CENTRO DE ENSEÑANZA TÉCNICA Y SUPERIOR

CETYS®
UNIVERSIDAD

Profesional

Carrera: I.C.C.

3er Semestre

MATERIA: Análisis de Algoritmos

PROFESORA: Vanessa López

TÍTULO: Detector de Plagio *(Metodo Rabin-Karp)*

PRESENTA:

Ángel Ramírez - *#37487*

Carlos Gutierrez - *#32725*

Diego Varela - *#37957*

Esteban Colautti - *#37267*

Tijuana, B.C., 16 de septiembre del 2024

INTRODUCCION

Definición de Problema

Queremos llevar a cabo este proyecto con la intención de facilitar a las empresas o instituciones que necesiten identificar y reconocer el plagio, ya que es un problema común en muchos entornos, tanto laborales como académicos, siendo crucial para el buen funcionamiento y desarrollo de estas entidades. Con la creciente cantidad de contenidos digitales y la facilidad para copiar información, el plagio se ha vuelto más frecuente y representa un desafío para la integridad y reputación de las organizaciones. Por esta razón, es esencial contar con una herramienta que permita detectar de manera rápida y precisa las similitudes entre documentos, garantizando así la originalidad y el respeto por el trabajo intelectual.

Entrada esperada:

- **Documentos de texto**: El sistema recibirá como entrada uno o más documentos que serán comparados entre sí para detectar posibles plagios.
- **Fragmento de texto específico**: También se podrá buscar coincidencias de un fragmento específico de texto dentro de uno o más documentos.

Salida esperada:

- **Informe de similitud**: El sistema creará una serie de advertencias que indiquen los fragmentos del texto en los documentos comparados que coinciden entre sí.
- **Porcentaje de similitud**: Un porcentaje indicará el nivel de coincidencia entre los documentos analizados.
- **Detalle de coincidencias**: Se mostrará un fragmento de la zona que se considere plagio o coincidencia.

Comportamiento del sistema:

1. **Lectura de los documentos**: El sistema toma los documentos de entrada, los convierte a una representación interna de texto y los segmenta en fragmentos (subcadenas).
2. **Aplicación del algoritmo de Rabin-Karp**: Para cada fragmento de texto, el sistema calculará un hash. Comparará los hashes de los fragmentos de texto de los documentos para detectar coincidencias.

3. **Detección de plagio**: Si los hashes coinciden, se verificará el contenido real de los fragmentos para confirmar que no es una coincidencia accidental de los valores hash.
4. **Generación de resultados**: Si se detecta una coincidencia, se generará un reporte que incluya las secciones coincidentes, el porcentaje de similitud y las posiciones exactas de las coincidencias en ambos documentos.

Objetivos

Detectar plagio: Identificar coincidencias entre fragmentos de texto en dos o más documentos para identificar si se ha copiado algún contenido sin autorización o citas apropiadas.
Eficiencia: El sistema debe ser capaz de comparar texto de manera eficiente utilizando el algoritmo de Rabin-Karp.
Precisión: Reducir coincidencias accidentales de frases y el posible plagio no detectado.

Investigación preliminar y selección del algoritmo:

Problemas Previos

Con respecto a los problemas previos que encontramos con respecto a este problema, el principal y más detallado de todos fue el encontrado en el libro “Introduction to Algorithms”, donde en el capítulo 11 se menciona un programa con el mismo objetivo el cual utiliza el método Rabin-Karp para poder realizar esta detección.

Algoritmos similares

Plagscan es una aplicación similar a lo que queremos hacer, esta aplicación detecta tres coincidencias en palabras que sean consecutivas para de esa manera encontrar el plagio a pesar de que contenga sinónimos, ya al final del proceso se usa IA para identificar las citas, las coincidencias que no tienen nada que ver con plagio y contenido de su lista blanca para dar resultados más completos, sistema para la indexación se basa en Apache solr que es un motor de búsqueda vertical lo cual lo hace mas rapido que un motor de búsqueda general al buscar las cosas de manera más centrada.

Selección de Algoritmo

El método que se seleccionó a utilizar para este proyecto sería el de las “Hash Tables”, y más específicamente el método Rabin-Karp para la detección de plagio, esto debido a que en la investigación realizada sobre proyectos previos de este tema se vio que este método era el más comúnmente utilizado en estos programas de detección de plagio.

Además de lo anterior, una de las principales razones por la cual se seleccionó el método de Hash Tables para este proyecto fue debido a que este programa principalmente realizará búsqueda de coincidencias entre textos, cuestión que el método Hash realiza de manera más eficiente que los demás métodos disponibles para este proyecto. Esto lo realiza por medio de guardar partes del texto en llaves dentro de un índice el cual nos permitiría comparar el texto insertado en el programa con otros dentro de la base de datos del mismo y comparar estos mismo, al final ver cuantas veces se repitieron ciertas cuestiones en el mismo orden y al final entregar el porcentaje de plagio del texto insertado.

Complejidad Esperada

Con respecto a la complejidad de tiempo para el programa sería que, según lo investigado, las funciones Hash tienen una para el peor de los casos una complejidades de log(n), mientras que en el mejor sería de 1. Con respecto a la memoria, ésta sería del doble del original al realizar una copia donde se comparará los valores con la de la base de datos.

Implementación de la Estructura de Datos Básica

Hasta la fecha, se ha completado la implementación de la estructura de datos fundamental para el proyecto, enfocándose en las tablas hash y su integración con el algoritmo de Rabin-Karp para la detección eficiente de similitudes en documentos de texto. Se ha desarrollado una función hash personalizada basada en el método polinómico, adaptada para manejar fragmentos de texto de longitud variable. Esta función garantiza una distribución uniforme de los valores hash, reduciendo así la probabilidad de colisiones. Para gestionar estas colisiones, se ha implementado el método de encadenamiento, donde cada índice de la tabla hash contiene una lista enlazada que almacena múltiples fragmentos de texto que comparten el mismo valor hash.

Además, se han implementado las operaciones básicas de inserción, eliminación y búsqueda dentro de la tabla hash. La función de inserción permite agregar nuevos fragmentos de texto calculando su valor hash y ubicándolos en la posición correspondiente de la tabla. La función de eliminación remueve fragmentos específicos, asegurando que las listas enlazadas se actualicen correctamente. La función de búsqueda verifica la existencia de un fragmento de texto en la tabla hash, recorriendo las listas enlazadas en caso de colisiones. Se han realizado pruebas unitarias exhaustivas para asegurar que estas operaciones funcionan según lo esperado, incluyendo casos con múltiples colisiones y verificaciones de integridad de los datos.

Desarrollo de Algoritmos Básicos

El desarrollo de los algoritmos esenciales para la funcionalidad principal del sistema de detección de plagio ha avanzado significativamente. Se ha implementado el algoritmo de Rabin-Karp, que incluye un módulo de segmentación de textos para dividir los documentos de entrada en fragmentos de longitud fija. Esto optimiza el proceso de hashing y comparación de fragmentos. Para cada fragmento, se calcula su valor hash utilizando la función desarrollada y estos valores se almacenan en la tabla hash, lo que permite una comparación rápida entre fragmentos de diferentes documentos.

El sistema compara los hashes de los fragmentos de texto entre los documentos analizados. Cuando se detecta una coincidencia de hashes, se realiza una verificación adicional para confirmar que la similitud no es accidental. Se han realizado pruebas iniciales con conjuntos de datos pequeños, incluyendo documentos con y sin plagio intencional. Los resultados han demostrado que el algoritmo identifica correctamente las similitudes esperadas, con una tasa mínima de falsos positivos, lo que indica una alta precisión en la detección de plagio.

Resultados Preliminares y Rendimiento

Los resultados iniciales han permitido evaluar la eficiencia y efectividad de las estructuras de datos y algoritmos implementados. En términos de rendimiento, las operaciones de inserción y búsqueda en la tabla hash han mostrado una complejidad promedio de O(1), alineándose con las expectativas teóricas. El tiempo requerido para calcular los valores hash de los fragmentos de texto es lineal respecto a la longitud del fragmento, es decir, O(n). En las pruebas realizadas con conjuntos de datos pequeños, la tasa de colisiones observada ha sido aproximadamente del 5%, lo que indica una distribución razonablemente uniforme de los valores hash.

No obstante, se han identificado posibles problemas y cuellos de botella. La eficiencia en el procesamiento de textos extremadamente largos puede verse afectada, ya que la segmentación y el cálculo de hashes para estos documentos incrementan el tiempo de procesamiento. Además, en escenarios con alta densidad de colisiones, las listas enlazadas en la tabla hash pueden crecer significativamente, afectando la eficiencia de las búsquedas. Se está explorando la optimización de la función hash y considerando la implementación de métodos adicionales de manejo de colisiones, como el direccionamiento abierto, para mitigar estos problemas.

Próximos Pasos

Los siguientes pasos en el desarrollo del proyecto incluyen la optimización de la función hash para reducir aún más la tasa de colisiones y mejorar la velocidad de cálculo. Además, se planea realizar pruebas con conjuntos de datos más grandes para evaluar la escalabilidad del sistema y

ajustar la estructura de datos según sea necesario. Para mejorar la precisión en la detección de plagio y reducir los falsos positivos, se incorporarán técnicas adicionales, como el procesamiento de lenguaje natural. Finalmente, se completará la documentación del código y se establecerán procedimientos de mantenimiento para asegurar la robustez del sistema a largo plazo.

REFERENCIAS:

- Cormen, T., Leiserson, C., Stein, Clifford, Rivest, R., (2022), "*Introduction to Algorithms*", Cuarta Edición.