

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 tokens¹ usando la línea de código

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

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

Para convertir las palabras a minúsculas² utilizaremos la línea:

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

El siguiente proceso que es eliminar las palabras vacías³ 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 palabras⁴, 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:

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))
```

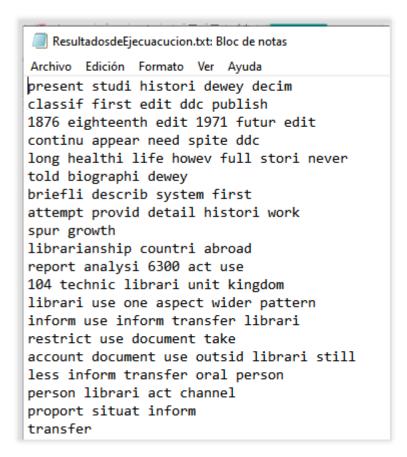
El código final junto con los dos documentos TodosLibros.txt y Q.txt han sido puestos en un repositorio de Github, la dirección para el repositorio se encuentra en la página del equipo: https://recinfoe3.blogspot.com/2021/03/practica-4-vocabulario.html

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

Diccionario.txt: Bloc de notas Archivo Edición Formato Ver Ayuda 0 00 000 029 044 054 071 071182 073 077 084 099 1 10 100 104 1086 109 11 1100 11 405 4081 41 410 419 42 43 431 44 45 450 46 461 47 48 49 5 50 500 5000 5013 5020 adduc adequ adequaci adher adi adject adjoin adjudg adjunct adjust adler adminis rican angular anim annel annot announc annual annum anomi anonym anoth ansa ansi ck attain attempt attend attent attenu attest attitud attract attribut audienc a othekswissenschaft bibliothequ bibliotherapi big biggest bigot bilater bilbiogra d budget budgetari buffer build builder built bulk bulki bull bulletin burden bu rt chase check checklist checkout cheer chelat chem7071 chemcon chemic chemicu c ri come comfort comma command commenc commend commensur comment commentari comme tant constantli constitu constitut constrain constraint constrict construct cons ptarithmet crystal csc csum ct ctotal cuddli cue culmin cultiv cultur cumbersom art department departur depend depict deplor deposit depositori depress depriv d rupt dissatisfact dissemin dissens dissent dissert dissimilar distaff distanc di nt elementari eleph elev eleven elhil elicit elimin elit ellsworth elm elmer elo r evelyn even event eventu ever everi everyday everyon everyth everywher evid ev l featur februari feder federalist fee feedback feeder feel feet fell fellow fel eer freez french frequenc frequent fresh freshman freud friend friendship fright ess guid guidanc guidelin guy h haa habit habitat habitu hail half halkin hall h ge hulm human humanist humbler humboldt hundr hundredth hyatt hybrid hydrogen hy

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

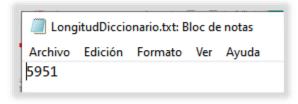
Diccionario.txt: Bloc de notas

Archivo Edición Formato Ver Ayuda

0 00 000 029 044 054 071 071182 073 077 084 099 1 10 100 104 1086 109 11 1100 111 112 1130 114 116 117 12 12 405 4081 41 410 419 42 43 431 44 45 450 46 461 47 48 49 5 50 500 5000 5013 5020 51 52 529 53 54 542 545 546 adduc adequ adequaci adher adi adject adjoin adjudg adjunct adjust adler administr admir admir rican angular anim annel annot announc annual annum anomi anonym anoth ansa ansi answer answerback anteced a ck attain attempt attend attent attenu attest attitud attract attribut audienc audio audiovisu augment augus

(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

0 00 000 029 044 054 071 071182 073 077 084 099 1 10 100 104 1086 109 11 1100 111 112 1130 114 116 117 12 120 1200 12000 124 125 12k 13 130 131 132 133 135 136 138 14 140 1400 1401 1403 1428 144 1463 1469 148 14th 15 150 1500 151 153 154 156 15th 16 160 1608 165 1665 167 169 17 170 1700 175 1750 176 1776 178 18 1836 1837 1847 185 1861 1866 1867 1868 1869 1871 1873 1874 1876 1877 188 1890 1893 1895 19 1900 1905 1907 191 1915 1916 1918 1919 1920 1921 1922 1927 1928 1929 1930 1931 1933 1935 1936 1937 1938 1939 194 1940 1941 1944 1947 1948 1950 1951 1952 1953 1954 1955 1956 1957 1958 1959 1960 1961 1961b 1962 1963 1964 1965 1966 1967 1967a 1968 1969 1970 1971 1972 1973 1974 1975 1976 1980 2 20 200 2000 204 2050 207 209 21 2104 212 215 22 220 225 2260 2267 227 23 24 240 241 242 245 246 24th 25 250 254 256 26 261 262 265 267 27 2740 2741 275 276 2788 28 281 285 286 290 298 2d 2x2 2x8 3 30 300 3000 3035 31 313 315 318 319 322 328 33 334 3400 342 343 349 35 350 357 36 360 370 38 380 39 3950 3rd 3x5 4 40 400 4020 403 405 4081 41 410 419 42 43 431 44 45 450 46 461 47 48 49 5 50 500 5000 5013 5020 51 52 529 53 54 542 545 546 55 56 568 57 570 57c 58 583 59 591 592 5x6 5x8 6 60 600 604 608 61 610 62 628 63 6300 64 646 647 66 6600 669204 67 676 68 69 6b 6c 7 70 704 7000 7004 74*C* 74030 73 733 73C 737 73 73 74 74C 74C 7C 7C 7C 7C 77 774 70 700 70 Página 1 de 11 - 5951 palabras - 🗐 - Español (Mexico)



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