

Laboratorio 1: Grimms' Fairy Tales

Huerta Aguilar, Jesus., Huitzil Juárez, Guadalupe Quetzalli., Pérez Sánchez, Jasmine.,

Ruiz Ramírez, Gabino

Benemérita Universidad Autónoma de Puebla

Recuperación de la Información

Fecha: 23 enero de 2024

Resumen: En esta práctica se utilizó el lenguaje Python para crear un código que, a partir de un texto dado, separe el texto en tokens, elimine signos de puntuación, convierta el texto en minúsculas y elimine las palabras vacías, para ello hacemos uso de la librería nltk, así como código predefinido que ya trae la librería que permite este objetivo, como: split, string punctuation, re, stopwords, entre otros. El caso de re, permite tener más control para eliminar ciertos caracteres que por defecto no están.

Palabras clave: tokens, palabras vacías, re, nltk, Python.

I. INTRODUCCIÓN

Día a día la información va creciendo cada vez más, de tal forma que va siendo más difícil la búsqueda de esta para encontrar datos específicos. En esta práctica la recuperación de la información no solo se convierte en una herramienta esencial, sino también en un medio de aprendizaje que nos capacita para simplificar la información de manera efectiva, extrayendo así datos concretos y significativos. En el presente análisis, exploraremos la importancia de este proceso.

II. OBJETIVO Y PLANTEAMIENTO DEL PROBLEMA

Aprender a preparar los textos para que sean de utilidad en el proceso de recuperación de información. Para ello deberás separar el texto en tokens, eliminar los tokens inútiles (signos de puntuación, números, palabras vacías) y convertir a minúsculas.

III. DESARROLLO EXPERIMENTAL

Para realizar esta práctica necesitaremos:

- Software Python
- Librería NLTK
- Libro de la página: <https://www.gutenberg.org/files/2591/2591-0.txt>

Una vez instalado el programa, la librería y descargado el libro procedemos a iniciar con la práctica:

1. Del texto dado, separar las palabras usando el espacio como delimitador e imprimir el resultado de las primeras 100 palabras:
 - a. Para realizar el proceso primero importaremos las librerías, en seguida abriremos el archivo de texto, para poder hacer uso de él.

```
1 import re
2 import string
3 import nltk
4 import os
5 from nltk.corpus import stopwords
6
7 nltk.download('punkt')
8 nltk.download('stopwords')
9
10 # Lee el contenido del archivo externo
11 archivo_path = os.path.join(os.path.dirname(__file__), 'lab1.txt')
12 with open(archivo_path, 'r', encoding='utf-8') as archivo:
13     texto = archivo.read()
```

- b. Para separar las palabras usaremos una función que trae la librería nltk, ahora para delimitarlas por el espacio usaremos la función texto.split().

```
15 # Tokenización usando NLTK
16 palabras = nltk.word_tokenize(texto)
17
18 # separar por espacios
19 palabras = texto.split()
20
21 print("\n///// TEXTO SIN ESPACIOS\n")
22 print(palabras[:100])
```

2. Del resultado obtenido en el paso anterior, eliminar los signos de puntuación:
 - a. Primero realizaremos una investigación sobre el funcionamiento de re, para continuar con este paso.
 - b. Para poder eliminar los signos de puntuación usaremos re, y requerimos imprimir la función string.punctuation para observar que signos elimina, sin embargo esta función no permite eliminar el signo ‘

```
24 # mostrar signos de puntuacion
25 print("\n///// SIGNOS DE PUNTUACION\n")
26 print(string.punctuation)
27
```

- c. Para eliminar el signo faltante (‘) crearemos una variable en la que se encuentre este signo y la añadiremos a la función re para poder eliminarlo:

```
28 # Definir expresión regular para eliminar signos de puntuación
29 simbolos_extra = '‘'
30 re_punc = re.compile('%s%s' % (re.escape(string.punctuation), re.escape(simbolos_extra)))
31
32 # Eliminar signos de puntuación de cada palabra
33 stripped = [re_punc.sub('', w) for w in palabras]
34
35 print("\n///// TEXTO SIN SIGNOS DE PUNTUACION\n")
36 print(stripped[:100])
37
```

3. Realizar una investigación del funcionamiento de re y realizar un ejemplo poniendo en práctica esta función:
 - a. Para poder realizar otro ejemplo, escogimos como texto un cuento que viene en el libro dado en materiales, llamado "THE ROBBER BRIDEGROOM"
 - b. En este caso creamos otro código en el que la función re elimine las vocales del texto en las primeras 100 palabras:
4. En el texto del paso 1 y paso 2 a continuación colocaremos el texto en minúsculas de igual manera solo serán las primeras 100 palabras:
 - a. Primero investigaremos como funciona y para qué sirve lower().
 - b. Una vez que hayamos adquirido el conocimiento sobre la función 'lower', procederemos a utilizarla para obtener las primeras 100 palabras en minúsculas. Implementaremos un ciclo que, por cada palabra, la convertirá a minúsculas:

```

38 # Convertir a minusculas cada palabra de la lista
39 for i in range(len(stripped)):
40     stripped[i] = stripped[i].lower()
41
42
43 print("\n///// TEXTO EN MINUSCULAS\n")
44 print(stripped[:100])
45

```

5. Eliminar palabras vacías:
 - a. Importar palabras vacías en idioma ingles
 - b. "from nltk.corpus import stopwords"
 - c. stop_words = stopwords.words('english')"
 - d. Imprime todas las palabras vacías, revísalas, y escribe las primeras 5.

```

46 # Obtener lista de stopwords en ingles
47 stopwords_english = set(stopwords.words('english'))
48
49 # Convertir el conjunto de stopwords a una lista
50 stopwords_english_list = list(stopwords_english)
51
52 # Imprimir las primeras 5 palabras vacías
53 print("\n///// PRIMERAS 5 PALABRAS VACIAS\n")
54 print(stopwords_english_list[:5])
55

```

- e. Elimina palabras vacías y escribe nuevamente las 100 primeras palabras.

```

57 # Eliminar palabras vacías
58 filtered_words = [word for word in stripped if word not in stopwords_english]
59
60 # Imprime las primeras 100 palabras en minúsculas sin símbolos ni stopwords
61 print("\n///// TEXTO SIN PALABRAS VACIAS\n")
62 print(filtered_words[:100])
63

```

IV. DISCUSIÓN Y RESULTADOS

De acuerdo con los pasos de nuestra práctica nuestros resultados fueron los siguiente:

1. En la salida de consola se muestran todas las palabras separadas del 1 al 100, y divididas por el símbolo de espacio, también observamos que los símbolos de puntuación se añadieron junto con las palabras, por ejemplo: "Hola:" se convirtió en una sola palabra.



```
HE', 'COAL,', 'AND']
PS C:\Users\mini_OneDrive\Documents\Code Test> & C:/Users/mini_/Appdata/Local/Programs/Python/Python312/python.exe "c:/Users/mini_/OneDrive/Docu
ntos/Code Test/Test 1/1d1-2.py"
[nltk data] Downloading package punkt to
[nltk data]   C:\Users\mini_/Appdata\Roaming\nltk data...
[nltk data] Package punkt is already up-to-date!
['The', 'Project', 'Gutenberg', 'ebook', 'of', 'Grimms'', 'Fairy', 'Tales', 'by', 'Jacob', 'Grimm', 'and', 'Wilhelm', 'Grimm', 'This', 'ebook', 'is
', 'for', 'the', 'use', 'of', 'anyone', 'anywhere', 'in', 'the', 'United', 'States', 'and', 'most', 'other', 'parts', 'of', 'the', 'world', 'at', 'n
o', 'cost', 'and', 'with', 'almost', 'no', 'restrictions', 'whatsoever.', 'you', 'may', 'copy', 'it', 'give', 'it', 'away', 'or', 're-use', 'it', '
under', 'the', 'terms', 'of', 'the', 'Project', 'Gutenberg', 'license', 'Included', 'with', 'this', 'ebook', 'or', 'online', 'at', 'www.gutenberg.or
g.', 'If', 'you', 'are', 'not', 'located', 'in', 'the', 'United', 'States', 'you', 'will', 'have', 'to', 'check', 'the', 'laws', 'of', 'the', 'coun
try', 'where', 'you', 'are', 'located', 'before', 'using', 'this', 'ebook.', 'title:', 'Grimms'', 'Fairy', 'Tales']
PS C:\Users\mini_OneDrive\Documents\Code Test> []
```

Para la investigación acerca de re que solicita en los puntos 2 y 3 se obtuvo lo siguiente:

Resumen

Con re, podemos trabajar con expresiones regulares, que son patrones utilizados para buscar y manipular cadenas de texto. Algunas funciones clave del módulo "re" son: re.search(), re.match(), re.findall(), y re.sub(), que nos sirven para buscar patrones, encontrar todas las coincidencias, y realizar sustituciones en cadenas de texto, entre otras operaciones relacionadas con expresiones regulares.

Como se usa

- 1.- Importar el modulo

```
import re
```

- 2.- Funciones Principales:

re.search(pattern, string): Busca la primera ocurrencia del patrón en la cadena.

re.match(pattern, string): Busca el patrón solo al principio de la cadena.

re.findall(pattern, string): Encuentra todas las ocurrencias del patrón en la cadena.

re.sub(pattern, replacement, string): Reemplaza todas las ocurrencias del patrón con la cadena de reemplazo.

- 3.- Patrones

Los patrones en las expresiones regulares pueden incluir caracteres literales y meta caracteres para definir reglas más complejas. Por ejemplo, "\d" representa cualquier dígito, "\w" representa cualquier carácter alfanumérico, etc.

Ejemplo:

```
import re

texto = "La fecha es 21-01-2024."
patron = r"\d{2}-\d{2}-\d{4}" # Patrón para encontrar fechas en formato DD-MM-AAAA
resultado = re.search(patron, texto)

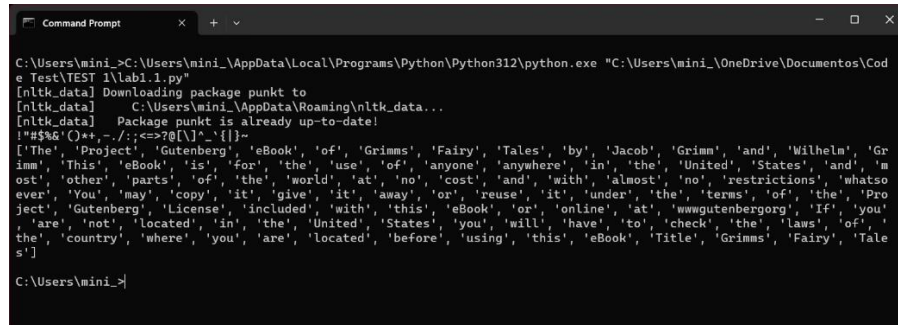
if resultado:
    print("Fecha encontrada:", resultado.group())
else:
    print("Fecha no encontrada.")
```

Resultado:

```
>>> ===== RESTART: C:\Users\PC\Downloads\asda.py =====
Fecha encontrada: 21-01-2024
>>>
```

2. El resultado del segundo paso obtuvimos lo siguiente:

b. Para poder observar el resultado el programa se tiene que ejecutar directamente desde la consola del nuestro ordenador (cmd) y una vez que logramos ejecutar el programa podemos observar que la primera impresión es mostrar los signos de puntuación que elimina por default re, en seguida vemos que realizó la eliminación completa de signos de puntuación que estaban junto a las palabras, sin afectarlas, de igual manera se mostrarán las palabras del 1 al 100.



```
C:\Users\mini>C:\Users\mini\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\mini\OneDrive\Documentos\Cod
e Test\TEST 1\lab1.1.py"
[nltk_data] Downloading package punkt to
[nltk_data]   C:\Users\mini\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
!""$%&'()*+,-./:;<=>@[\\]^_`{|}~
['The', 'Project', 'Gutenberg', 'eBook', 'of', 'Grimms', 'Fairy', 'Tales', 'by', 'Jacob', 'Grimm', 'and', 'Wilhelm', 'Gr
imm', 'This', 'eBook', 'is', 'for', 'the', 'use', 'of', 'anyone', 'anywhere', 'in', 'the', 'United', 'States', 'and', 'm
ost', 'other', 'parts', 'of', 'the', 'world', 'at', 'no', 'cost', 'and', 'with', 'almost', 'no', 'restrictions', 'whatso
ever', 'You', 'may', 'copy', 'it', 'give', 'it', 'away', 'or', 'reuse', 'it', 'under', 'the', 'terms', 'of', 'the', 'Pro
ject', 'Gutenberg', 'License', 'included', 'with', 'this', 'eBook', 'or', 'online', 'at', 'www.gutenberg.org', 'If', 'you'
, 'are', 'not', 'located', 'in', 'the', 'United', 'States', 'you', 'will', 'have', 'to', 'check', 'the', 'laws', 'of', '
the', 'country', 'where', 'you', 'are', 'located', 'before', 'using', 'this', 'eBook', 'Title', 'Grimms', 'Fairy', 'Tale
s']

C:\Users\mini>
```

3. En el paso tres se realizó un código aparte en el cual poníamos en práctica el funcionamiento de re:

b. Al guardar el texto "THE ROBBER BRIDEGROOM" en un archivo de texto llamado prueba, logramos eliminar correctamente las vocales de las primeras 100 palabras:

```

1 import re
2 import string
3 import nltk
4 import os
5
6 nltk.download('punkt') # descarga de tener el tokenizador punkt descargado
7
8 # Lee el contenido del archivo externo
9 archivo_path = os.path.join(os.path.dirname(__file__), 'prueba.txt')
10 with open(archivo_path, 'r', encoding='utf-8') as archivo:
11     texto = archivo.read()
12
13 # Tokenización usando NLTK
14 palabras = nltk.word_tokenize(texto)
15
16 texto = texto.lower()
17
18 palabras = texto.split()
19
20 print(string.punctuation)
21
22 # Definir expresiones regulares para eliminar signos de puntuación
23 simbolos_extra = '.,:;'
24 re_punc = re.compile(f'[{simbolos_extra}]')
25
26 # Eliminar signos de puntuación de cada palabra
27 strings = [re_punc.sub('', w) for w in palabras]
28
29 # Imprimir los primeros 100 palabras
30 print(strings[:100])
31
32
33

```

```

C:\Users\mini_>C:\Users\mini\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\mini\OneDrive\Documentos\Code Test\TEST 1\lab1.2.py"
[nltk_data] Downloading package punkt to
[nltk_data]   C:\Users\mini\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
['\\th', 'rbbr', 'brdgrm', 'thr', 'ws', 'nc', 'll', 'mllr', 'wh', 'hd', 'n', 'btfl', 'dgtr', 'nd', 's', 'sh', 'ws', 'grw',
n', 'p', 'h', 'ws', 'nxs', 'tth', 'sh', 'shld', 'b', 'wll', 'mrrd', 'nd', 'prvdd', 'fr', 'h', 'sd', 't', 'hmslf', 'll',
'wll', 'gv', 'hr', 't', 'th', 'frst', 'stbl', 'mn', 'wh', 'cms', 'nd', 'sks', 'fr', 'hr', 'hnd', 'nt', 'lng', 'ftr',
',', 'str', 'pprd', 'nd', 's', 'h', 'pprd', 't', 'b', 'vry', 'rch', 'nd', 'th', 'mllr', 'cld', 's', 'nthng', 'n', 'hm',
'wth', 'whch', 't', 'fnd', 'flt', 'h', 'btrhd', 'hs', 'dgtr', 't', 'hm', 'bt', 'th', 'grl', 'dd', 'nt', 'cr', 'fr',
'th', 'mn', 's', 'l', 'grl', 'ght', 't', 'cr', 'fr']
C:\Users\mini_>

```

4. Para el penúltimo paso conseguimos lo siguiente:

- a. En la investigación acerca de “lower()”:

El método lower() se utiliza para convertir todos los caracteres de una cadena a minúsculas. Cuando se aplica lower() a una cadena, esta retorna una nueva cadena donde todas las letras en mayúsculas se han convertido a minúsculas.

Ejemplo:

```

texto = "Hola Mundo"
texto_en_minusculas = texto.lower()

print(texto_en_minusculas)

```

Resultado:

```

===== RESTART: C:/Users/PC/Downloads/lower.py =====
hola mundo
>>>

```

- b. Logramos convertir las primeras 100 palabras del primer texto, en minúsculas, sin afectar lo que ya teníamos anteriormente, que es la separación de las palabras y eliminación de signos, además, para poder observar el cambio debimos ejecutar el programa de la misma manera que en el paso 2, a través de cmd:

```
Símbolo del sistema X + - □ X

///// TEXTO EN MINUSCULAS

['the', 'project', 'gutemberg', 'ebook', 'of', 'grimms', 'fairy', 'tales', 'by', 'jacob', 'grimm', 'and', 'wilhelm', 'grimm', 'this', 'ebook', 'is', 'for', 'the', 'use', 'of', 'anyone', 'anywhere', 'in', 'the', 'united', 'states', 'and', 'most', 'other', 'parts', 'of', 'the', 'world', 'at', 'no', 'cost', 'and', 'with', 'almost', 'no', 'restrictions', 'whatsoever', 'you', 'may', 'copy', 'it', 'give', 'it', 'away', 'or', 'reuse', 'it', 'under', 'the', 'terms', 'of', 'the', 'project', 'gutemberg', 'license', 'included', 'with', 'this', 'ebook', 'or', 'online', 'at', 'www.gutemberg.org', 'if', 'you', 'are', 'not', 'located', 'in', 'the', 'united', 'states', 'you', 'will', 'have', 'to', 'check', 'the', 'laws', 'of', 'the', 'country', 'where', 'you', 'are', 'located', 'before', 'using', 'this', 'ebook', 'title', 'grimms', 'fairy', 'tales']

C:\Users\Jesús Huerta Aguilar>
```

5. En este último paso los resultados se muestran a continuación:

- d. En el primer párrafo muestran los signos de puntuación a eliminar, debemos tomar en cuenta que los pasos 1,2,4,5 se realizaron en un mismo programa, en el segundo parrado se muestran las primeras 5 palabras vacías que se encuentran en el texto y serán eliminadas en las primeras 100 palabras del mismo texto.
- e. En el tercer párrafo se encuentra el texto ya sin las palabras vacías, observamos que, al eliminarlas, el programa escoge las siguientes palabras correspondientes de acuerdo con el texto para completar las primeras 100 palabras mostradas, sin embargo, estas nuevas palabras no se encuentran en minúsculas y contienen los signos de puntuación.

```
Símbolo del sistema X + - □ X

///// TEXTO SIN ESPACIOS

['The', 'Project', 'Gutemberg', 'eBook', 'of', 'Grimms', 'Fairy', 'Tales', 'by', 'Jacob', 'Grimm', 'and', 'Wilhelm', 'Grimm', 'This', 'eBook', 'is', 'for', 'the', 'use', 'of', 'anyone', 'anywhere', 'in', 'the', 'United', 'States', 'and', 'most', 'other', 'parts', 'of', 'the', 'world', 'at', 'no', 'cost', 'and', 'with', 'almost', 'no', 'restrictions', 'whatsoever', 'You', 'may', 'copy', 'it', 'give', 'it', 'away', 'or', 'reuse', 'it', 'under', 'the', 'terms', 'of', 'the', 'Project', 'Gutemberg', 'License', 'included', 'with', 'this', 'eBook', 'or', 'online', 'at', 'www.gutemberg.org', 'if', 'you', 'are', 'not', 'located', 'in', 'the', 'United', 'States', 'you', 'will', 'have', 'to', 'check', 'the', 'laws', 'of', 'the', 'country', 'where', 'you', 'are', 'located', 'before', 'using', 'this', 'eBook', 'Title:', 'Grimms', 'Fairy', 'Tales']

///// SIGNOS DE PUNTUACION

!#$%&'()*+,-./:;<=>?@[\]^_`{|}~

///// TEXTO SIN SIGNOS DE PUNTUACION

['The', 'Project', 'Gutemberg', 'eBook', 'of', 'Grimms', 'Fairy', 'Tales', 'by', 'Jacob', 'Grimm', 'and', 'Wilhelm', 'Grimm', 'This', 'eBook', 'is', 'for', 'the', 'use', 'of', 'anyone', 'anywhere', 'in', 'the', 'United', 'States', 'and', 'most', 'other', 'parts', 'of', 'the', 'world', 'a', 't', 'no', 'cost', 'and', 'with', 'almost', 'no', 'restrictions', 'whatsoever', 'You', 'may', 'copy', 'it', 'give', 'it', 'away', 'or', 'reuse', 'it', 'under', 'the', 'terms', 'of', 'the', 'Project', 'Gutemberg', 'License', 'included', 'with', 'this', 'eBook', 'or', 'online', 'at', 'www.gutemberg.org', 'if', 'you', 'are', 'not', 'located', 'in', 'the', 'United', 'States', 'you', 'will', 'have', 'to', 'check', 'the', 'laws', 'of', 'the', 'country', 'where', 'you', 'are', 'located', 'before', 'using', 'this', 'eBook', 'Title:', 'Grimms', 'Fairy', 'Tales']

///// TEXTO EN MINUSCULAS

['the', 'project', 'gutemberg', 'ebook', 'of', 'grimms', 'fairy', 'tales', 'by', 'jacob', 'grimm', 'and', 'wilhelm', 'grimm', 'this', 'ebook', 'is', 'for', 'the', 'use', 'of', 'anyone', 'anywhere', 'in', 'the', 'united', 'states', 'and', 'most', 'other', 'parts', 'of', 'the', 'world', 'a', 't', 'no', 'cost', 'and', 'with', 'almost', 'no', 'restrictions', 'whatsoever', 'you', 'may', 'copy', 'it', 'give', 'it', 'away', 'or', 'reuse', 'it', 'under', 'the', 'terms', 'of', 'the', 'project', 'gutemberg', 'license', 'included', 'with', 'this', 'ebook', 'or', 'online', 'at', 'www.gutemberg.org', 'if', 'you', 'are', 'not', 'located', 'in', 'the', 'united', 'states', 'you', 'will', 'have', 'to', 'check', 'the', 'laws', 'of', 'the', 'country', 'where', 'you', 'are', 'located', 'before', 'using', 'this', 'ebook', 'title', 'grimms', 'fairy', 'tales']

///// PRIMERAS 5 PALABRAS VACÍAS

['now', 'below', 'y', 'them', "she's"]

///// TEXTO SIN PALABRAS VACÍAS

['project', 'gutemberg', 'ebook', 'grimms', 'fairy', 'tales', 'jacob', 'grimm', 'wilhelm', 'grimm', 'ebook', 'use', 'anyone', 'anywhere', 'united', 'states', 'parts', 'world', 'cost', 'almost', 'restrictions', 'whatsoever', 'may', 'copy', 'give', 'away', 'reuse', 'terms', 'project', 'gutemberg', 'license', 'included', 'ebook', 'online', 'www.gutemberg.org', 'located', 'united', 'states', 'check', 'laws', 'country', 'located', 'using', 'ebook', 'title', 'grimms', 'fairy', 'tales', 'author', 'jacob', 'grimm', 'wilhelm', 'grimm', 'translators', 'edgar', 'taylor', 'marian', 'edwards', 'release', 'date', 'april', '2001', 'ebook', '2591', 'recently', 'updated', 'june', '28', '2021', 'language', 'english', 'character', 'set', 'encoding', 'utf8', 'produced', 'emma', 'dudding', 'john', 'bickers', 'dagny', 'david', 'widger', 'start', 'project', 'gutemberg', 'ebook', 'grimms', 'fairy', 'tales', 'grimms', 'jacob', 'grimm', 'wilhelm', 'grimm', 'preparers']

C:\Users\Jesús Huerta Aguilar>
```

V. CONCLUSIONES

A través de la práctica constante, hemos logrado adquirir un conocimiento más profundo sobre el manejo de diversas funciones relacionadas con el procesamiento de texto en Python. Entre estas funciones, destacan la habilidad para dividir el texto en tokens, suprimir la puntuación y transformar el texto a minúsculas. Este proceso nos ha permitido consolidar nuestra comprensión, brindándonos una base más robusta para la preparación y manipulación de texto con el objetivo de recuperar información, todo ello mediante el uso de Python y la biblioteca NLTK.

VI. BIBLIOGRAFÍA

1. Python Software Foundation. (2024). re - Operaciones con expresiones regulares. Recuperado de <https://docs.python.org/es/3/library/re.html>