Contador y buscador

Arturo Daza - 506231039

I. Introducción

Dos algoritmos que nos permite entender más la estructura de datos y la complejidad algoritmica son un contador de palabras y la busqueda de una palabra en una lista. Para abordar estos desafío, se ha desarrollado un algoritmo que aprovecha la técnica de memoización para contar y clasificar las palabras más repetidas en un conjunto de documentos. Este algoritmo es versátil, ya que permite realizar esta tarea tanto con como sin la inclusión de "stop words", palabras comunes que a menudo se excluyen del análisis para centrarse en las palabras más significativas. Además la una función de búsqueda de palabras, que proporciona los índices de posición de una palabra (que se pide al usuario) en la lista de documentos, esto permite una identificación rápida y efectiva de dónde y cómo se utiliza una palabra en el conjunto de datos.

II. Desarollo

II-A. Algoritmo con técnica memoización

El algoritmo presentado aquí se compone de varias funciones interconectadas. En primer lugar, se realiza un proceso de preprocesamiento del texto para convertir todas las letras a minúsculas y dividir el texto en palabras individuales. A continuación, se utiliza la técnica de memoización para contar las ocurrencias de cada palabra en el texto. La memoización es una estrategia que almacena resultados previos para evitar cálculos repetitivos y, de esta manera, mejora la eficiencia del algoritmo.

El algoritmo permite dos modos de funcionamiento: uno en el que se incluyen "stop wordsz otro en el que se excluyen. Las "stop words"son palabras comunes como "de", .[el]", "la", etc., que a menudo se eliminan del análisis de texto debido a su frecuente aparición y falta de relevancia en la identificación de palabras clave.

En el proceso de evaluación de rendimiento utilizando la biblioteca time de Python, se observa una diferencia notable al optar por incluir o excluir las "stop words". Esto se debe a una característica clave del algoritmo: al excluir las "stop words", se genera una lista adicional que contiene todas estas palabras. Luego, se realiza una comprobación extra para verificar si cada palabra en la lista de documentos no se encuentra en este conjunto de palabras excluidas antes de contarla. A continuación, se detallan los resultados de este análisis temporal de rendimiento:

- ***Resultados con "stop word": tiempo de ejecición de 0.024244 segundos***
- ***Resultados sin "stop word": tiempo de ejecición de 0.074673 segundos***

Añadiendo al análisis anterior, ***la complejidad algorítmica utilizando la notación Big O, tomando el peor de los casos, es O(n) para ambos algoritmos***. Esto se debe a que solo se ha agregado una declaración ïf.ªdicional, lo que significa que aún hay un solo bucle que itera hasta el tamaño "n", que corresponde al tamaño de la lista.

II-B. Algoritmo de búsqueda

Para el algoritmo de busqueda se utilizo le algoritmo de indices invertidos. En lugar de escanear todos los documentos cada vez que se busca una palabra, un índice invertido precomputa las relaciones entre palabras y documentos. Esto permite una búsqueda eficiente y rápida, ya que solo es necesario buscar en el índice invertido en lugar de recorrer todo el corpus de documentos.

La complejidad del algoritmo para crear un índice invertido es generalmente de O(N * M), donde N es el número de documentos y M es el número promedio de palabras por documento. En la fase de construcción del índice, se recorren todos los documentos y se procesan todas las palabras en cada documento. Por lo tanto, la complejidad es proporcional al tamaño del corpus completo.

Por otro lado, la complejidad de búsqueda en un índice invertido es generalmente de O(1). Debido a la estructura de datos eficiente del índice invertido, la búsqueda se reduce a una operación de búsqueda en un diccionario, que es una operación de tiempo constante en promedio. Esto significa que la búsqueda es muy rápida y no depende del tamaño total de la colección de documentos, sino solo de la cantidad de documentos que contienen la palabra clave.

Al realizar pruebas de complejidad temporal en el algoritmo de búsqueda basado en índices invertidos, hemos obtenido un tiempo máximo de ejecución de tan solo 0.000999 segundos. Este resultado demuestra la eficiencia de esta técnica, ya que incluso en el peor de los casos, cuando se busca una palabra en todos los documentos, el tiempo de búsqueda es extremadamente bajo. De hecho, la complejidad del algoritmo es tan baja que tener cien ceros a la izquierda del cero no sería suficiente para calcular el tiempo en segundos, lo que ilustra la velocidad con la que se puede acceder a la información en un índice invertido.

III. Conclusión

En conclusión, los algoritmos presentados ofrecen soluciones eficaces para dos desafíos fundamentales relacionados con la estructura de datos y la complejidad algorítmica en el procesamiento de texto. El primero utiliza la técnica de memoización para contar y clasificar las palabras más repetidas en un conjunto de documentos, lo que resulta en una

complejidad algorítmica de O(n) en el peor de los casos. Además, el algoritmo es versátil, permitiendo el procesamiento tanto con como sin palabras comunes ("stop words"), lo que afecta ligeramente el rendimiento, pero sigue siendo rápido en ambos casos.

Por otro lado, el segundo algoritmo emplea índices invertidos para la búsqueda eficiente de palabras en documentos. Este enfoque demuestra ser altamente eficiente, con tiempos de ejecución extremadamente bajos, incluso en el peor de los casos. Su complejidad algorítmica es de O(N * M) para la construcción del índice, donde N es el número de documentos y M es el número promedio de palabras por documento, mientras que la búsqueda en el índice invertido tiene una complejidad de O(1), lo que lo convierte en una herramienta poderosa para la recuperación de información y motores de búsqueda. Ambos algoritmos destacan la importancia de elegir la estructura de datos y el enfoque algorítmico adecuados para abordar desafíos específicos en el procesamiento de texto y la recuperación de información.