



**FIME**

**PROCESAMIENTO DE  
LENGUAJE NATURAL 24**

**INVESTIGAR TODAS LAS  
FUNCIONES QUE TIENE NLTK**

**PROFESOR: OSWALDO  
CARILLO ZEPEDA**

**ALUMNA:CARMEN ITZEL  
CHAVEZ RODRIGUEZ**

**GRADO: 6TO Y GRUPO: D  
FECHA:18/04/ 2024**

## ¿Qué es NLTK?

El Natural Language Toolkit (NLTK) es una plataforma usada para construir programas para análisis de texto. La plataforma fue liberada originalmente para análisis de texto. La plataforma fue liberada originalmente por Steven Bird y Edward Loper en conjunto con un curso de lingüística computacional en la Universidad de Pennsylvania en 2001. Hay un libro de acompañamiento para la plataforma llamado Procesamiento de Lenguaje Natural con Python.

NATURAL  
LANGUAGE  
PROCESSING

USING



PYTHON

```
from nltk.stem import WordNetLemmatizer
nltk.download('wordnet')

# Inicializar el lematizador de Wordnet
lematizador = WordNetLemmatizer()

print(lematizador.lemmatize("bats"))
print(lematizador.lemmatize("feet"))

# Executed at 2023-07-25 12:34:42 in 2ms
```

bat  
foot

Nos gustaría buscar la palabra language. Podemos hacer esto simplemente usando la plataforma NLTK como sigue:

```
1 import nltk  
2  
3 file = open('NLTK.txt', 'r')  
4 read_file = file.read()  
5 text = nltk.Text(nltk.word_tokenize(read_file))  
6  
7 match = text.concordance('language')
```

reducir las palabras a su forma base o raíz, conocida como «lema». El lema es la forma más básica y genérica de una palabra como el infinitivo en el caso de los verbos.

```
from nltk.stem import WordNetLemmatizer  
nltk.download('wordnet')  
  
# Inicializar el lematizador de Wordnet  
lematizador = WordNetLemmatizer()  
  
print(lematizador.lemmatize("bats"))  
print(lematizador.lemmatize("feet"))  
Executed at 2023.07.25 13:34:42 in 2ms  
  
bat  
foot
```

reducir las palabras a su forma base o raíz, conocida como «lema». El lema es la forma más básica y genérica de una palabra como el infinitivo en el caso de los verbos.

Este proyecto recopila libros en formato electrónico (libros en dominio público, es decir, sin derechos de autor y por tanto legalmente descargables). Si queremos cargar este corpus y observar los archivos que contiene, podemos hacer en un intérprete interactivo

```
>>> from nltk.corpus import gutenberg  
>>> libros=gutenberg.fileids()  
>>> print(libros)  
['austen-emma.txt', 'austen-persuasion.txt', 'austen-sin.txt'  
'burgess-busterbrown.txt', 'carroll-alice.txt', 'chesterton-  
'edgeworth-parientes.txt', 'melville-moby_dick.txt', 'milton-  
'shakespeare-macbeth.txt', 'whitman-leaves.txt']
```

acceder a las palabras de un determinado libro de la colección haciendo lo siguiente:

```
>>> import nltk  
>>> mobi=nltk.corpus.gutenberg.words("melville-moby_dick.txt")  
>>> print(mobi)  
[['[', 'Moby', 'Dick', 'by', 'Herman', 'Melville', ...]  
>>> print(len(mobi))  
260819
```

es el PlaintextCorpusReader, que sirve, como indica su nombre, para leer corpus que estén en formato de texto plano (es decir, solo el texto, sin ningún tipo de anotación)

```
m nltk.corpus.reader.plaintext import PlaintextCorpusReader  
corpus = PlaintextCorpusReader(".", "mobi_dick.txt")  
  
oracion in corpus.sents():  
print(oracion)
```

Lo que sí que avanzamos ahora es que al PlaintextCorpusReader se le puede indicar qué tokenizador y segmentador tiene que utilizar.

```
import nltk
from nltk.corpus.reader.plaintext import PlaintextCorpusReader
from nltk.tokenize import RegexpTokenizer

segmentador= nltk.data.load("catalan.pickle")
tokenitzador=RegexpTokenizer('[ldsmLDSM]\''|\w+|[^\w\s]+')

corpus = PlaintextCorpusReader(".", 'DOGC-2015-cat.txt',word_tokenizer=tokenitzador,sent_tokenizer=segmentador)
for oracion in corpus.sents():
    print(oracion)
```

para que nos dé todas las palabras (o tokens) en vez de todas las oraciones y para que nos proporcione el número total de palabras

```
import nltk
from nltk.corpus.reader.plaintext import PlaintextCorpusReader
from nltk.tokenize import RegexpTokenizer

segmentador= nltk.data.load("catalan.pickle")
tokenitzador=RegexpTokenizer('[ldsmLDSM]\''|\w+|[^\w\s]+')

corpus = PlaintextCorpusReader(".", 'DOGC-2015-cat.txt',word_tokenizer=tokenitzador,sent_tokenizer=segmentador)
for paraula in corpus.words():
    print(palabra)
print("TOTAL PALABRAS:",len(corpus.words()))
```

Separa el texto por espacios en blanco, como lo hemos hecho en el ejemplo anterior. En este caso los espacios en blanco pueden ser los caracteres: espacio en blanco, tabulador y nueva línea

```
import nltk
oracion="El Sr. Martínez llegará mañana de Alicante con la R.E."
tokens=nltk.tokenize.WhitespaceTokenizer(). tokenize(oracion)
print(tokens)
```

el tokenizador por expresiones regulares considera que el que expresamos son los tokens, pero podemos definir el que son los separadores de tokens con el modificador **gaps=True**.

```
import re
from nltk.tokenize import RegexpTokenizer
oracion="El Sr. Martínez llegará mañana de Alicante con la R.E.N.F.
tokenizador=RegexpTokenizer('\s+',gaps=True)
tokens=tokenizador.tokenize(oracion)
print(tokens)
```

El proceso de segmentación también se puede hacer con `sent_tokenizer()`, que llama a una instancia especial del `PunktSentenceTokenizer` que ha sido entrenada y que funciona bastante bien para varias lenguas europeas.

```
from nltk.tokenize import sent_tokenize
parrafo1="Hoy hace un día muy bonito. Mañana Albert
parrafo2="El Sr. Martínez llegará mañana de Alicante.
El día siguiente visitará al Dr. Rovira en la Avda.
segmentos1=sent_tokenize(parrafo1)
print(segmentos1)
segmentos2=sent_tokenize(parrafo2)
print(segmentos2)
```