
	Universidad Nacional de San Agustín de Arequipa Facultad de Ingeniería de Producción y Servicios Escuela Profesional de Ingeniería de Sistemas	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLD-001	Página: 1

INFORME DE LABORATORIO

INFORMACION BASICA					
ASIGNATURA:	Estructura de Datos y Algoritmos				
TITULO DE LA PRACTICA:	Laboratorio - Trabajo Final				
NUMERO DE LA PRACTICA:	07	AÑO LECTIVO:	2023 - A	N° SEMESTRE:	III
FECHA DE PRESENTACION:	19 Julio 2023	HORA DE PRESENTACION:	23:59 PM		
INTEGRANTE (s): <ul style="list-style-type: none"> Coronado Peña Javier Forocca Mamani Maxs Sebastian Joaquin Diego Claudio Nina Suyu 				NOTA:	
DOCENTE (s): <ul style="list-style-type: none"> Mg. Edith Giovanna Cano Mamani 					

1. Tarea

Este proyecto tiene por objetivo que los alumnos implementen un sistema de detección de plagio simple (sistema a ser implementado), en el cual, usando como referencia una base de datos, el usuario del sistema enviará un párrafo escrito por el mismo (o copiado de alguna fuente) y lo enviará al sistema, el cual se encarga de realizar las consultas sobre la base de datos a fin de determinar si existió plagio o no.

Consideraciones importantes

1. **Eficiencia del sistema:** En general, los sistemas priman la eficiencia y la velocidad con la que realizan sus funcionalidades, y en este proyecto no podíamos dejar de lado esto. Garantizar velocidad en detectores de plagio muchas veces requiere que se realicen pre-procesamientos sobre los textos, a fin de reducir la cantidad de contenido a ser consultado.



Pre-procesamientos comunes incluyen: eliminación de caracteres especiales (por ejemplo, puntos, comas, parentesis, etc.), extracción de raíces de las palabras (por ejemplo, “act-” sería la raíz de las palabras “actor”, “actores”, “actriz”, etc.), etc. Estos pre-procesamientos reducen considerablemente la complejidad de la búsqueda de coincidencias, pues son menos comparaciones a ser realizadas.

Sin embargo, el uso de pre-procesamientos algunas veces impactan en la eficiencia de la detección de plagio, pues se pierde la comparación exacta de palabras y frases.

2. **Número de integrantes:** Este proyecto será realizado en grupos entre 3 y 4 integrantes. Cada grupo tendrá un líder, el cual debe enviar en las fechas correspondientes los diferentes entregables solicitados por los profesores.

2. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.

	<p>Universidad Nacional de San Agustín de Arequipa Facultad de Ingeniería de Producción y Servicios Escuela Profesional de Ingeniería de Sistemas</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLD-001</p>	<p>Página: 2</p>

- https://github.com/MaxsForocca/Lab_EDA_PlagiarismDetector.git

3. Implementacion del codigo

3.1. Clase NodeTrie

Esta clase representa un nodo en la estructura de datos Trie.

Atributos:



- `NodeTrie[] children`: Arreglo de nodos hijos que representa las letras del alfabeto.
- `boolean isWord`: Indica si el nodo actual marca el final de una palabra.

Métodos:

- `NodeTrie()`: Constructor que inicializa el arreglo de nodos hijos y establece `isWord` como falso.
- `getChildren()`: Retorna el arreglo de nodos hijos.
- `isAWord()`: Devuelve `true` si el nodo actual marca el final de una palabra.
- `setIsWord(boolean isWord)`: Establece el indicador `isWord` para marcar si el nodo forma una palabra.
- `getChild(char c)`: Retorna el nodo hijo correspondiente a la letra `c`.
- `setChild(char c, NodeTrie child)`: Establece el nodo hijo `child` correspondiente a la letra `c`.

```

1
2 package Trie;
3
4 public class NodeTrie {
5     private NodeTrie[] children;
6     private boolean isWord;
7
8     public NodeTrie(){
9         children = new NodeTrie[26];
10        isWord = false;
11    }
12
13    public NodeTrie[] getChildren(){
14        return children;
15    }
16
17    public boolean isAWord(){
18        return isWord;
19    }
20
21    public void setIsWord(boolean isWord){
22        this.isWord = isWord;
23    }
24
25    public NodeTrie getChild(char c){
26        return children[c - 'a'];
27    }
28
29    public void setChild(char c, NodeTrie child){
30        this.children[c - 'a'] = child;
31    }
32 }
```

	<p>Universidad Nacional de San Agustín de Arequipa Facultad de Ingeniería de Producción y Servicios Escuela Profesional de Ingeniería de Sistemas</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLD-001</p>	<p>Página: 3</p>

3.2. Clase Trie

Esta clase representa la estructura de datos Trie que almacena palabras y permite búsquedas eficientes.

Atributos:

- `NodeTrie root`: El nodo raíz del Trie.

Métodos:

- `Trie()`: Constructor que inicializa la estructura Trie creando un nodo raíz.
- `insert(String word)`: Inserta una palabra en el Trie, creando nodos hijos según las letras de la palabra.
- `search(String word)`: Realiza una búsqueda en el Trie para verificar si una palabra existe en él.

```



1 package Trie;
2
3
4 public class Trie {
5     private NodeTrie root;
6
7     public Trie(){
8         root = new NodeTrie();
9     }
10
11     public void insert(String word){
12         NodeTrie aux = root;
13         for(char c : word.toCharArray() ){
14             if(aux.getChild(c) == null){
15                 aux.setChild(c, new NodeTrie());
16             }
17             aux = aux.getChild(c);
18         }
19         aux.setIsWord(true);
20     }
21
22     public boolean search(String word){
23         NodeTrie aux = root;
24         // recorrer segun los caracteres de word
25         for(char c : word.toCharArray()){
26             aux = aux.getChild(c);
27             if(aux == null)
28                 return false;
29         }
30         // si no una palabras
31         return aux.isAWord();
32     }
33 }
```

3.3. Clase PlagiarismDetector

Esta clase se encarga de cargar los archivos de texto, construir los tries y realizar la detección de plagio utilizando la estructura Trie.

Atributos:

- `static String folderPath`: Ruta de la carpeta que contiene los archivos de texto.

	<p>Universidad Nacional de San Agustín de Arequipa Facultad de Ingeniería de Producción y Servicios Escuela Profesional de Ingeniería de Sistemas</p>	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLD-001	Página: 4

- `static File[] files`: Arreglo que almacena los archivos de texto en la carpeta.
- `static ArrayList<Trie> tries`: Lista de objetos Trie para almacenar las palabras de los archivos.

Métodos:



- `main(String[] args)`: Punto de entrada del programa que carga los archivos, construye los tries y realiza la detección de plagio.
- `loadFiles(File[] files)`: Carga los archivos de texto, procesa las palabras y crea los objetos Trie correspondientes.
- `procesarTexto(String str)`: Realiza un procesamiento de texto, quitando los caracteres con tildes o cambiando la 'ñ' por la 'n', retorna un arreglo de las palabras ya procesadas de la cadena str.
- `detectarPlagioPalabras(String texto)`: Realiza la detección de plagio para un texto ingresado, calculando el porcentaje de similitud con los tries almacenados.
- `verifyPlagiarism(String path)`: Método por implementar para verificar los resultados de la detección de plagio.

Esta clase en general busca determinar el porcentaje de similitud entre el texto ingresado y los archivos de texto previamente cargados en los tries. Cada trie se utiliza para verificar la presencia de palabras del texto ingresado en los archivos almacenados, calculando el porcentaje de coincidencia en relación al total de palabras en el texto.

```

1
2 package plagiarismdetector;
3
4 import java.io.File;
5 import java.io.FileNotFoundException;
6 import java.util.ArrayList;
7 import java.util.Scanner;
8
9 import Trie.Trie;
10 import java.text.Normalizer;
11
12 public class PlagiarismDetector {
13     static String folderPath = "./textos"; //ruta de la carpeta
14     static File folder = new File(folderPath); // indicamos la carpeta de archivos
15     static File[] files = folder.listFiles(); // obtenemos los archivos de la carpeta
16     static ArrayList<Trie> tries= new ArrayList<>(); //array de tries
17     public static void main(String[] args) {
18         loadFiles(files);
19         System.out.println(tries.get(0).search("constante"));
20         Scanner sc = new Scanner(System.in);
21         String textoIngresado = sc.nextLine();
22         ArrayList<Double> plagioDB = detectarPlagioPalabras(textoIngresado);
23         for(Double d : plagioDB){
24             System.out.println(d);
25         }
26     }
27     public static boolean loadFiles(File[] files){
28         if (files != null) {
29             // iterador para el arraylist de Trie
30             int idxTries = 0;
31             // recorrer los archivos del arreglo
32             for (File file : files) {



```

	<p>Universidad Nacional de San Agustín de Arequipa Facultad de Ingeniería de Producción y Servicios Escuela Profesional de Ingeniería de Sistemas</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLD-001</p>	<p>Página: 5</p>

```

33     if (file.isFile()) {
34         // leer archivo y crear un nuevo Trie para ese archivo
35         System.out.println("Leyendo archivo: " + file.getName());
36         tries.add(new Trie());
37         try {
38             Scanner scanner = new Scanner(file);
39             // recorrer cada linea del archivo
40             while (scanner.hasNextLine()) {
41                 String line = scanner.nextLine();
42                 // filtamos los caracteres especiales y lo volvemos en minusculas
43                 // cada palabra se queda guardado en un arreglo
44                 String[] words = procesarTexto(line);
45                 // introducimos cada palabra al Trie correspondiente
46                 for(String word:words)
47                     tries.get(idxTries).insert(word);
48                 System.out.println(line);
49             }
50             scanner.close();
51         } catch (FileNotFoundException e) {
52             System.out.println(e);
53         }
54         System.out.println("-----");
55     }
56     // aumentamos el iterador
57     idxTries++;
58 }
59 return true;
60 } else {
61     System.out.println("La carpeta no existe o no contiene archivos.");
62     return false;
63 }
64 }
65
66 public static String[] procesarTexto(String str){
67     String normalizedText = Normalizer.normalize(str, Normalizer.Form.NFD);
68     // Eliminar caracteres diacriticos
69     normalizedText = normalizedText.replaceAll("\\p{M}", "");
70     // Reemplazar caracteres especiales
71     normalizedText = normalizedText.replaceAll("'", "n");
72     // filtamos los caracteres especiales y lo volvemos en minusculas
73     // cada palabra se queda guardado en un arreglo
74     String[] words = normalizedText.toLowerCase().split("\\W+");
75     return words;
76 }
77 // Devuelve un arreglo del porcentaje de similitud de cada texto de la DB
78 public static ArrayList<Double> detectarPlagioPalabras(String texto){
79     String[] palabrasText = procesarTexto(texto);
80     int palabrasTotal = palabrasText.length;
81     ArrayList<Double> percentPlagio = new ArrayList<>();
82     for(Trie t: tries){
83         double percent = 0.0;
84         int cantPalabrasPlagio = 0;
85         for(String p: palabrasText){
86             if(t.search(p)){
87                 System.out.println(p);
88                 cantPalabrasPlagio++;
89             }

```

	<p>Universidad Nacional de San Agustín de Arequipa Facultad de Ingeniería de Producción y Servicios Escuela Profesional de Ingeniería de Sistemas</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLD-001</p>	<p>Página: 6</p>

```

90     }
91     percent = cantPalabrasPlagio * 100.0 / palabrasTotal;
92     percentPlagio.add(percent);
93 }
94 return percentPlagio;
95 }
96
97 public ResultChecker verifyPlagiarism(String path){
98     ResultChecker result = null;
99
100    return result;
101 }
102 }
```

3.4. Clase Vista



Esta clase se encarga realizar la interfaz gráfica de usuario GUI. Que utiliza la librería swing.

Atributos:

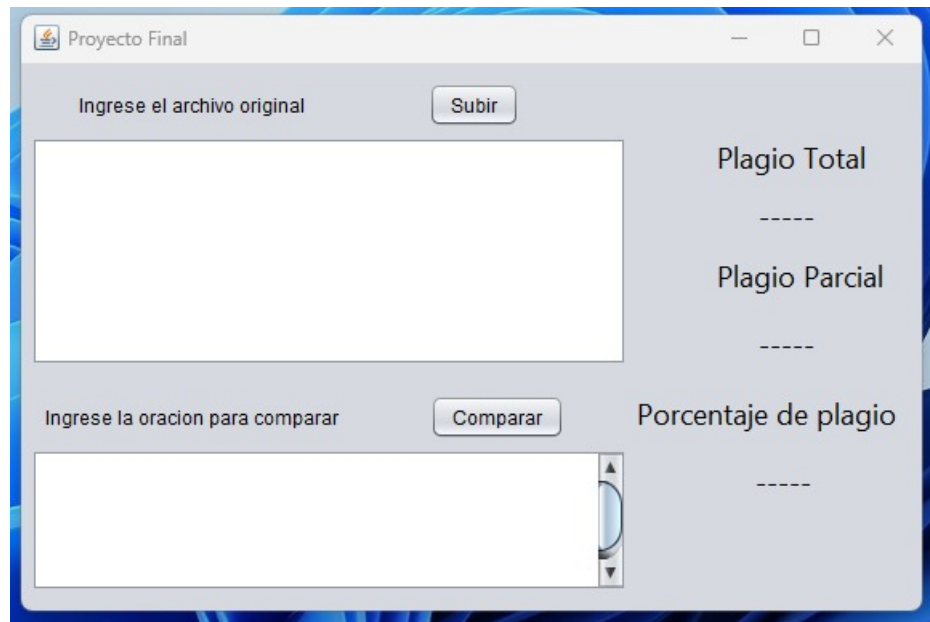
- `javax.swing.JButton btnCom, btnSel, btnSel2`: Botones de la interfaz.
- `javax.swing.JLabel jLabel1, jLabel2, jLabel3, jLabel4, jLabel5, jLabel6, jLabel7, lbPT`: Etiquetas de texto de la interfaz.
- `javax.swing.JPanel jPanel1`: Contenedor para los componentes de la interfaz.
- `javax.swing.JScrollPane jScrollPane1, jScrollPane2`: Barras de desplazamiento para componentes de la interfaz.
- `javax.swing.JTextArea txtaArchivo, txtaArchivo2`: Áreas de texto de la interfaz.

Métodos:

- `Vista()`: Constructor que llama al método `initComponents()` establece la ubicación de la ventana en el centro de la pantalla.
- `btnSelActionPerformed(java.awt.event.ActionEvent evt)`: Este método maneja los eventos del botón 'subir' y crea un cuadro de selección de archivos usando `JFileChooser`, el archivo seleccionado se lee mediante `FileReader` y `BufferedReader`, y su contenido se muestra en la primera área de texto mediante el método `setText()`.
- `btnSel2ActionPerformed(java.awt.event.ActionEvent evt)`: Lo mismo que el anterior.
- `main(String args[])`: El metodo principal que se encarga de configurar el aspecto visual de la aplicación y de crear y mostrar la ventana de la GUI.

	<p>Universidad Nacional de San Agustín de Arequipa Facultad de Ingeniería de Producción y Servicios Escuela Profesional de Ingeniería de Sistemas</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLD-001</p>	<p>Página: 7</p>

4. Vistas



The screenshot shows a web-based plagiarism detection interface. It has a title bar 'Proyecto Final' with standard window controls. The main area is divided into two columns. The left column contains two input fields: the top one is labeled 'Ingrese el archivo original' with a 'Subir' button, and the bottom one is labeled 'Ingrese la oracion para comparar' with a 'Comparar' button. The right column displays the results: 'Plagio Total' followed by a dashed line, 'Plagio Parcial' followed by a dashed line, and 'Porcentaje de plagio' followed by a dashed line. The interface has a blue and white color scheme.

5. Referencias

- <https://www.geeksforgeeks.org/trie-insert-and-search/>
- <https://www.baeldung.com/trie-java>
- <https://www.techiedelight.com/es/implement-trie-data-structure-java/>