Comparacion de algoritmos

Implementada:

RabinKarp

El tiempo de complejidad de este algoritmo de búsqueda de subcadena Rabin-Karp es aproximadamente O((n - m) * m), donde n es la longitud del texto txt y m es la longitud del patrón pat. Aquí hay una desglose de las operaciones principales en el algoritmo:

1. Calcular el hash del patrón y del texto inicial: O(m).
2. Iterar sobre cada ventana de tamaño m en el texto: O(n - m).
3. Calcular el hash del texto actual: O(1).
4. Comparar el hash del patrón con el hash del texto actual: O(1).
5. En el peor de los casos, comparar cada carácter del patrón con el texto actual: O(m).

De la Librería:

Exact word match

Considere estos Elementos definidos en dos Documentos diferentes:

Wayne Gracia Jr.
Grace Hilton

Con un simple proceso de tokenización, cada palabra aquí puede considerarse un token, y si otro elemento tiene la misma palabra, se califica según el número de tokens coincidentes. En este ejemplo, las palabras Wayne y Grace coinciden con 2 palabras de un total de 3 en cada elemento. Un mecanismo de puntuación les igualará con un resultado de 0,67

Consideramos que Exact word match, es con el que deberiamos comparar debido a que RabinKarp se uso mas que todo para comparacion de textos planos, y Exact Word Match es el algoritmo que usa la libreria para este tipo de comparaciones. Pero durante la busqueda de donde se encuentra el metodo exacto, donde se aplica el exact word match, no pudimos encontrar el metodo en la libreria, ya que por la naturaleza del algoritmo podria ser un equals de caracteres. Por lo que para realizar el analisis se esta tomando en cuenta el algoritmo NGram en la seccion de Utils que igualmente trabaja sobre Strings.

El primer algoritmo utiliza la técnica de n-gramas para generar subcadenas de longitud fija a partir de una cadena dada. Este enfoque tiene una complejidad de tiempo de O(n * m), donde n es la longitud de la cadena de entrada y m es el tamaño del n-grama. Este algoritmo construye n-gramas de la cadena de entrada y los devuelve como una secuencia de cadenas.

Por otro lado, el segundo algoritmo es una implementación del algoritmo de búsqueda de Rabin-Karp, que busca un patrón dado en una cadena de texto utilizando el hash del patrón y deslizando una ventana sobre la cadena de texto. La complejidad de tiempo promedio del algoritmo de búsqueda de Rabin-Karp es O(n + m), donde n es la longitud de la cadena de texto y m es la longitud del patrón. Sin embargo, en el peor de los casos, la complejidad puede llegar a ser O(n * m).

Comparación:

Complejidad de tiempo: En términos de complejidad de tiempo, el algoritmo de búsqueda de Rabin-Karp tiende a ser más eficiente en el peor de los casos, especialmente para patrones largos y cadenas de texto grandes. Sin embargo, en el mejor de los casos, la técnica de n-gramas puede ser más rápida, especialmente para patrones y cadenas de texto relativamente pequeñas.

Precisión: En cuanto a la precisión, ambos algoritmos tienen el potencial de encontrar coincidencias exactas entre el patrón y la cadena de texto. Sin embargo, el algoritmo de búsqueda de Rabin-Karp puede requerir precauciones adicionales para manejar colisiones de hash y false positives, lo que puede afectar su precisión en ciertos casos.