UNIVERSIDAD NACIONAL DE SAN AGUSTÍN
FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS
ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

CURSO: ESTRUCTURA DE DATOS Y ALGORITMO

ACTIVIDAD: Detector de plagio

Profesor: Ing. Jorge Cristhian Chamby Diaz

Integrantes:

- Aco Tito, Anthony Edwin - aacot@unsa.edu.pe
- Chura Puma Mario Franco - mchurapum@unsa.edu.pe
- Contreras Mamani, Claudia - ccontrerasma@unsa.edu.pe
- Llaique Chullunquia, Angie Carolina - allaiquec@unsa.edu.pe

Link del repositorio: https://github.com/Mario-Chura/Proyecto-EDA.git

AREQUIPA – PERU

2022

METODOLOGÍA:

Estructura de datos: Como estructura de datos hemos escogido un árbol AVL por las siguientes características:

- La propiedad de balanceo garantiza que la altura del árbol sea de O(log n).
- En cada nodo del árbol se guarda información de la altura.
- La altura del árbol vacío es -1.
- Al realizar operaciones de inserción o eliminación se debe actualizar la información de altura de los nodos y recuperar la propiedad de balanceo si fuera necesario, es decir, si hubiera sido destruida.

Clases utilizadas:

***Class Node*:** Estructura del nodo del árbol AVL (Cada nodo contendrá un grupo de 10 palabras)

Class AVLTree: Implementación del arbol AVL (Se implementan métodos de inserción, balance, rotación a la derecha, rotación a la izquierda)

Class Phrase: Esta clase almacena un grupo de 10 palabras en su atributo "*Data*", luego estos grupos de 10 palabras mediante el método "*addword*" se usaran para ser insertados en el árbol AVL de la base de datos, también se usa el método "*addword*"para separar en grupos de 10 palabras el texto a ser comparado.

Class Document: En esta clase posee como atributos nombre *"fileName"* y el grupo de 10 palabras *"phase"*; además posee el métodos *"createAVL()"* el cual nos permite crear los árboles de la base de datos; también tenemos el método *"matching"* el cual realiza la búsqueda del un frase de 10 palabra en los diferentes árboles de la base de datos.

Class PlagiarismChecker: Esta clase implementa el método *"LoadFiles(String[] paths)"* el cual nos permite cargar los documentos para la base y el documento a revisar, también tenemos el método *"verifyPlagiarism(String path)"* en el cual se hacen las verificaciones para corroborar si hay plagio

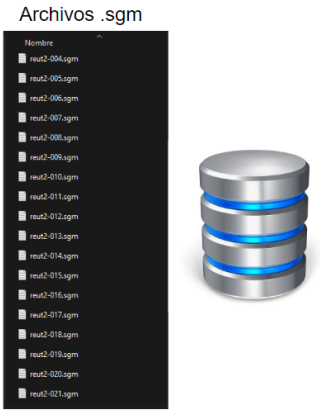
Class ResultChecker: Esta clase tiene como atributos un arreglo de booleanos con los resultados de las verificaciones.

Class GUI: Esta clase contiene la interfaz para cargar archivos y realizar la ejecución de las verificaciones.

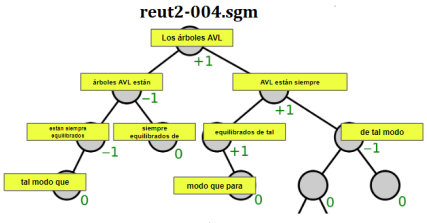
Class Main: Esta clase inicia la ejecución de la interfaz.

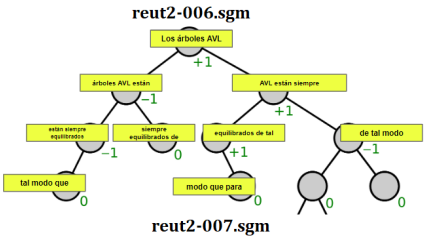
Funcionamiento:

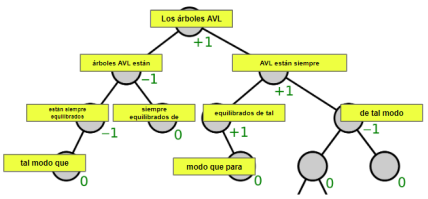
1. Primero se ingresa la base de datos los cuales están compuestos por los archivos .sgm

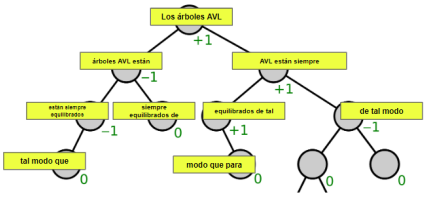


2. La base de datos mediante la clase *"createAVL()"* se introduce en un árbol, este árbol tendrá como nodos un grupo de 10 palabras, este grupo de palabras se forman mediante el método *"addword(String word)"*

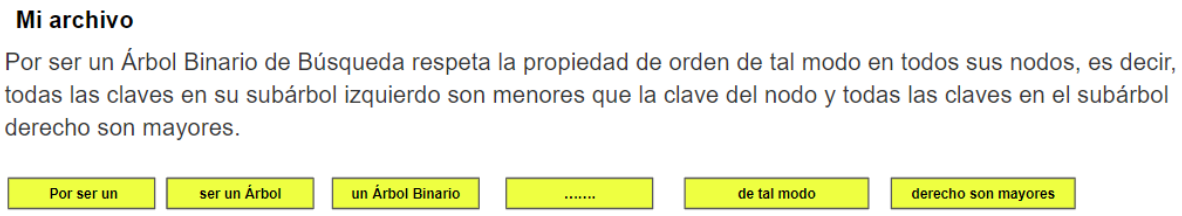




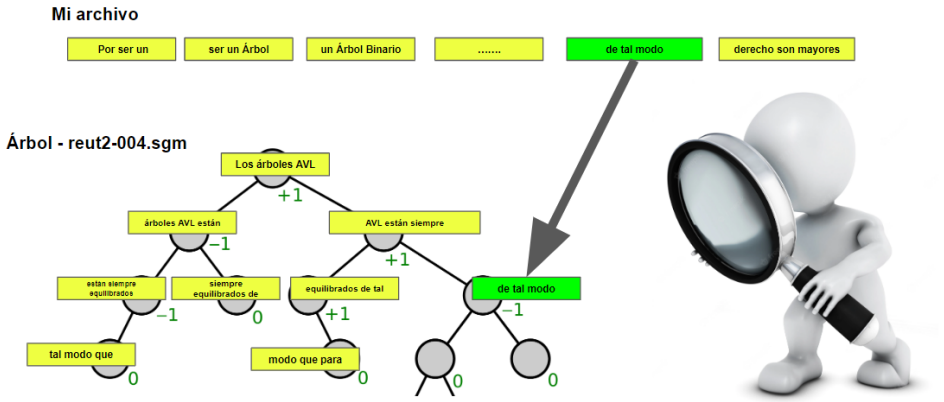




3. Se introduce el texto que deseamos verificar si es plagio, para ello este texto mediante el método *"addword(String word)"* dentro de un bucle irá formando grupos de 10 palabras como se muestra en la imagen:

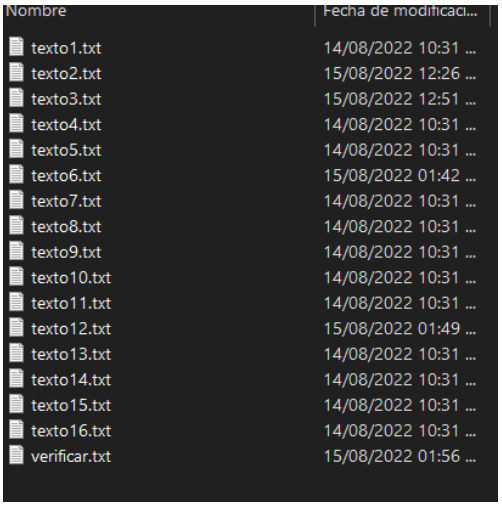


4. Se realizará la búsqueda de cada grupo de 10 palabras del texto a buscar en cada árbol que conforman la base de datos esta verificación usa el método *"matching",* si encuentra la coincidencia de un grupo de 10 palabras se le considera plagio.

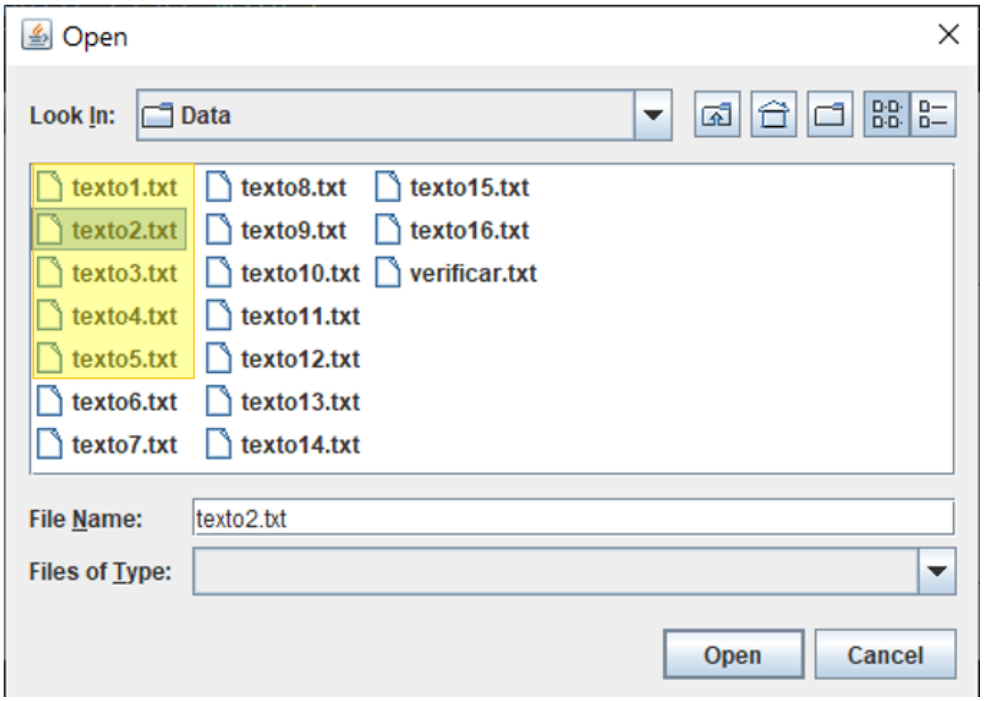


RESULTADOS DE EXPERIMENTOS:

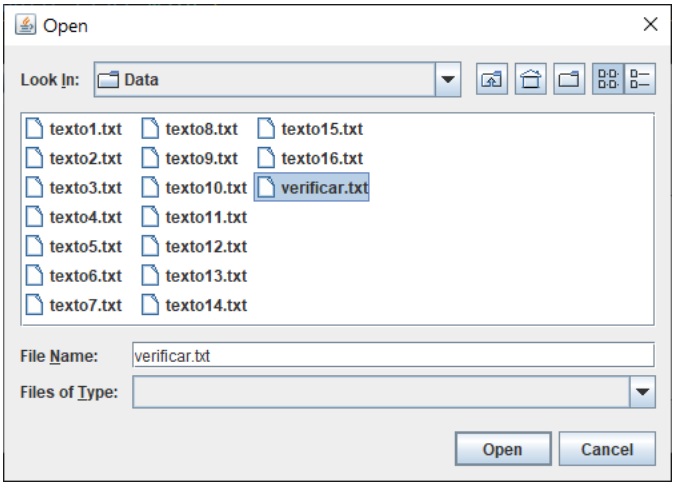
Para los experimentos, nosotros subimos previamente varios archivos de texto a la base de datos de la carpeta Data, así como también el archivo que va a ser verificado, en este caso "verificar.txt"



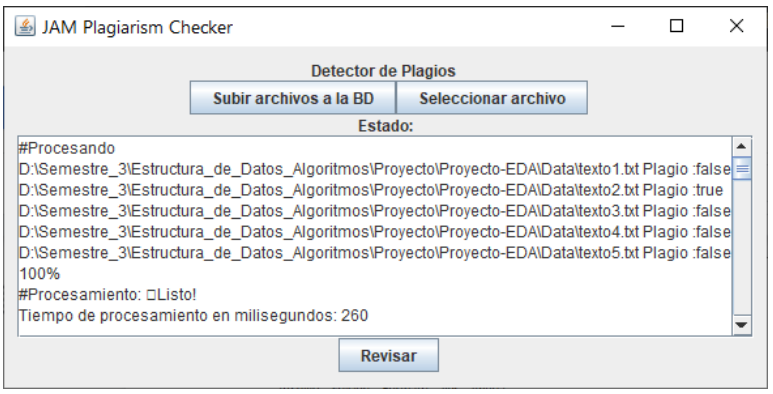
Para empezar subimos las direcciones de los archivos de texto que están en la base de datos, en este caso solo usaremos cinco.

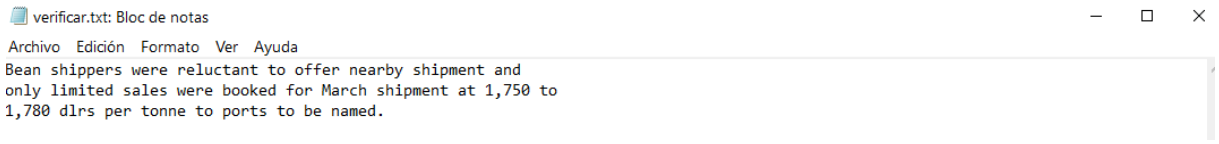


Luego indicamos el archivo que analizaremos.

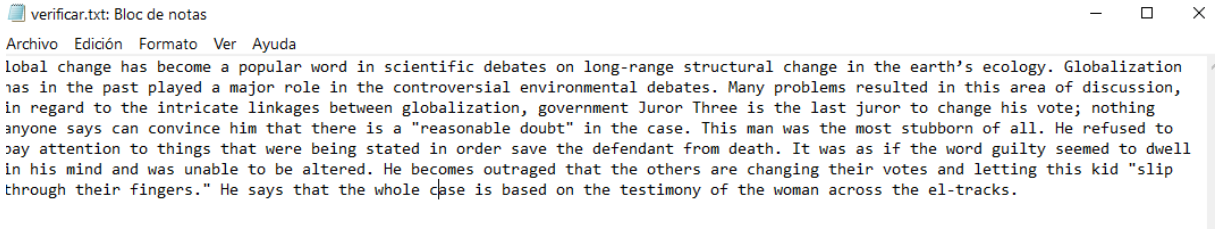


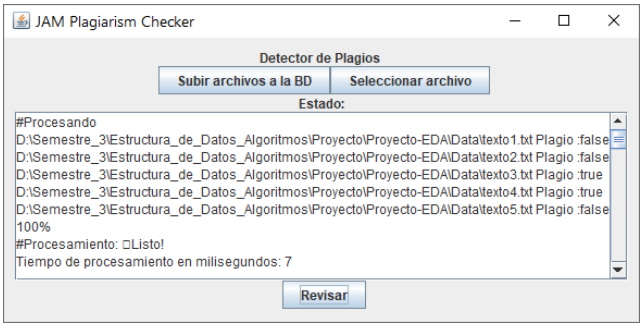
En este caso nosotros, habíamos copiado tal cual una parte del texto 2, por lo que nos mostró que si había plagio y fue extraído de “texto2.txt”



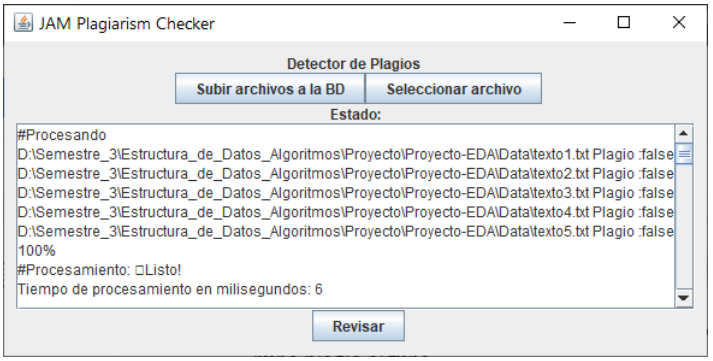


Como una segunda prueba copiamos palabras de “texto3.txt” y “texto4.txt”, y como era de esperarse nos indicaba los respectivos resultados.





Ahora probando, con el archivo de comparación vacío, y como era de esperar indicó que no hubo plagio alguno.



CONCLUSIONES:

- En el presente trabajo pudimos conocer más sobre el árbol AVL, ya que la implementamos como estructura para el almacenamiento de la base de datos, siendo usado con frecuencia su operación de búsqueda el cual es de orden O(log n), la que nos permite encontrar coincidencias para verificar si el archivo de entrada tiene plagio o no.

- Tras el análisis, estudio e investigación acerca de los conceptos importantes a tener en cuenta, más la ayuda y consejos proporcionados por el docente, se pudo hallar la mejor manera para desarrollar y programar bien el sistema con mayor facilidad.

- Al tener la libertad de poder usar diferentes clases ya existentes, se nos hizo más fácil algunas acciones como la de subir archivos con la clase JFileChooser.

- Como limitaciones tuvimos la adaptación al modelo ya predeterminado por el docente, eso con respecto a la Clase *PlagiarismChecker* y *ResultChecker*. Se hizo complicado plasmar nuestras ideas en la estructura de nuestro proyecto.

- Se tomó en cuenta que los textos compartidos por el docente estaban tanto en español como en inglés así que nos limitó a quitar los conectores, pero no nos impidió reemplazar los signos de puntuación que son universales.

REFERENCIAS:

Pérez, G. M. C.-. (2016). *Definición de los árboles AVL*. AVL Definición. Recuperado 2022, de http://163.10.22.82/OAS/AVL_Definicion/index.html

Javaz, Z. (s. f.). *Diseño e implementación de un árbol binario balanceado (árbol AVL) de algoritmo y estructura de datos Java - programador clic*. Programmerclick. Recuperado 15 de agosto de 2022, de https://programmerclick.com/article/3400189243/

w3schools (s. f.). Java Tutorial. Recuperado 15 de agosto de 2022, de https://www.w3schools.com/java/default.asp

Oracle. (s. f.). Java Platform SE 7. Recuperado 15 de agosto de 2022, de https://docs.oracle.com/javase/7/docs/api/