Proyecto Final

PRE-PROCESAMIENTO Y CONSULTA DE TEXTO

Aaron Ramirez Martinez | Recuperacion de la información | 17/11/2021
Matricula: 201969648
Profesor: Arturo Olvera

Tabla de contenido

Introduccion

Actualmente la recuperacion de informacion es parte dde la vida digital cotidiana, pues al hablar de recuperacion de la informacion, hablamos acerca de como es que se genera una consulta en internte de medir el nivel de coincidencia en un conjunto de palabras, de como es que los buscadores mas populares del mundo trabajan, es por ello que quisimos estudiar su comportamiento y realizar una practica para poder observar de mejor manera el estudio de dichas actividades.

Objetivo de la practica

El objetivo de la practica que a continuacion se muestra es:

- Prepocesar un corpus con mas de 400 subcorpus
- Organizar el corpues y elimar *stop words* para un mejor manejo de la informacion.
- Se dara una consulta A en texto nomal y este tambien debe ser preprocesado.
- Realizar una consulta del texto A contra en corpus y determinar en que pocision se encuentran las coincidencias.
- Mostrar ambas partes preprocesadas.
- Al corpus se le deben extraer las palabras mas usadas y graficarlas.

Proyecto realizado.

A continuación, se muestra el código que se empleo en el desarrollo de la práctica, se muestra comentado y explica de manera breve el funcionamiento de cada línea de código:

```
# -*- coding: utf-8 -*-
"""
Created on Mon Aug 30 20:16:05 2021

@author: Aaron Ramirez
"""
#Abrimos el documento de txt y lo guardamos en una variable
with open('Corpus.txt','r',encoding= "utf8") as miarchivo:
    texto = miarchivo.read()

#Importamos las librerias requeridas.
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import string
import nltk

from collections import Counter
from collections import OrderedDict



#las cadenas a continuacion son las consultas que realizaremos
cadena='separation anxiety in infancy (i.e. up to two years of age) and in preschool children, particularly separation of a child from its mother'
#cadena='the toxicity of organic selenium compounds'
```

```
#cadena='language development in infancy and pre-school age'


#obtenemos las stop_words en el mismo lenguaje que el corpus

stop_words= set(stopwords.words('english'))

word_tokens = word_tokenize(texto) #tookenizar significa utilizar toda la palabra y no solo un caracter

word_tokens1 = word_tokenize(cadena)


##### PREPROCESAMIENTO DEL TEXTO #####


word_tokens = list(filter(lambda token : token not in string.punctuation,word_tokens,)) #Eliminamos caracteres de puntuacón del corpus

word_tokens1= list(filter(lambda token : token not in string.punctuation,word_tokens1)) #Eliminamos caracteres de puntuación de la consulta

filtro=[] #Declaramos una variable de tipo lista que contendrá el corpus una vez finalizado el preprocesamiento

filtro1=[]  #Declaramos una variable de tipo lista que contendrá la consulta una vez finalizado el preprocesamiento

aux=[]#Utilizamos una variable auxiliar para realizar la consulta


for palabra in word_tokens: #iniciamos el ciclo para eliminar stop words

    if palabra not in stop_words:

        filtro.append(palabra)


for i in word_tokens1:

    if i not in stop_words:

        filtro1.append(i)
```

```
##### CONSULTA DE TEXTO #####

for palabra1 in filtro1: #Recorremos la lista de consulta

   if (palabra1 in filtro): #Preguntamos si la palabra se encuentra en el corpus

      aux.append(filtro.index(palabra1)) # Si la consulta es verdadera obtenemos el
indice


##### Imprimimos en donde se encuentran las coincidencias #####


if(len(aux)==0): #si la lista de coincidencias esta vacia se regresa que no hubo
coincidencias

   print("Match not found")

else:

   for j in range(0,len(aux)):

      print("{} found on {}".format(filtro1[j],aux[j])) #Imprimimos Las coincidencias




c=Counter(filtro) # Obtenemos la propiedad contador de la libreria usada

fdist=nltk.FreqDist(filtro) # Usamos una funcion de la libreria para obtener la
frecuencia el

                        #la distribución del corpues preprocesado

fdist.plot(20,cumulative=True) #Graficamos los primeros 20 téminos más usuales


##### GUARDAMOS EL CORPUES PREPOSESADO #####

y=OrderedDict(c.most_common())


with open('salida.txt','w') as file:

   for k,v in y.items():

      file.write(f'{k} ' )
```

Resultados.

Usando la cadena *'separation anxiety in infancy (i.e. up to two years of age) and in preschool children, particularly separation of a child from its mother'* se obtienen las siguientes coincidencias:

```
In [4]: runcell(0, 'C:/Users/Aaron Ramirez/Desktop/Buap/otonio2021/
recuperacion de la informacion/proyecto/procesadorTexto.py')
separation found on 8683
anxiety found on 8975
infancy found on 9548
i.e found on 11876
two found on 48
years found on 5367
age found on 4030
preschool found on 20866
children found on 8976
particularly found on 1873
separation found on 8683
child found on 9357
mother found on 9317
```

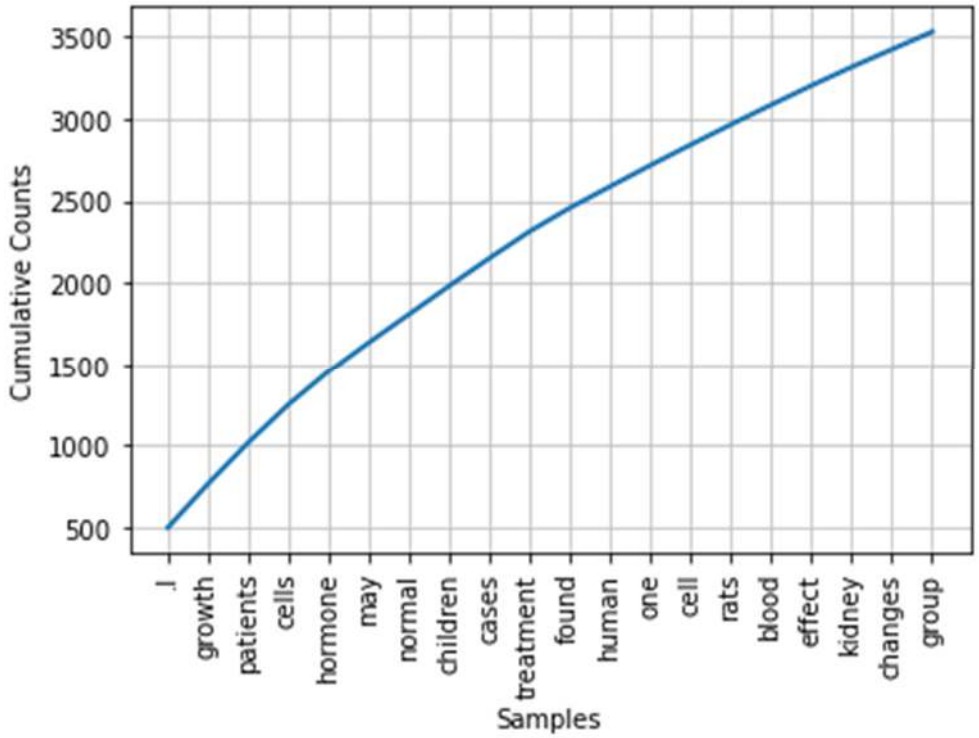
Usando la cadena *'the toxicity of organic selenium compounds'* se obtienen las siguientes coincidencias:

```
In [5]: runcell(0, 'C:/Users/Aaron Ramirez/Desktop/Buap/otonio2021/
recuperacion de la informacion/proyecto/procesadorTexto.py')
toxicity found on 12648
organic found on 7101
selenium found on 11855
compounds found on 12362
```

Y , finalmente usando la cadena *'language development in infancy and pre-school age'* obtenemos las siguientes coincidencias:

```
In [6]: runcell(0, 'C:/Users/Aaron Ramirez/Desktop/Buap/otonio2021/
recuperacion de la informacion/proyecto/procesadorTexto.py')
language found on 20069
development found on 89
infancy found on 9548
pre-school found on 4030
```

Representación grafica de las palabras más usadas.



Utilizando una función propia de la librería NLTK extraemos y graficamos los términos más comunes que se presentan en el corpus dado.

Experimentos.

Para el siguiente experimento utilizaremos una cadena con caracteres especiales dentro del texto para poder ver que el preprocesamiento se hace de manera correcta, los signos son los siguientes *!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~* obteniendo los siguiente:

```
In [9]: runcell(0, 'C:/Users/Aaron Ramirez/Desktop/Buap/otonio2021/
recuperacion de la informacion/proyecto/procesadorTexto.py')
Match not found
```

Ahora usaremos únicamente con las llamadas *stop words* usando la siguiente cadena *of not few so on where as no how d before shouldve has weren than will,* se obtiene los siguiente:

```
In [11]: runcell(0, 'C:/Users/Aaron Ramirez/Desktop/Buap/otonio2021/
recuperacion de la informacion/proyecto/procesadorTexto.py')
Match not found
```

Finalmente usemos una cadena de texto que no se encuentre en el corpus 'Perro gato' y observamos lo que sucede.

```
In [25]: runcell(0, 'C:/Users/Aaron Ramirez/Desktop/Buap/otonio2021/
recuperacion de la informacion/proyecto/procesadorTexto.py')
Match not found
```

No existe una coincidencia en ninguno de los casos.

De esta manera concluimos el proyecto presentando los avances y resultados esperados.

Referencias.

NLTK – dlegorreta

By Container: dlegorreta Publisher: dlegorreta Year: 2015 URL: https://dlegorreta.wordpress.com/tag/nltk/

Comprobar si la entrada es un número entero en Python

By Manav Narula Container: Delft Stack Year: 2021 URL: https://www.delftstack.com/es/howto/python/user-input-int-python/

Tutorial de NLP con Python NLTK (ejemplos simples)

By Mokhtar Ebrahim Container: Like Geeks Publisher: LikeGeeks Year: 2017 URL: https://likegeeks.com/es/tutorial-de-nlp-con-python-nltk/

Ejemplos de FingerPrint.wp_fp en Python, ejemplos de fingerprint.FingerPrint.wp_fp en Python - HotExamples

By Container: Hotexamples.com Year: 2021 URL: https://python.hotexamples.com/es/examples/fingerprint/FingerPrint/wp_fp/python-fingerprint-wp_fp-method-examples.html