Benemérita Universidad Autónoma de Puebla

Recuperación de la Información

Profesora: Maya Carrillo Ruiz

Práctica 4:

Vocabulario

Alumnos:

- Ramírez Varela Axel Daniel
- Bañuelos López Marco Antonio

Primavera 2021

Para esta práctica, del enlace: http://ir.dcs.gla.ac.uk/resources/test_collections/cisi/

Con la colección que fue anteriormente analizada, obtendremos el vocabulario (es decir todas las palabras diferentes) de esta misma colección. Como ya lo hicimos la práctica pasada, realizamos los 5 pasos para preprocesar el texto, y guardamos el análisis en un documento que denominamos “ResultadosdeEjecuacucion.txt”, el programa esta vez generará adicionalmente un archivo con las palabras del vocabulario y otro donde se registre la longitud del vocabulario.

Palabras Clave: Vocabulario, Diccionario

I. INTRODUCCIÓN

En las prácticas pasadas realizamos el preprocesamiento de un texto corto obteniendo sus primeras 100 palabras, para hasta ahora, realizar el análisis completo de una colección almacenada en un solo archivo de texto, que contendría un aproximado de 1460 documentos y 112 consultas; este archivo lo podemos encontrar en el repositorio que compartimos como “TodosLibros.txt”. Para la práctica ahora lo único que agregaremos será la organización de las palabras con la condición que la lista que obtengamos esta ordenada de forma alfabética y además que no se repitan palabras en nuestro “diccionario”, organizarla en un texto en blanco y además un archivo que nos proporcione el número de palabras que existen en nuestro diccionario.

II. OBJETIVO Y PLANTEAMIENTO DEL PROBLEMA

Actualmente, ya tenemos programados los pasos para realizar el preprocesamiento de la colección, uno de los objetivos de esta práctica es obtener un “diccionario”, refiriéndonos a la colección completa de palabras que son consideradas de valor ordenadas alfabéticamente, además de poder obtener un número exacto de estas palabras, todo este proceso nos sirve con el objetivo de obtener un tesauro por ejemplo, el cual nos permitiría representar conceptos, no solo datos, sino darle un significado a todo este conjunto de palabras por grupos, sin necesidad de intervención por parte de un usuario, sino únicamente con la ayuda del programa

III. DESARROLLO EXPERIMENTAL

La tarea primordial para el preprocesamiento de la colección, consiste en el análisis correcto del texto en sí; obviamente no es útil un análisis si no se realizan los pasos de separación de tokens, eliminar puntuación, etc.

Para el procesamiento de un texto tenemos que realizar 5 pasos

- Separar las palabras en tokens
- Eliminar signos de puntuación
- Convertir todas las palabras a minúsculas
- Eliminar palabras vacías
- Truncar las palabras

Tomemos en cuenta que, para realizar estas prácticas, programaremos en Python apoyándonos en el uso de la librería NLTK, con funciones que nos permiten realizar todas las tareas.

Para el primer paso, separaremos en tokens[1] usando la línea de código

```
words = text[y].split()
```

En el caso para eliminar los signos de puntuación importaremos primero “import re” y utilizamos la línea:

```
words= re.split(r'\W+', text[y])
```

la función re.split() nos permite separar y eliminar los signos de puntuación de las palabras que encuentre.

Para convertir las palabras a minúsculas[2] utilizaremos la línea:

```
words = [word.lower() for word in words]
```

De esta manera con la función lower(), nos permite que la variable words que contiene todo el texto, para todas las mayúsculas que encuentre las convierta a minúsculas.

El siguiente proceso que es eliminar las palabras vacías[3] que lo haceel programa con la línea:

```
nltk_stopwords = set(stopwords.words('english'))
```

De esta forma estamos especificando que solo eliminaremos palabras vacías en inglés.

Finalmente, para truncar palabras[4], se programaron las siguientes líneas

```
for x in range(1,101):
    ww= text_without_stopword[x]
    todo.append(ps.stem(ww))
print(todo[1:100])
```

para reducir las palabras de la colección palabra a su raíz o stem.

Ya que describimos como realiza el preprocesamiento de datos tenemos entonces que manejar el análisis para todo el texto, guardarlo en otro archivo, y por comprobación observar la pantalla

De la colección hacemos el análisis de todos los documentos juntos en un solo texto, donde guardamos el texto que inicie en ".W" y terminara cuando encontrase en una nueva línea un ".X" Para entonces solo tomar el texto a procesar y leer línea por línea para trabajarla uno por uno. Este proceso lo llevamos a cabo principalmente en las siguientes líneas:

```
if text[x] == ".W"+"\n":
    y = x+1
    words=[]
    while text[y] != ".X"+"\n":
```

Posteriormente del while realizará todo el análisis con los 5 pasos ya mencionados. Así al ejecutar, el programa realizará dos funciones, imprimirá en pantalla TODAS las oraciones del texto mostrando los 5 pasos antes mencionados en cada oración, y la otra es que en el archivo "ResultadosdeEjecuacion.txt" del archivo rar guardará los resultados finales del archivo de texto totalmente convertido, esto para una mejor lectura del texto, el archivo "TodosLibros.txt" no sufrirá ningún cambio.

Para explicar cómo se crea el diccionario y el archivo contador de palabras primero abriremos el archivo de ResultadosdeEjecuacion.txt para comenzar analizando ahora nuestra colección ya preprocesada y asignaremos su contenido a una variable 'text':

```
filename = "ResultadosdeEjecuacucion.txt"
file = open(filename,encoding="utf8")
text = file.read()
file.close()
```

El código que realizara diccionario es el siguiente, donde primero dividiremos las palabras del texto con la función ‘split’ a la variable ‘text’ y la asignaremos a otra variable llamada ‘textoprocesado’, esto con el objetivo de separar las cadenas de texto en palabras separadas para luego agruparlas en una sola cadena que le facilitará la tarea al programa; posteriormente realizaremos los cambios de ordenar las palabras y agruparlas, con la función set() [7] agruparemos las palas en una sola cadena, y con la función sorted() [8] tiene la tarea de ordenar todas las palabras que encuentre de forma alfabética de menor a mayor es decir comenzando con números para posteriormente listar primero las palabras con “a” y terminar con las palabras que comiencen con “z”

```
Textoprocesado = text.split()
Textoprocesado=set(Textoprocesado)
Textoprocesado=sorted(Textoprocesado)
```

Abriremos un archivo denominado “Diccionario” en formato .txt y guardaremos todas las pablaras ya organizadas y separadas por espacio en este archivo con las siguientes 4 lineas, la primera línea abre el archivo, la segunda guarda en una variable ‘Texto’ cada palabra separándola por espacios, en la tercera escribe el contenido de la variable ‘Texto’ ahora si en el documento ‘Diccionario.txt’ y el cuarto cierra el archivo guardando todas las palabras

```
file = open("Diccionario.txt", "w")
Texto = " ".join(Textoprocesado)
file.write(Texto)
file.close()
```

Las últimas líneas realizan el conteo de las palabras en el diccionario, la primera línea, con la función len() [9] realiza todo el trabajo al contar el número de elementos que existen en la variable ‘Textoprocesado’ ya antes mencionado, el resto de líneas abre el archivo donde podemos leer el número total de palabras, las escribe en el archivo y guarda, finalizando el programa y cumpliendo el objetivo de la práctica

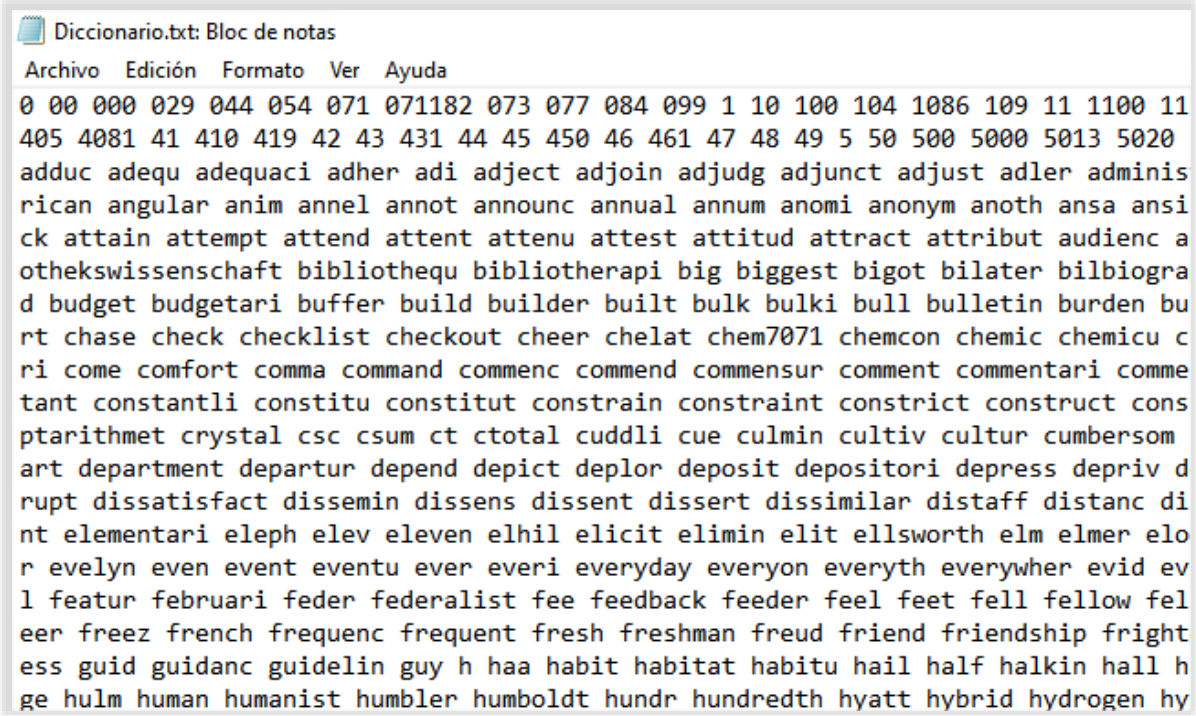
```
Longi = len(Textoprocesado)
file = open("LongitudDiccionario.txt", "w")
file.write(str(Longi))
```

```
        file.close()
```

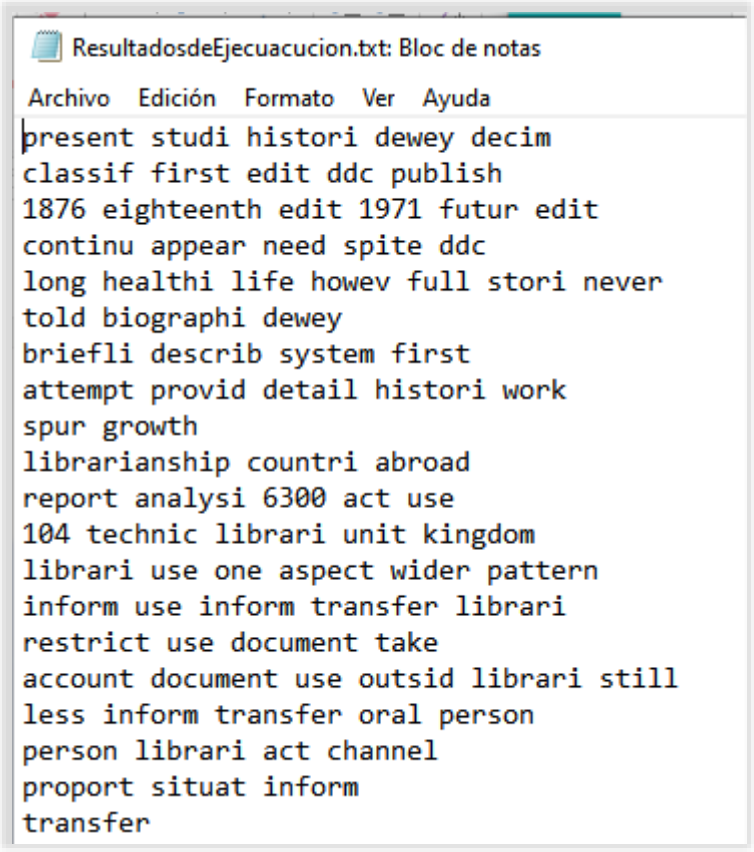
El código final junto con los dos documentos TodosLibros.txt y Q.txt han sido puestos en un repositorio de Github, la dirección para el repositorio se encuentra en la página del equipo: https://recinfoe3.blogspot.com/2021/03/practica-4-vocabulario.html

IV. DISCUSIÓN Y RESULTADOS

Como resultado principal tenemos los dos documentos donde podemos comprobar que estén organizados las palabras ya preprocesadas, como prueba tenemos la captura de pantalla con las primeras palabras organizadas



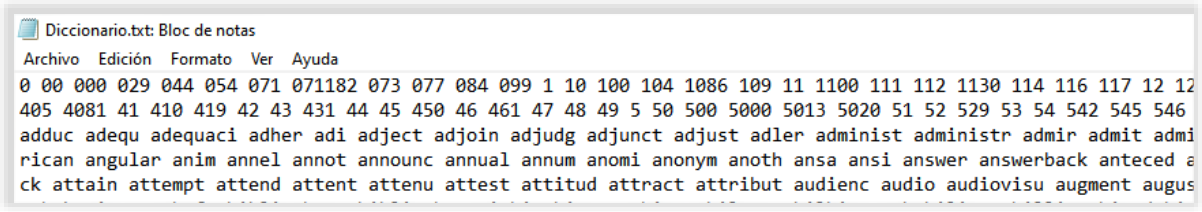
Como nota hay que señalar que el documento cuenta con más palabras de las primeras letras mostradas, pero para comodidad y rápida comprobación se muestra este extracto, una imagen con más palabras lo pueden encontrar en la página de la materia con el link antes mostrado. Posteriormente tenemos para comparación el texto con las palabras truncadas y preprocesadas en desorden, contra la impresión final del texto con los cambios realizados



Texto Preprocesado

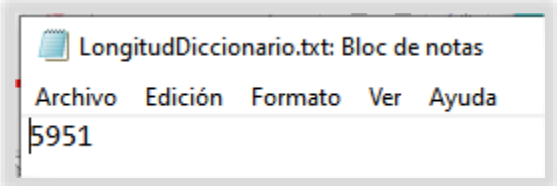
(Primeras palabras preprocesadas de la colección)

Texto Ordenado

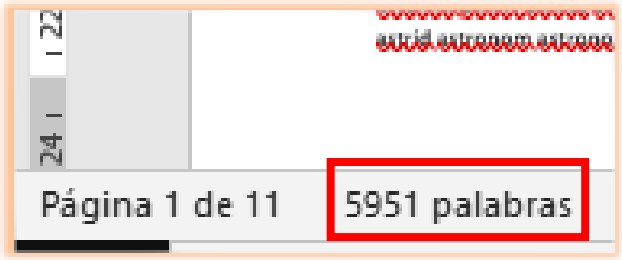


(Fila con las primeras palabras ordenadas)

Además, que el archivo “LongitudDiccionario.txt” nos proporciona el número correcto de palabras existentes



Para comprobar que el número fuera correcto, seleccionamos y copiamos las palabras del documento “Diccionario.txt” y las pegamos a un documento Word, el cual contabiliza automáticamente las palabras



Como podemos comprobar la función len() cumplió con el objetivo de contar las palabras totales y logramos de manera exitosa el objetivo de la práctica

V. CONCLUSIONES

Cumplimos entonces de manera satisfactoria los objetivos propuestos al inicio de la práctica tenemos en cuenta que podemos optimizar el programa por ejemplo para el diccionario podríamos agregar un contador y separar las palabras por renglones para mayor comodidad del usuario, pero no olvidemos que este es otro paso que nos llevaría para que analicemos de manera más compleja un grupo de datos, esto refiriéndonos a obtener conceptos proporcionando grupos de datos, y obteniendo por ejemplo el tema o género del documento a analizar o realizar un sistema más complejo recordemos que nosotros podemos darle un significado a los datos que

leamos, pero el programa y la computadora no pueden proporcionarlo por si solos, con la correcta codificación podemos lograr tareas más complejas.

VI. BIBLIOGRAFÍA

1. https://unipython.com/como-limpiar-el-texto-manualmente-usando-nltk/
2. https://likegeeks.com/es/tutorial-de-nlp-con-python-nltk/
3. https://ichi.pro/es/detenga-las-palabras-vacias-usando-diferentes-bibliotecas-de-python-281092180678227
4. https://pythonprogramming.net/stemming-nltk-tutorial/
5. https://medium.com/qu4nt/reducir-el-n%C3%BAmero-de-palabras-de-un-texto-lematizaci%C3%B3n-y-radicalizaci%C3%B3n-stemming-con-python-965bfd0c69fa
6. https://medium.com/@roquelopez/algoritmo-de-porter-para-el-espa%C3%B1ol-en-java-dd44ea7b0a10
7. https://www.w3schools.com/python/python_sets.asp
8. https://www.programiz.com/python-programming/methods/built-in/sorted
9. https://www.w3schools.com/python/ref_func_len.asp