Análisis Data

Braian Felipe Ramírez Ortiz
Fundación Universitaria Konrad Lorenz

I. Introducción

Dado el taller 3 y 4 de estructura de datos, se propone un texto para su análisis. Se pide desarrollar dos algoritmos: uno que recorra el documento reconociendo cada una de las palabras del mismo e identificando cuántas veces se repiten las palabras en él, imprimiendo como resultado una lista con cada palabra del documento mostrando la cantidad de veces que esta se encuentra dentro de este.

El segunda parte del algoritmo análiza el documento reconociendo la posición de cada una de las palabras dentro de este y retornando como resultado una nueva lista con las palabras del texto y su respectivos indices de ubicación.

II. Resultados

Para analizar el comportamiento del algoritmo se emplearon las bibliotecas time y psutil para estudiar la complejidad espacial y temporal, se ejecutó el algoritmo tres veces obteniendo varios datos de tiempo y memoria para asi generar un contraste de datos, se tienen encuentas las especificaciones tecnicas de la maquina en la cual se desarrollo el código.

Maquina	Sistema Operativo	Procesador	Memoria ram	Disco SSD
Lenovo V14 G2 ALC	Windows 11	Ryzen 7	40 GB	512GB

Figura 1.

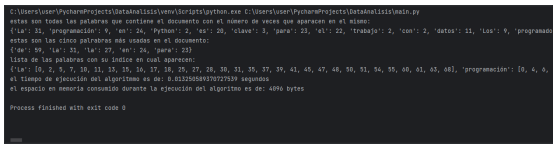

Figura 2. Ejecución 1

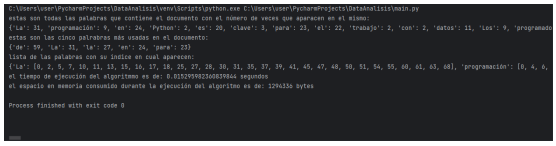

Figura 3. Ejecución 2

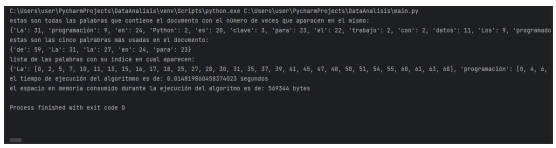

Figura 4. Ejecución 3

article

III. Código en Python

```python
import time
import psutil

def contar_palabras(lista):
    cache = []
    contador_palabras = {}

    for linea in lista:
        for palabra in linea.split():
            if palabra in cache:
                contador_palabras[palabra] +=
                    1
            else:
                cache.append(palabra)
                contador_palabras[palabra] = 1

    return contador_palabras

def palabras_mas_usadas(documento):
    cinco_mayores =
        dict(sorted(documento.items(), key =
        lambda item: item[1], reverse = True)
        [:5])
    return cinco_mayores

def buscar_palabra(documento):
    cache = []
    indice = {}
    for contador_linea, linea in
        enumerate(documento):
        for palabra in linea.split():
            if palabra in cache:
                indice[palabra].append(contador_linea)
            else:
                indice[palabra] =
                    [contador_linea]
                cache.append(palabra)
    return indice
```

En el análisis temporal se obtuvo como resultado un tiempo promedio de 0,014 segundos en la ejecución del algoritmo, mientras que en el análisis espacial se obtuvo 0,62 mega bytes en promedio de la memoria empleada por el algoritmo en su ejecución.

IV. Conclusión

Con la implementación de este tipo de algoritmos, destaca lo fácil que se vuelve el analizar textos en búsqueda de información relevante en un breve período de tiempo. Teniendo esto como punto de partida, podemos emplear estos algoritmos en el estudio de grandes bancos de datos en los cuales se guarda una gran cantidad de información. Si su análisis se realizara de forma manual, emplearía una gran cantidad de tiempo.

Con el análisis temporal y espacial, podemos encontrar formas para mejorar el rendimiento de nuestros algoritmos con

el fin de hacerlos más eficaces en su ejecución, consumiendo la menor cantidad de recursos posibles.