Taller 3

Yudy Carolina Guevara Cely
Fundación Universitaria Konrad Lorenz
yudyc.guevarac@konradlorenz.edu.co

***Index Terms*—Espacio, tiempo, diccionario.**

I. Introduction

Sabemos que memorización es implementad como parte de las estructuras de datos para poder analizar el como podemos en JavaScript modificar el comportamiento de una función para que sus resultados se cacheen.

II. Process

Importación de módulos

```
import time

import sys
```

Fig. 1. Representation of the mechanism induced by traps on the average drain current.

Resultados (imagen, descripción, explicación)

```
print ("desarrollo", buscar(" desarrollo ",my_documents ))




end_time =time.time()


print("el tiempo de ejecucio +{}".format(end_time - start_time))


size_my_documents_after = sys.getsizeof(my_documents)
size_diccionario_frecuencia_after = sys.getsizeof(diccionario_frecuencia)
memory_diff_my_documents = size_my_documents_after - size_my_documents
memory_diff_diccionario = size_diccionario_frecuencia_after - size_diccionario_frecuencia
print("Uso de memoria para my_documents antes del procesamiento:", size_my_documents, "bytes")
print("Uso de memoria para diccionario_frecuencia antes del procesamiento:", size_diccionario_frecuencia, "bytes")


print("Uso de memoria para my_documents después del procesamiento:", size_my_documents_after, "bytes")
print("Uso de memoria para diccionario_frecuencia después del procesamiento:", size_diccionario_frecuencia_after, "bytes")


print("Diferencia en uso de memoria para my_documents:", memory_diff_my_documents, "bytes")
print("Diferencia en uso de memoria para diccionario_frecuencia:", memory_diff_diccionario, "bytes")
```

Fig. 2. Representation of the mechanism induced by traps on the average drain current.

My documet contiene los documentos que vamos a entrar a analizar lo podemos notar cada caracteres entre documentos sería un documento que se encuentra contenido en My documents

```
my_documents = [

    "La programación en Python es clave para el trabajo con datos",

    "Los programadores en Java tienen un alto interés en pasar a Python",

    "La optimización de algoritmos es fundamental en el desarrollo de software",

    "Las bases de datos relacionales son esenciales para muchas aplicaciones",

    "El paradigma de programación funcional gana popularidad",

    "La seguridad informática es un tema crucial en el desarrollo de aplicaciones web",

    "Los lenguajes de programación modernos ofrecen abstracciones poderosas",

    "La inteligencia artificial está transformando diversas industrias",

    "El aprendizaje automático es una rama clave de la ciencia de datos",
```

Fig. 3. Representation of the mechanism induced by traps on the average drain current.

A texto se le esta asignado m documets join permite que ese lo tome

```
146
147     ]
148
149     texto=" ".join(my_documents)
```

Fig. 4. Representation of the mechanism induced by traps on the average drain current.

Permite que dentro de los documentos no se tomen en cuenta los símbolos en cuenta y además que salte de documento en documento

```
    quitar=", : ; . \n !\""
```

Fig. 5. Representation of the mechanism induced by traps on the average drain current.

Con esta línea de código se da inicio al contabilizador de lo que dura el proceso de ejecución

```
    start_time =time.time()
```

Fig. 6. Representation of the mechanism induced by traps on the average drain current.

Línea con ciclo for que toma los caracteres dentro de los documentos y toma función quitar para eliminar los símbolos mencionados anteriormente

```
for caracter in quitar:
```

Fig. 7. Representation of the mechanism induced by traps on the average drain current.

Hace que los símbolos se reemplacen por espacios en la blanco

```
texto=texto.replace(caracter,  " ")
```

Fig. 8. Representation of the mechanism induced by traps on the average drain current.

En esta línea de código empleamos el función Split para poder separar y evaluar individualmente las palabras

```
texto=" ".join( texto.split() )
```

Fig. 9. Representation of the mechanism induced by traps on the average drain current.

En esta línea de código se crea el diccionario que a va a contener las palabras con su latencia

```
diccionario_frecuencia={}
```

Fig. 10. Representation of the mechanism induced by traps on the average drain current.

Con este par de Código se da apertura a la ejecución de la función que nos va a permitir medir la complejidad espacial

```
size_my_documents = sys.getsizeof(my_documents)

size_diccionario_frecuencia = sys.getsizeof(diccionario_frecuencia)
```

Fig. 11. Representation of the mechanism induced by traps on the average drain current.

En resumen, en estas líneas de código el for con el condicional anidado se encarga de verificar si la palabra ya esta o no dentro de el diccionario para adicionarla o sumarle

```
for palabra in palabras:

        if palabra in diccionario_frecuencia:

            diccionario_frecuencia[palabra]+=1

        else :

            diccionario_frecuencia[palabra]=1

#print(diccionario_frecuencia)
```

Fig. 12. Representation of the mechanism induced by traps on the average drain current.

Se encarga de ordenar la lista de mayor a menor con la llave lambda y el reverse se pone por que la llave originalmente lo ordena de menor a mayor

```
ordenardatos=dict(sorted(diccionario_frecuencia.items(),key=lambda item:item[1],reverse=True))
```

Fig. 13. Representation of the mechanism induced by traps on the average drain current.

Imprime la lista

```
ordenardatos=dict(sorted(diccionario_frecuencia.items(),key=la

for x in ordenardatos:

      print(x,ordenardatos[x])
```

Fig. 14. Representation of the mechanism induced by traps on the average drain current.

Definición de la función para buscar la palabra dentro de documentos

```
def buscar(word,documentos):

    posicion=[]

    for documento in documentos: #i es cada documento

       if  word in documento :

          posicion.append(documentos.index(documento))

    return posicion
```

Fig. 15. Representation of the mechanism induced by traps on the average drain current.

Código para ver la complejidad temporal y espacial

```
print ("desarrollo", buscar(" desarrollo ",my_documents ))




end_time =time.time()


print("el tiempo de ejecucio ={}".format(end_time - start_time))


size_my_documents_after = sys.getsizeof(my_documents)
size_diccionario_frecuencia_after = sys.getsizeof(diccionario_frecuencia)
memory_diff_my_documents = size_my_documents_after - size_my_documents
memory_diff_diccionario = size_diccionario_frecuencia_after - size_diccionario_frecuencia
print("Uso de memoria para my_documents antes del procesamiento:", size_my_documents, "bytes")
print("Uso de memoria para diccionario_frecuencia antes del procesamiento:", size_diccionario_frecuencia, "bytes")


print("Uso de memoria para my_documents después del procesamiento:", size_my_documents_after, "bytes")
print("Uso de memoria para diccionario_frecuencia después del procesamiento:", size_diccionario_frecuencia_after, "bytes")


print("Diferencia en uso de memoria para my_documents:", memory_diff_my_documents, "bytes")
print("Diferencia en uso de memoria para diccionario_frecuencia:", memory_diff_diccionario, "bytes")
```

Fig. 16. Representation of the mechanism induced by traps on the average drain current.

III. Conclusion

En resumen, esta parte del código toma el diccionario que contiene las frecuencias de las palabras y lo ordena en función de las frecuencias en orden descendente. Luego, imprime cada palabra junto con su frecuencia, mostrando primero las palabras con mayor frecuencia. Este proceso proporciona una visión clara de cuáles son las palabras más frecuentes en el texto.