620462$$

El MLE de la palabra "whale" en Moby Dick es 0.005607878474620462

8. ¿Cuál es el MLE de "whale" en el corpus de WSJ?

[39]:
```
recuento_whale_wsj = wsj_tokens_nopunct.count("whale")
```

[40]:
```
recuento_whale_wsj
```

[40]: 0

$$P_{WSJ}("whale") = \frac{recuento("whale")}{totalcorpus} = \frac{0}{totalcorpus} = 0$$

El MLE de la palabra "whale" en WSJ es 0