

Taller-3-4 Buscador

1st Juan Esteban Pinzon Preciado
Fundación Universitaria Konrad Lorenz
Bogotá, Colombia
Juane.Pinzonp@konradlorenz.edu.co

Abstract—Esta publicación proporcionará detalles sobre el desarrollo y la implementación de ambos algoritmos. Además, se explorará el papel de los índices invertidos en la recuperación de información, destacando su papel fundamental en los motores de búsqueda y el procesamiento de textos a gran escala. Utilizando estos algoritmos y su comprensión de los índices invertidos, se pretende mejorar la capacidad para analizar y extraer información valiosa de grandes cantidades de datos de texto.

Index Terms—Optimización, recursos, lógica, complejidad, análisis

I. INTRODUCCIÓN

La capacidad de procesar y analizar eficazmente grandes conjuntos de datos de texto desempeña un papel crítico en campos como el procesamiento del lenguaje natural y la recuperación de información. En este artículo, exploraremos algoritmos diseñados para abordar dos tareas fundamentales en este contexto: la clasificación de palabras más frecuentes y la recuperación de documentos que contienen palabras clave específicas. Estos algoritmos utilizan conceptos como la memorización y la indexación inversa para mejorar la capacidad de extraer información valiosa de vastos volúmenes de datos de texto.

A lo largo de este análisis, examinaremos en detalle el código y sus componentes para comprender cómo estos algoritmos pueden ser implementados de manera efectiva. Además, exploraremos su utilidad en aplicaciones prácticas, como motores de búsqueda y análisis de texto a gran escala. Al final de este artículo, habremos adquirido una comprensión sólida de cómo estos algoritmos pueden contribuir significativamente a la mejora del procesamiento de datos textuales.

II. ANÁLISIS CÓDIGO

A. Ocurrencia de palabras

```
def contar_ocurrencias(palabra, diccionario):  
    if palabra not in diccionario:  
        return 0  
    ocurrencias = 0  
    for documento in diccionario[palabra]:  
        ocurrencias += documento.count(palabra)  
    return ocurrencias
```

B. Clasificación palabras

```
def clasificar_palabras(diccionario):  
    palabras_ordenadas = sorted(diccionario.keys(),  
                                key=lambda palabra: contar_ocurrencias(palabra))  
    print("Clasificación de palabras más repetidas")  
    for palabra in palabras_ordenadas:  
        ocurrencias = contar_ocurrencias(palabra, diccionario)  
        print(f"{palabra}: {ocurrencias} veces")
```

C. Ranking palabras más concurridas

```
for documento in my_documents:  
    for palabra in documento.split():  
        if palabra not in diccionario:  
            diccionario[palabra] = []  
            diccionario[palabra].append(documento)  
    clasificar_palabras(diccionario)
```

D. Buscar palabra en específico

```
def buscar(diccionario, palabra_a_buscar):  
    if palabra_a_buscar in diccionario:  
        return diccionario[palabra_a_buscar]  
    else:  
        return []
```

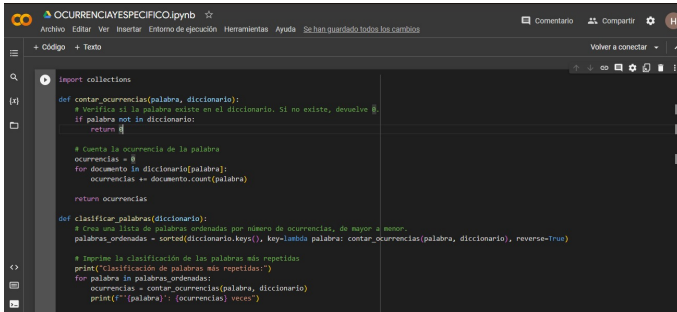
E. Cargar diccionario con la información a iterar

```
for i, documento in enumerate(my_documents):  
    palabras = documento.split()  
    for palabra in palabras:  
        if palabra not in diccionario:  
            diccionario[palabra] = [i]  
        else:  
            diccionario[palabra].append(i)
```

F. Palabra específica a buscar

```
palabra_a_buscar = "Python"  
documentos = buscar(diccionario, palabra_a_buscar)  
if documentos:  
    print(f"Documentos que contienen '{palabra_a_buscar}'")  
    for documento_index in documentos:  
        print(my_documents[documento_index])  
else:  
    print(f"'{palabra_a_buscar}' no se encontró")
```

III. CÓDIGO



```
import collections

def contar_ocurrencias(palabra, diccionario):
    # Verifica si la palabra existe en el diccionario. Si no existe, devuelve 0.
    if palabra not in diccionario:
        return 0

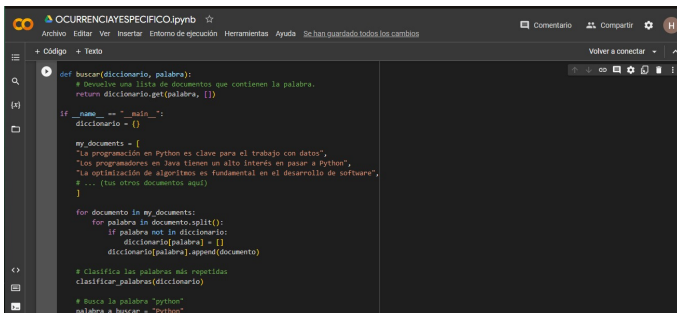
    # Cuenta la ocurrencia de la palabra
    ocurrencias = 0
    for documento in diccionario[palabra]:
        ocurrencias += documento.count(palabra)

    return ocurrencias

def clasificar_palabras(diccionario):
    # Crea una lista de palabras ordenadas por número de ocurrencias, de mayor a menor.
    palabras_ordenadas = sorted(diccionario.keys(), key=lambda palabra: contar_ocurrencias(palabra, diccionario), reverse=True)

    # Imprime la clasificación de las palabras más repetidas
    print("Clasificación de palabras más repetidas")
    for palabra in palabras_ordenadas:
        ocurrencias = contar_ocurrencias(palabra, diccionario)
        print(f"({palabra}): {ocurrencias} veces")
```

Fig. 1. Importaciones y creación de biblioteca para los datos.



```
def buscar(diccionario, palabra):
    # Devuelve una lista de documentos que contienen la palabra.
    return diccionario.get(palabra, [])

if __name__ == "__main__":
    diccionario = {}

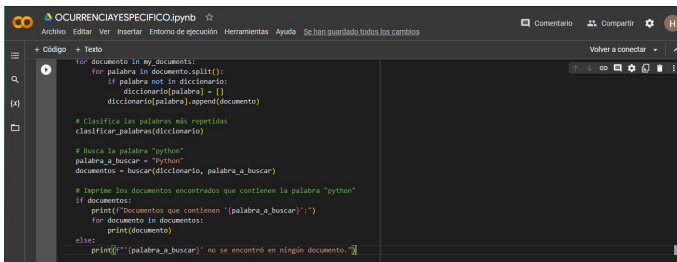
    my_documents = [
        "La programación en Python es clave para el trabajo con datos",
        "Los programadores en Java tienen un alto interés en pasar a Python",
        "La optimización de algoritmos es fundamental en el desarrollo de software",
        # ... (Los otros documentos aquí)
    ]

    for documento in my_documents:
        for palabra in documento.split():
            if palabra not in diccionario:
                diccionario[palabra] = []
            diccionario[palabra].append(documento)

    # Clasifica las palabras más repetidas
    clasificar_palabras(diccionario)

    # Busca la palabra "python"
    palabra_a_buscar = "python"
```

Fig. 2. Se realiza el ciclo para que pase por cada una de las palabras.



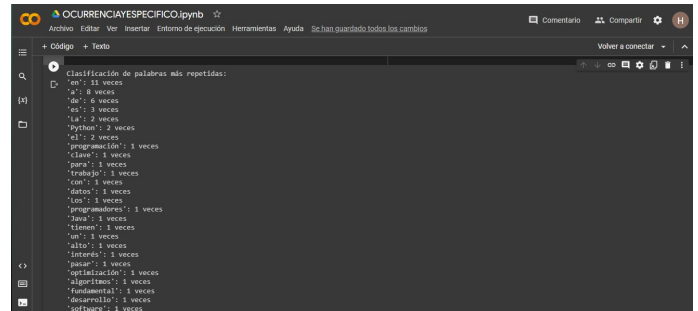
```
for documento in my_documents:
    for palabra in documento.split():
        if palabra not in diccionario:
            diccionario[palabra] = []
        diccionario[palabra].append(documento)

# Clasifica las palabras más repetidas
clasificar_palabras(diccionario)

# Busca la palabra "python"
palabra_a_buscar = "python"
documentos = buscar(diccionario, palabra_a_buscar)

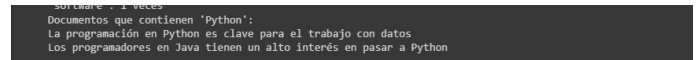
# Imprime los documentos encontrados que contienen la palabra "python"
if documentos:
    print("Documentos que contienen '" + palabra_a_buscar + "':")
    for documento in documentos:
        print(documento)
else:
    print(f"({palabra_a_buscar}) no se encontró en ningún documento.")
```

Fig. 3. Clasificación de palabras.



```
Clasificación de palabras más repetidas:
"en": 13 veces
"se": 8 veces
"de": 6 veces
"en": 6 veces
"la": 3 veces
"python": 2 veces
"el": 2 veces
"programación": 1 veces
"clave": 1 veces
"para": 1 veces
"trabajo": 1 veces
"con": 1 veces
"datos": 1 veces
"los": 1 veces
"programadores": 1 veces
"java": 1 veces
"tiempo": 1 veces
"en": 1 veces
"alto": 1 veces
"interés": 1 veces
"optimización": 1 veces
"algoritmos": 1 veces
"fundamental": 1 veces
"desarrollo": 1 veces
"software": 1 veces
```

Fig. 4. Palabras más repetidas.



```
Documentos que contienen 'python':
La programación en Python es clave para el trabajo con datos.
Los programadores en Java tienen un alto interés en pasar a Python
```

Fig. 5. Búsqueda de la palabra.

IV. CONCLUSIONES

- 1) Este código proporciona una funcionalidad eficiente y versátil para el procesamiento de grandes volúmenes de datos de texto. Su implementación sencilla lo hace adecuado para una variedad de aplicaciones, desde la clasificación de palabras hasta la recuperación de documentos. Su capacidad para manejar grandes conjuntos de datos lo convierte en una herramienta valiosa en un entorno de información en constante crecimiento. La utilización de diccionarios permite una relación efectiva entre palabras y los documentos que las contienen, garantizando un acceso rápido y eficiente a la información relevante. Además, la presentación ordenada de las palabras más frecuentes junto con sus apariciones contribuye a una comprensión clara de los datos, lo que resulta especialmente valioso en aplicaciones relacionadas con la búsqueda y el análisis de texto. En resumen, este código es una solución eficiente y versátil que puede ser una base sólida para proyectos de investigación y desarrollo en el campo del procesamiento de lenguaje natural y la recuperación de información.