Informe de Laboratorio Detector de Plagio

Tema: Proyecto Final

Nota

Estudiante	CUI	Correo
Condorios Yllapuma Jorge	20222076	jcondorios@unsa.edu.pe
Cusilayme García José Luis	20220598	jcusilaymeg@unsa.edu.pe
Mamani Mamani Alexis	20222066	almamanima@unsa.edu.pe
Valdivia Luna Carlo Joaquín	20220567	cvaldivialu@unsa.edu.pe

Facultad	Asignatura	Docente
Escuela Profesional de Ingeniería de Sistemas	Estructura de Datos Semestre: III	Mg. Edith Giovanna Cano Mamani ecanoma@unsa.edu.pe

Laboratorio	Tema	Duración
Detector de Plagio	Proyecto Final	02 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - A	Del 16 Julio 2023	Al 08 Agosto 2023

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:
  - **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.

- **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. Equipos, materiales y temas utilizados

- Sistema Operativo Windows 10 Home 64 bits (10.0, compilation 19045)
- VIM 9.0.
- OpenJDK 64-Bits 17.0.7.
- Git 2.39.2.
- Cuenta en GitHub con el correo institucional.
- Java

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/JorgeCY21/EDA_LAB_D`
- URL para el laboratorio 06 en el Repositorio GitHub.
- `https://github.com/JorgeCY21/EDA_LAB_D/tree/main/Lab_07`

4. Actividades con el repositorio GitHub

4.1. Creando e inicializando repositorio GitHub

- Se realizaron los siguientes comandos en la computadora:

Listing 1: Dirijíéndonos al directorio de trabajo

```
D:\
```

Listing 2: Clonando repositorio GitHub

```
D:\ git clone https://github.com/JorgeCY21/EDA_LAB_D
```

Listing 3: Inicializando directorio para laboratorio 06

```
D:\ cd EDA_LAB_D
D: \EDA_LAB_D> mkdir Lab_07
D: \EDA_LAB_D> cd Lab_07
```

4.2. Commits

Listing 4: Commit: Creamos el trie principal

```
D: \EDA_LAB_D\Lab_07> git add .
D: \EDA_LAB_D\Lab_07> git commit -m ""
D: \EDA_LAB_D\Lab_07> git push -u origin main
```

5. Explicación de Código

5.1. Resumen

El proyecto **Detector de Plagio** es una aplicación de detección de plagio que utiliza estructuras Trie (árbol de prefijos) para almacenar y comparar palabras entre diferentes archivos y texto ingresado.

La aplicación permite a los usuarios cargar archivos de texto, ingresar texto manualmente y luego verificar si hay similitudes o plagio entre los contenidos de estos archivos y el texto ingresado, mostrando los resultados a través de una interfaz gráfica. La implementación incluye clases para la GUI, la gestión de resultados y la manipulación eficiente de estructuras Trie.

El funcionamiento de este proyecto se divide en 4 componentes y se realiza de la siguiente manera:

- Los usuarios interactúan con la GUI para cargar archivos de texto, ingresar texto manualmente o realizar verificaciones de plagio. Mediante **GUI** (Clase): Se implementa la interfaz gráfica de usuario (GUI)
- La clase **PlagiarismChecker** maneja la carga de archivos y almacena su contenido en estructuras Trie. Proporciona métodos para verificar si un archivo contiene similitudes con otros archivos previamente cargados.
- La clase **TrieNode** Implementa la estructura de nodo Trie, permitiendo la inserción y búsqueda eficiente de palabras, así como la comparación de contenido entre diferentes nodos Trie, normalizando las palabras para una comparación uniforme.
- La clase **ResultChecker**Almacena y gestiona los resultados de detección de plagio para cada archivo, indicando si se encontraron similitudes o no.
- Y finalmente la **GUI** muestra los resultados de detección de plagio a través de la interfaz, resaltando si se encontraron similitudes o no entre los contenidos.

5.2. GUI

- **Importación de paquetes:** Se importan diferentes clases y paquetes necesarios para construir la interfaz gráfica y realizar operaciones de lectura de archivos y detección de plagio.

```
1  import java.awt.Color;
2  import java.awt.event.ActionEvent;
3  import java.awt.event.ActionListener;
4  import java.io.BufferedReader;
5  import java.io.File;
6  import java.io.FileReader;
7  import java.io.IOException;
8  import java.util.logging.Level;
9  import java.util.logging.Logger;
10 import javax.swing.BorderFactory;
11 import javax.swing.JButton;
12 import javax.swing.JFileChooser;
13 import javax.swing.JLabel;
14 import javax.swing.JOptionPane;
15 import javax.swing.JPanel;
16 import javax.swing.JScrollPane;
17 import javax.swing.JTextArea;
18 import javax.swing.SwingConstants;
19 import javax.swing.UIManager;
```

- **Variables de instancia y constantes:**
  - **HEIGHT** y **WIDTH**: Representan las dimensiones de la ventana.
  - **bodyPanel** Un panel donde se agregarán los componentes de la GUI.
  - Otros componentes como botones (btnVerifPlagio, btnSubirDoc, btnSubirArchivos), área de texto (textArea), seleccionador de archivos (fileChooser), entre otros.

```
23      private final int HEIGHT = 400;
24      private final int WIDTH = 700;
25
26      private JPanel bodyPanel;
27
28      private JButton btnVerifPlagio;
29      private JButton btnSubirDoc;
30      private JButton btnSubirArchivos;
31
32      private JTextArea textArea;
33      private JFileChooser fileChooser;
34      private PlagiarismChecker plagioChecker;
35
36      private GUI gui;
```

- **Constructor GUI:**
  - Configura la apariencia y comportamiento de la ventana principal.
  - Inicializa **bodyPanel** y lo establece como el contenido de la ventana.
  - Llama a **initComponents()** para inicializar los componentes de la GUI.
  - Finalmente, se hace visible la ventana

```
public GUI() {
    gui = this;

    this.setTitle("Detector de Plagio");
    this.setSize(WIDTH, HEIGHT);
    this.setLocationRelativeTo(null);
    this.setDefaultCloseOperation(EXIT_ON_CLOSE);

    bodyPanel = new JPanel();
    bodyPanel.setLayout(null);
    setContentPane(bodyPanel);

    initComponents();
    this.setVisible(true);

}
```

- **Método initComponents():**
  - Crea y coloca los componentes en el bodyPanel, como etiquetas, botones y un área de texto.
  - Asigna oyentes de eventos a los botones para manejar acciones como verificar plagio, cargar archivo y cargar archivos múltiples.

```
private void initComponents() {

    JLabel lbl = new JLabel("Ingrese su texto aquí o cargue sus archivos");
    lbl.setBounds(40, 23, 326, 40);
    bodyPanel.add(lbl);

    btnVerifPlagio = new JButton("Verificar plagio");
    btnVerifPlagio.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {

            try {
                if (plagioChecker.isEmpty()) {

                    JOptionPane.showMessageDialog(gui, "Debe de subir archivos a la base antes de comparar",
                            "ERROR - BASE DE DATOS VACIA", JOptionPane.ERROR_MESSAGE);

                    return;
                }

                gui.setSize(WIDTH, HEIGHT + 150);
                gui.setLocationRelativeTo(null);

                TrieNode aux = new TrieNode();
                String[] palabras = textArea.getText().split("\\s+");

                for (String palabra : palabras) {
                    aux.insert(palabra);
                }

                System.out.println("TextArea: " + aux);

                mostrarResultados(plagioChecker.verifyPlagiarism(aux).getResults());

            } catch (Exception error) {
                System.out.println(error.getMessage());
            }



                }
            }
        });
        btnVerifPlagio.setBounds(60, 300, 150, 40);
        bodyPanel.add(btnVerifPlagio);

        btnSubirDoc = new JButton("Cargar Archivo");
        btnSubirDoc.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                subirArchivoTextArea();
            }
        });
        btnSubirDoc.setBounds(472, 23, 150, 40);
        bodyPanel.add(btnSubirDoc);

        btnSubirArchivos = new JButton("Cargar Archivos");
        btnSubirArchivos.setBounds(472, 300, 150, 40);
        bodyPanel.add(btnSubirArchivos);
        btnSubirArchivos.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {

                mostrarFileChooser();

            }
        });

        textArea = new JTextArea(5, 20);
        textArea.setBackground(new Color(255, 255, 255));
        textArea.setLineWrap(true);
        textArea.setWrapStyleWord(true);

        JScrollPane scrollPane = new JScrollPane(textArea);
        scrollPane.setBounds(30, 80, WIDTH - 75, 200);
        bodyPanel.add(scrollPane);

    }
```

- **Método mostrarResultados(boolean[] results):**
  - Recibe un arreglo de booleanos que representan los resultados de detección de plagio para diferentes archivos.
  - Crea y muestra etiquetas que indican si hubo plagio o no para cada archivo en la parte inferior de la ventana.

```
protected void mostrarResultados(boolean[] results) {

    int iniColumna = 20;
    int numArchivo = 1;
    for (boolean bl : results) {
        mostrarColumna(bl, iniColumna, numArchivo);

        numArchivo++;
        iniColumna += 100;
    }
}
```

- **Método mostrarColumna(boolean bl, int iniColumna, int numArchivo):**
  - Crea y coloca etiquetas y paneles de color para mostrar los resultados de plagio para cada archivo.

```
private void mostrarColumna(boolean bl, int iniColumna, int numArchivo) {
    JLabel lblArchivo = new JLabel("Archivo N°" + numArchivo);
    JLabel lblSiNo = new JLabel((bl) ? "Si" : "No");
    JLabel color = new JLabel("      ");

    color.setOpaque(true);
    color.setBackground((bl) ? Color.GREEN : Color.RED);

    lblArchivo.setBounds(iniColumna, 400, 100, 30);
    lblArchivo.setBorder(BorderFactory.createLineBorder(Color.BLACK));
    lblArchivo.setHorizontalAlignment(SwingConstants.CENTER);
    lblArchivo.setOpaque(true);
    lblArchivo.setBackground(Color.WHITE);
    bodyPanel.add(lblArchivo);

    lblSiNo.setBounds(iniColumna, 430, 100, 30);
    lblSiNo.setBorder(BorderFactory.createLineBorder(Color.BLACK));
    lblSiNo.setHorizontalAlignment(SwingConstants.CENTER);
    lblSiNo.setOpaque(true);
    lblSiNo.setBackground(Color.WHITE);
    bodyPanel.add(lblSiNo);

    color.setBounds(iniColumna, 460, 100, 30);
    color.setBorder(BorderFactory.createLineBorder(Color.BLACK));
    bodyPanel.add(color);

}
```

- **Método subirArchivoTextArea():**
  - Muestra un cuadro de diálogo para seleccionar un archivo.
  - Lee el contenido del archivo seleccionado y lo muestra en el área de texto.

```
private void subirArchivoTextArea() {
    fileChooser = new JFileChooser();
    fileChooser.setMultiSelectionEnabled(false);

    int resultado = fileChooser.showOpenDialog(this);

    if (resultado == JFileChooser.APPROVE_OPTION) {
        File archivoSeleccionado = fileChooser.getSelectedFile();
        String path = archivoSeleccionado.getAbsolutePath();

        try {
            BufferedReader reader = new BufferedReader(new FileReader(path));
            StringBuilder contenido = new StringBuilder();
            String linea;

            while ((linea = reader.readLine()) != null) {
                contenido.append(linea).append("\n");
            }

            reader.close();

            textArea.setText(contenido.toString());
        } catch (IOException e) {
            e.printStackTrace();
        }

    }
}
```

- **Método mostrarFileChooser():**
  - Muestra un cuadro de diálogo para seleccionar múltiples archivos.
  - Inicializa un objeto PlagiarismChecker y carga los archivos seleccionados en él.

```
private void mostrarFileChooser() {
    fileChooser = new JFileChooser();
    fileChooser.setMultiSelectionEnabled(true);

    int resultado = fileChooser.showOpenDialog(this);

    if (resultado == JFileChooser.APPROVE_OPTION) {
        File[] archivosSeleccionados = fileChooser.getSelectedFiles();
        String[] paths = new String[archivosSeleccionados.length];

        plagioChecker = new PlagiarismChecker(archivosSeleccionados.length);

        for (int i = 0; i < archivosSeleccionados.length; i++) {
            paths[i] = archivosSeleccionados[i].getAbsolutePath();
        }

        plagioChecker.loadFiles(paths);

    }
}
```

- **Método main(String args[]):**
  - Llama al método mejorandoApariencia() para establecer la apariencia de la GUI.
  - Crea una instancia de la clase GUI, lo que inicia la aplicación.

```
public static void main(String args[]) {
    mejorandoApariencia();
    new GUI();
}
```

- **Método mejorandoApariencia():**
  - Configura la apariencia de la GUI utilizando el LookAndFeel "Nimbus"para mejorar la presentación visual.

```
private static void mejorandoApariencia() {
    try {
        for (UIManager.LookAndFeelInfo info : UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(GUI.class.getName()).log(Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {
        Logger.getLogger(GUI.class.getName()).log(Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {
        Logger.getLogger(GUI.class.getName()).log(Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
        Logger.getLogger(GUI.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

5.3. PlagiarismChecker

- **Definición de la clase PlagiarismChecker:** Esta clase parece ser parte del sistema de detección de plagio y está encargada de cargar y verificar la similitud entre archivos utilizando un enfoque basado en Trie (árbol de prefijos).
- **Variables de instancia y constructor:**
  - **Tries:** Un arreglo de nodos Trie (árbol de prefijos), uno para cada archivo a ser verificado.
  - **resultChecker:** Un objeto que parece ser responsable de verificar los resultados de plagio.
  - El constructor recibe la cantidad de archivos que se van a verificar y crea un arreglo de nodos Trie y un objeto ResultChecker.

```
private TrieNode[] tries;
private ResultChecker resultChecker;

public PlagiarismChecker(int num) {
    tries = new TrieNode[num];
    resultChecker = new ResultChecker(num);
}
```

- **Método loadFiles(String[] paths):**
  - Carga los contenidos de los archivos especificados en los caminos de archivo dados en los nodos Trie correspondientes.
  - Devuelve **false** en este caso. No se manejan errores de lectura de archivos, solo se asume que todo funciona correctamente.

```
public boolean loadFiles(String[] paths) {

    for (int i = 0; i < paths.length; i++) {
        TrieNode trie = new TrieNode();

        loadFile(paths[i], trie);

        tries[i] = trie;
    }

    return false;
}
```

- **Método isEmpty():**
  - Verifica si alguno de los nodos Trie está vacío (no se ha cargado ningún archivo).
  - Si al menos uno está vacío, devuelve true, lo que indica que la base de datos está vacía.

```
public boolean isEmpty() {
    for (TrieNode trie : tries) {
        if (trie == null)
            return true;
    }

    return false;
}
```

- **Método loadFile(String path, TrieNode trie):**
  - Lee el contenido del archivo especificado por **path**.
  - Divide cada línea en palabras y las inserta en el Trie especificado.
  - Devuelve **true** si la carga se realiza correctamente, y false si hay algún problema de lectura.

```
private boolean loadFile(String path, TrieNode trie) {

    try (BufferedReader reader = new BufferedReader(new FileReader(path))) {
        String line;
        while ((line = reader.readLine()) != null) {
            String[] palabras = line.split("\\s+");

            for (String palabra : palabras) {
                trie.insert(palabra);
            }

        }
        return true;

    } catch (IOException e) {

        return false;
    }
}
```

- **Método verifyPlagiarism(TrieNode trie):**
  - Recibe un nodo Trie que contiene las palabras del texto que se desea verificar.
  - Compara el nodo Trie recibido con cada uno de los nodos Trie almacenados en el arreglo **tries**.
  - Muestra información sobre los nodos Trie y establece si hay plagio o no en el objeto**resultChecker**.

```
public ResultChecker verifyPlagiarism(TrieNode trie) {

    for (int i = 0; i < tries.length; i++) {
        System.out.println("Trie N°" + (i + 1) + ": " + tries[i]);

        boolean isPlagio = trie.equals(tries[i]);
        resultChecker.setPlagio(i, isPlagio);

    }

    return resultChecker;

}
```

5.4. TrieNode

- **Definición de la clase TrieNode:** Esta clase implementa un nodo de un Trie (árbol de prefijos) para el almacenamiento y búsqueda eficiente de palabras en un contexto de detección de plagio.
- **Variables de instancia y constructor:**
  - **children:** Un mapa que asocia caracteres a nodos Trie hijos.
  - **isEndWord:** Un booleano que indica si el nodo Trie representa el final de una palabra.
  - El constructor inicializa el mapa de hijos y establece **isEndWord** en falso.

```
private Map<Character, TrieNode> children;
private boolean isEndWord;

private static final int MIN_IGUAL = 10;

public TrieNode() {
    children = new HashMap<>();
    isEndWord = false;
}
```

- **Método insert(String word):**
  - Inserta una palabra en el Trie, siguiendo el camino correspondiente en función de los caracteres de la palabra.

- Normaliza la palabra usando el método normalizeWord() y luego itera sobre cada carácter para construir el Trie.

```
public void insert(String word) {
    if (word.length() <= 2)
        return;

    TrieNode current = this;
    for (char c : normalizeWord(word).toCharArray()) {
        if (c != 0) {
            current.children.putIfAbsent(c, new TrieNode());
            current = current.children.get(c);
        }
    }
    current.isEndWord = true;
}
```

- **Método search(String word):**
  - Busca una palabra en el Trie, devolviendo verdadero si la palabra se encuentra en el Trie y representa el final de una palabra.

```
public boolean search(String word) {
    TrieNode node = searchNode(word);
    return node != null && node.isEndWord;
}
```

- **Método searchNode(String word):**
  - Similar al método search(), pero devuelve el nodo Trie que representa el final de la palabra buscada en lugar de un booleano.

```
private TrieNode searchNode(String word) {
    TrieNode current = this;
    for (char c : normalizeWord(word).toCharArray()) {
        if (c != 0) {
            current = current.children.get(c);
            if (current == null) {
                return null;
            }
        } else {
            return null;
        }
    }
    return current;
}
```

- **Método normalizeWord(String word):**
  - Normaliza una palabra eliminando acentos y caracteres especiales, dejando solo letras minúsculas y números.

```
private String normalizeWord(String word) {
    StringBuilder normalizedWord = new StringBuilder();

    for (int i = 0; i < word.length(); i++) {
        char c = word.charAt(i);
        String normalized = Normalizer.normalize(String.valueOf(c),
                Normalizer.Form.NFD).replaceAll("\\p{M}", "");

        if (!normalized.isEmpty()) {
            char normalizedChar = Character.toLowerCase(normalized.charAt(0));
            if ((normalizedChar >= 'a' && normalizedChar <= 'z')
                    || (normalizedChar >= '0' && normalizedChar <= '9')) {
                normalizedWord.append(normalizedChar);
            }
        }
    }

    return normalizedWord.toString();
}
```

- **Método toString():**
  - Devuelve una representación en cadena de todas las palabras almacenadas en el Trie.

```java
@Override
public String toString() {
    List<String> words = new ArrayList<>();
    collectWords(this, new StringBuilder(), words);
    return "words: " + words + "\n";
}
```

- **Método equals(TrieNode datos):**
  - Compara el contenido del Trie actual con otro Trie (datos) para determinar si hay una cantidad mínima de palabras en común, que se establece como MIN_ IGUAL.

```java
public boolean equals(TrieNode datos) {
    List<String> words = new ArrayList<>();
    collectWords(this, new StringBuilder(), words);
    int foundWords = 0;

    for (String word : words) {
        if (datos.search(word)) {
            foundWords++;
        }
    }

    return foundWords >= MIN_IGUAL;
}
```

- **Método collectWords():**
  - Recopila todas las palabras almacenadas en el Trie actual y las agrega a la lista **words**.
  - Utiliza una técnica de búsqueda recursiva para explorar todas las combinaciones de letras.

```
private void collectWords(TrieNode current, StringBuilder currentWord, List<String> words) {
    if (current.isEndWord) {
        words.add(currentWord.toString());
    }

    for (char c = 'a'; c <= 'z'; c++) {
        TrieNode child = current.children.get(c);
        if (child != null) {
            currentWord.append(c);
            collectWords(child, currentWord, words);
            currentWord.deleteCharAt(currentWord.length() - 1);
        }
    }

    for (char c = '0'; c <= '9'; c++) {
        TrieNode child = current.children.get(c);
        if (child != null) {
            currentWord.append(c);
            collectWords(child, currentWord, words);
            currentWord.deleteCharAt(currentWord.length() - 1);
        }
    }
}
```

5.5. ResultChecker

- **Definición de la clase ResultChecker:** Esta clase parece ser responsable de almacenar y gestionar los resultados de detección de plagio para cada archivo. Contiene métodos para configurar los resultados y obtenerlos.
- **Variables de instancia y constructor:**
  - **result:** Un arreglo de booleanos que almacena los resultados de detección de plagio para cada archivo.
  - El constructor recibe la cantidad de archivos que se van a verificar y crea un arreglo de booleanos para almacenar los resultados.
- **Método setPlagio(int index, boolean isPlagio):**
  - Establece el resultado de detección de plagio para un archivo específico en el índice dado.
  - Recibe el índice del archivo y un valor booleano isPlagio que indica si se detectó plagio o no en ese archivo.
  - Actualiza el valor en el arreglo **result** en la posición correspondiente.
- **Método getResults():**
  - Devuelve el arreglo de booleanos result que contiene los resultados de detección de plagio para todos los archivos.

```java
public class ResultChecker {

    private boolean[] result;

    public ResultChecker(int num) {
        result = new boolean[num];
    }

    public void setPlagio(int index, boolean isPlagio) {
        result[index] = isPlagio;

    }

    public boolean[] getResults() {
        return result;
    }

}
```

6. Códigos Fuente

Listing 5: GUI.java

```java
import java.awt.Color;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;

import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JFileChooser;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.SwingConstants;
import javax.swing.UIManager;

public class GUI extends javax.swing.JFrame {

```

```
24    private final int HEIGHT = 400;
25    private final int WIDTH = 700;
26
27    private JPanel bodyPanel;
28
29    private JButton btnVerifPlagio;
30    private JButton btnSubirDoc;
31    private JButton btnSubirArchivos;
32
33    private JTextArea textArea;
34    private JFileChooser fileChooser;
35    private PlagiarismChecker plagioChecker;
36
37    private GUI gui;
38
39    public GUI() {
40       gui = this;
41
42       this.setTitle("Detector de Plagio");
43       this.setSize(WIDTH, HEIGHT);
44       this.setLocationRelativeTo(null);
45       this.setDefaultCloseOperation(EXIT_ON_CLOSE);
46
47       bodyPanel = new JPanel();
48       bodyPanel.setLayout(null);
49       setContentPane(bodyPanel);
50
51       initComponents();
52       this.setVisible(true);
53
54    }
55
56    private void initComponents() {
57
58       JLabel lbl = new JLabel("Ingrese su texto aquí o cargue sus archivos");
59       lbl.setBounds(40, 23, 326, 40);
60       bodyPanel.add(lbl);
61
62       btnVerifPlagio = new JButton("Verificar plagio");
63       btnVerifPlagio.addActionListener(new ActionListener() {
64          public void actionPerformed(ActionEvent e) {
65
66             try {
67                if (plagioChecker.isEmpty()) {
68
69                   JOptionPane.showMessageDialog(gui, "Debe de subir archivos a la base antes
                         de comparar",
70                         "ERROR - BASE DE DATOS VACIA", JOptionPane.ERROR_MESSAGE);
71
72                   return;
73                }
74
75                gui.setSize(WIDTH, HEIGHT + 150);
76                gui.setLocationRelativeTo(null);
77
78                TrieNode aux = new TrieNode();
```

```
79              String[] palabras = textArea.getText().split("\\s+");
80
81              for (String palabra : palabras) {
82                 aux.insert(palabra);
83              }
84
85              System.out.println("TextArea: " + aux);
86
87              mostrarResultados(plagioChecker.verifyPlagiarism(aux).getResults());
88
89           } catch (Exception error) {
90              System.out.println(error.getMessage());
91           }
92        }
93     });
94     btnVerifPlagio.setBounds(60, 300, 150, 40);
95     bodyPanel.add(btnVerifPlagio);
96
97     btnSubirDoc = new JButton("Cargar Archivo");
98     btnSubirDoc.addActionListener(new ActionListener() {
99        public void actionPerformed(ActionEvent e) {
100          subirArchivoTextArea();
101       }
102    });
103    btnSubirDoc.setBounds(472, 23, 150, 40);
104    bodyPanel.add(btnSubirDoc);
105
106    btnSubirArchivos = new JButton("Cargar Archivos");
107    btnSubirArchivos.setBounds(472, 300, 150, 40);
108    bodyPanel.add(btnSubirArchivos);
109    btnSubirArchivos.addActionListener(new ActionListener() {
110       public void actionPerformed(ActionEvent e) {
111
112          mostrarFileChooser();
113
114       }
115    });
116
117    textArea = new JTextArea(5, 20);
118    textArea.setBackground(new Color(255, 255, 255));
119    textArea.setLineWrap(true);
120    textArea.setWrapStyleWord(true);
121
122    JScrollPane scrollPane = new JScrollPane(textArea);
123    scrollPane.setBounds(30, 80, WIDTH - 75, 200);
124    bodyPanel.add(scrollPane);
125
126 }
127
128 protected void mostrarResultados(boolean[] results) {
129
130    int iniColumna = 20;
131    int numArchivo = 1;
132    for (boolean bl : results) {
133       mostrarColumna(bl, iniColumna, numArchivo);
134
```

```
135         numArchivo++;
136         iniColumna += 100;
137      }
138   }
139
140   private void mostrarColumna(boolean bl, int iniColumna, int numArchivo) {
141      JLabel lblArchivo = new JLabel("Archivo N" + numArchivo);
142      JLabel lblSiNo = new JLabel((bl) ? "Si" : "No");
143      JLabel color = new JLabel("   ");
144
145      color.setOpaque(true);
146      color.setBackground((bl) ? Color.GREEN : Color.RED);
147
148      lblArchivo.setBounds(iniColumna, 400, 100, 30);
149      lblArchivo.setBorder(BorderFactory.createLineBorder(Color.BLACK));
150      lblArchivo.setHorizontalAlignment(SwingConstants.CENTER);
151      lblArchivo.setOpaque(true);
152      lblArchivo.setBackground(Color.WHITE);
153      bodyPanel.add(lblArchivo);
154
155      lblSiNo.setBounds(iniColumna, 430, 100, 30);
156      lblSiNo.setBorder(BorderFactory.createLineBorder(Color.BLACK));
157      lblSiNo.setHorizontalAlignment(SwingConstants.CENTER);
158      lblSiNo.setOpaque(true);
159      lblSiNo.setBackground(Color.WHITE);
160      bodyPanel.add(lblSiNo);
161
162      color.setBounds(iniColumna, 460, 100, 30);
163      color.setBorder(BorderFactory.createLineBorder(Color.BLACK));
164      bodyPanel.add(color);
165
166   }
167
168   private void subirArchivoTextArea() {
169      fileChooser = new JFileChooser();
170      fileChooser.setMultiSelectionEnabled(false);
171
172      int resultado = fileChooser.showOpenDialog(this);
173
174      if (resultado == JFileChooser.APPROVE_OPTION) {
175         File archivoSeleccionado = fileChooser.getSelectedFile();
176         String path = archivoSeleccionado.getAbsolutePath();
177
178         try {
179            BufferedReader reader = new BufferedReader(new FileReader(path));
180            StringBuilder contenido = new StringBuilder();
181            String linea;
182
183            while ((linea = reader.readLine()) != null) {
184               contenido.append(linea).append("\n");
185            }
186
187            reader.close();
188
189            textArea.setText(contenido.toString());
190         } catch (IOException e) {
```

```
191             e.printStackTrace();
192          }
193
194       }
195    }
196
197    private void mostrarFileChooser() {
198       fileChooser = new JFileChooser();
199       fileChooser.setMultiSelectionEnabled(true);
200
201       int resultado = fileChooser.showOpenDialog(this);
202
203       if (resultado == JFileChooser.APPROVE_OPTION) {
204          File[] archivosSeleccionados = fileChooser.getSelectedFiles();
205          String[] paths = new String[archivosSeleccionados.length];
206
207          plagioChecker = new PlagiarismChecker(archivosSeleccionados.length);
208
209          for (int i = 0; i < archivosSeleccionados.length; i++) {
210             paths[i] = archivosSeleccionados[i].getAbsolutePath();
211          }
212
213          plagioChecker.loadFiles(paths);
214
215       }
216    }
217
218    public static void main(String args[]) {
219
220       mejorandoApariencia();
221
222       new GUI();
223
224    }
225
226    private static void mejorandoApariencia() {
227       try {
228          for (UIManager.LookAndFeelInfo info : UIManager.getInstalledLookAndFeels()) {
229             if ("Nimbus".equals(info.getName())) {
230                UIManager.setLookAndFeel(info.getClassName());
231                break;
232             }
233          }
234       } catch (ClassNotFoundException ex) {
235          Logger.getLogger(GUI.class.getName()).log(Level.SEVERE, null, ex);
236       } catch (InstantiationException ex) {
237          Logger.getLogger(GUI.class.getName()).log(Level.SEVERE, null, ex);
238       } catch (IllegalAccessException ex) {
239          Logger.getLogger(GUI.class.getName()).log(Level.SEVERE, null, ex);
240       } catch (javax.swing.UnsupportedLookAndFeelException ex) {
241          Logger.getLogger(GUI.class.getName()).log(Level.SEVERE, null, ex);
242       }
243    }
244
245 }
```

Listing 6: PlagiarismChecker.java

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class PlagiarismChecker {

   private TrieNode[] tries;
   private ResultChecker resultChecker;

   public PlagiarismChecker(int num) {
      tries = new TrieNode[num];
      resultChecker = new ResultChecker(num);
   }

   // retornar que no hubo error en la lectura
   public boolean loadFiles(String[] paths) {

      for (int i = 0; i < paths.length; i++) {
         TrieNode trie = new TrieNode();

         loadFile(paths[i], trie);

         tries[i] = trie;
      }

      return false;
   }

   public boolean isEmpty() {
      for (TrieNode trie : tries) {
         if (trie == null)
            return true;
      }

      return false;
   }

   private boolean loadFile(String path, TrieNode trie) {

      try (BufferedReader reader = new BufferedReader(new FileReader(path))) {
         String line;
         while ((line = reader.readLine()) != null) {
            String[] palabras = line.split("\\s+");

            for (String palabra : palabras) {
               trie.insert(palabra);
            }

         }
         return true;

      } catch (IOException e) {

         return false;
      }
```

```
56    }
57
58    // path ruta del archivo donde colocamos el texto con/sin plagio.
59    public ResultChecker verifyPlagiarism(String path) {
60       ResultChecker result = null;
61
62       // retornar resultados del sistema
63
64       return result;
65
66    }
67
68    public ResultChecker verifyPlagiarism(TrieNode trie) {
69
70       for (int i = 0; i < tries.length; i++) {
71          System.out.println("Trie N" + (i + 1) + ": " + tries[i]);
72
73          boolean isPlagio = trie.equals(tries[i]);
74          resultChecker.setPlagio(i, isPlagio);
75
76       }
77
78       return resultChecker;
79
80    }
81
82 }
```

Listing 7: ResultChecker.java

```
1
2  public class ResultChecker {
3
4     private boolean[] result;
5
6     public ResultChecker(int num) {
7        result = new boolean[num];
8     }
9
10    public void setPlagio(int index, boolean isPlagio) {
11       result[index] = isPlagio;
12
13    }
14
15    public boolean[] getResults() {
16       return result;
17    }
18
19 }
```

Listing 8: TrieNode.java

```
1  import java.text.Normalizer;
2  import java.util.ArrayList;
3  import java.util.HashMap;
```

```
4  import java.util.List;
5  import java.util.Map;
6
7  public class TrieNode {
8     private Map<Character, TrieNode> children;
9     private boolean isEndWord;
10
11    private static final int MIN_IGUAL = 10;
12
13    public TrieNode() {
14       children = new HashMap<>();
15       isEndWord = false;
16    }
17
18    public void insert(String word) {
19       if (word.length() <= 2)
20          return;
21
22       TrieNode current = this;
23       for (char c : normalizeWord(word).toCharArray()) {
24          if (c != 0) {
25             current.children.putIfAbsent(c, new TrieNode());
26             current = current.children.get(c);
27          }
28       }
29       current.isEndWord = true;
30    }
31
32    public boolean search(String word) {
33       TrieNode node = searchNode(word);
34       return node != null && node.isEndWord;
35    }
36
37    private TrieNode searchNode(String word) {
38       TrieNode current = this;
39       for (char c : normalizeWord(word).toCharArray()) {
40          if (c != 0) {
41             current = current.children.get(c);
42             if (current == null) {
43                return null;
44             }
45          } else {
46             return null;
47          }
48       }
49       return current;
50    }
51
52    private String normalizeWord(String word) {
53       StringBuilder normalizedWord = new StringBuilder();
54
55       for (int i = 0; i < word.length(); i++) {
56          char c = word.charAt(i);
57          String normalized = Normalizer.normalize(String.valueOf(c),
               Normalizer.Form.NFD).replaceAll("\\p{M}", "");
58
```

```
59          if (!normalized.isEmpty()) {
60             char normalizedChar = Character.toLowerCase(normalized.charAt(0));
61             if ((normalizedChar >= 'a' && normalizedChar <= 'z')
62                   || (normalizedChar >= '0' && normalizedChar <= '9')) {
63                normalizedWord.append(normalizedChar);
64             }
65          }
66       }
67
68       return normalizedWord.toString();
69    }
70
71    @Override
72    public String toString() {
73       List<String> words = new ArrayList<>();
74       collectWords(this, new StringBuilder(), words);
75       return "words: " + words + "\n";
76    }
77
78    public boolean equals(TrieNode datos) {
79       List<String> words = new ArrayList<>();
80       collectWords(this, new StringBuilder(), words);
81       int foundWords = 0;
82
83       for (String word : words) {
84          if (datos.search(word)) {
85             foundWords++;
86          }
87       }
88
89       return foundWords >= MIN_IGUAL;
90    }
91
92    private void collectWords(TrieNode current, StringBuilder currentWord, List<String> words)
       {
93       if (current.isEndWord) {
94          words.add(currentWord.toString());
95       }
96
97       for (char c = 'a'; c <= 'z'; c++) {
98          TrieNode child = current.children.get(c);
99          if (child != null) {
100            currentWord.append(c);
101            collectWords(child, currentWord, words);
102            currentWord.deleteCharAt(currentWord.length() - 1);
103         }
104      }
105
106      for (char c = '0'; c <= '9'; c++) {
107         TrieNode child = current.children.get(c);
108         if (child != null) {
109            currentWord.append(c);
110            collectWords(child, currentWord, words);
111            currentWord.deleteCharAt(currentWord.length() - 1);
112         }
113      }
```

```
114     }
115 }
```

7. Ejecución Y pruebas

Listing 9: Texto original a comparar

```
La inteligencia artificial es un campo de la informatica que se centra en la creacion de
    sistemas capaces de realizar tareas que normalmente requieren inteligencia humana. El
    cambio climatico es un fenomeno global que esta siendo causado en gran medida por la
    actividad humana, como la emision de gases de efecto invernadero.
```

Listing 10: Texto con plagio 1

```
  La inteligencia artificial es un campo de la informatica que se centra en la creacion de
      sistemas capaces de realizar tareas que normalmente requieren inteligencia humana.
```

Listing 11: Texto original sin plagio

```
  La robotica es una rama de la tecnologia que se ocupa del disenho, construccion y
      operacion de robots con el objetivo de automatizar y facilitar diversas tareas.
```

Listing 12: Texto con plagio 2

```
  El cambio climatico es un fenomeno global que esta siendo causado en gran medida por la
      actividad humana, como la emision de gases de efecto invernadero.
```

Figura 1: Ejecutamos el GUI

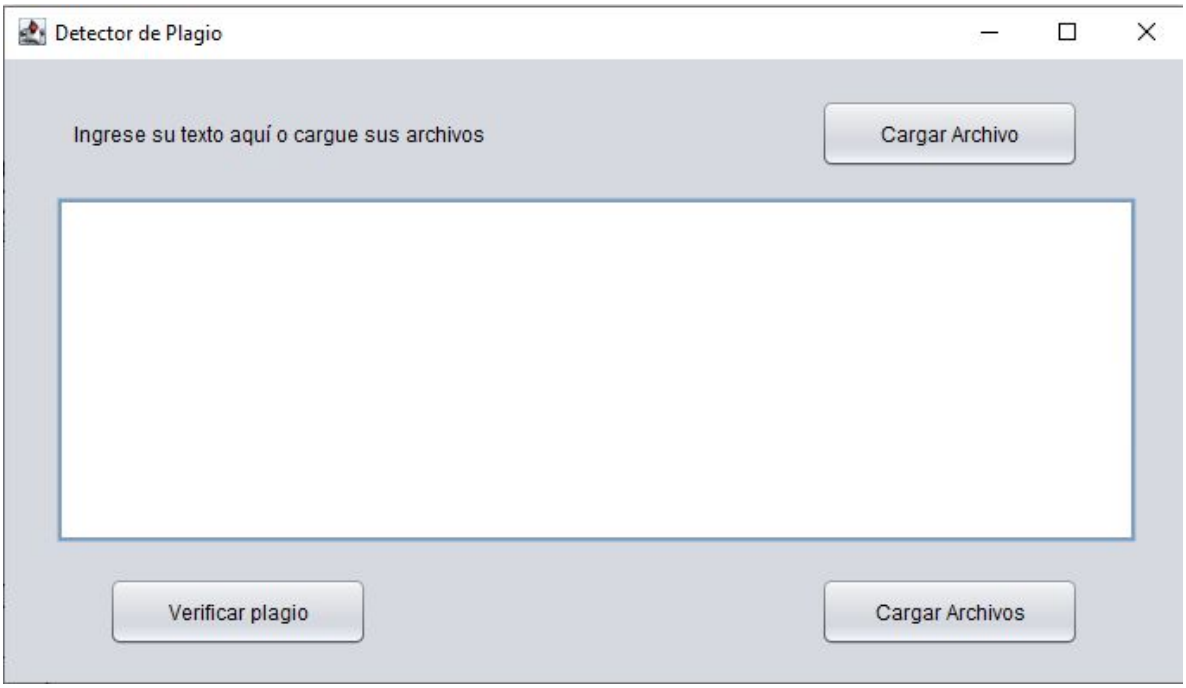


Figura 2: Subimos el archivo original

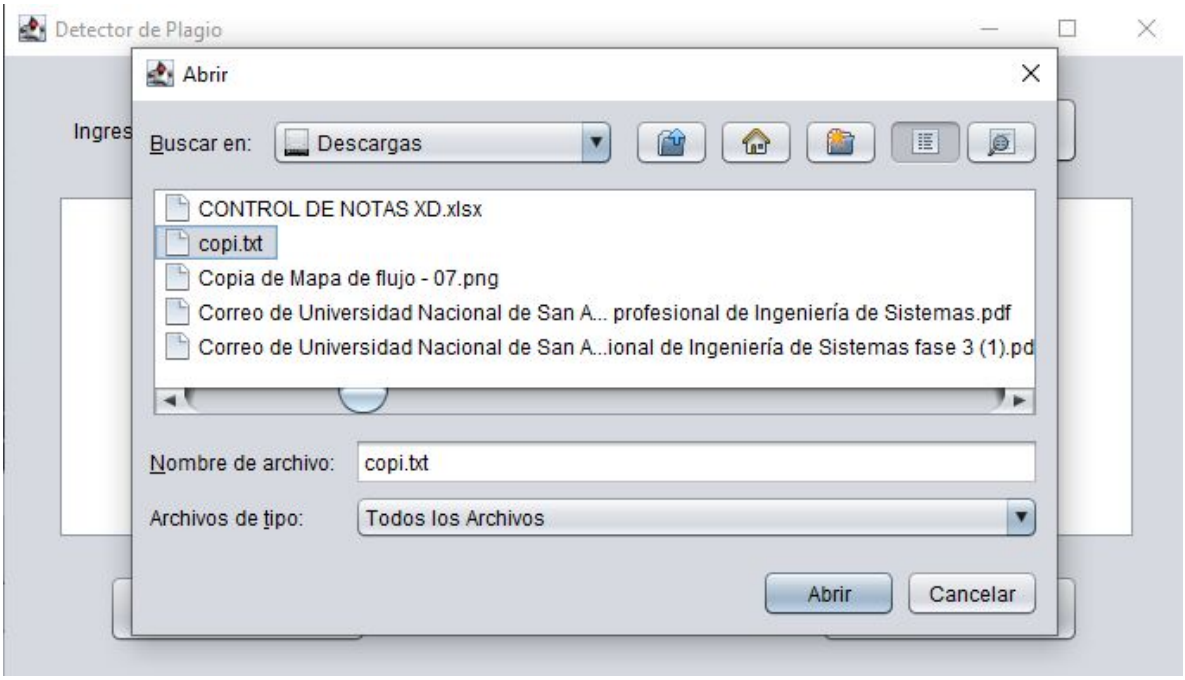


Figura 3: Impresion del texto en el input de texto

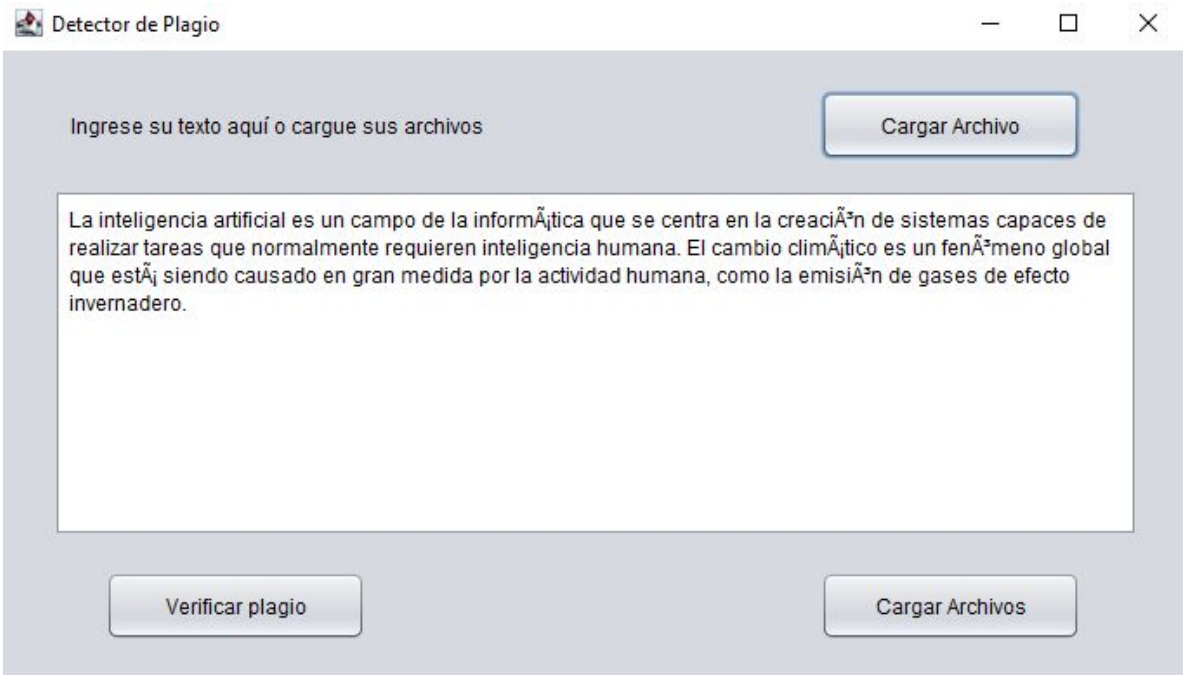


Figura 4: Subimos los archivos que queremos comparar

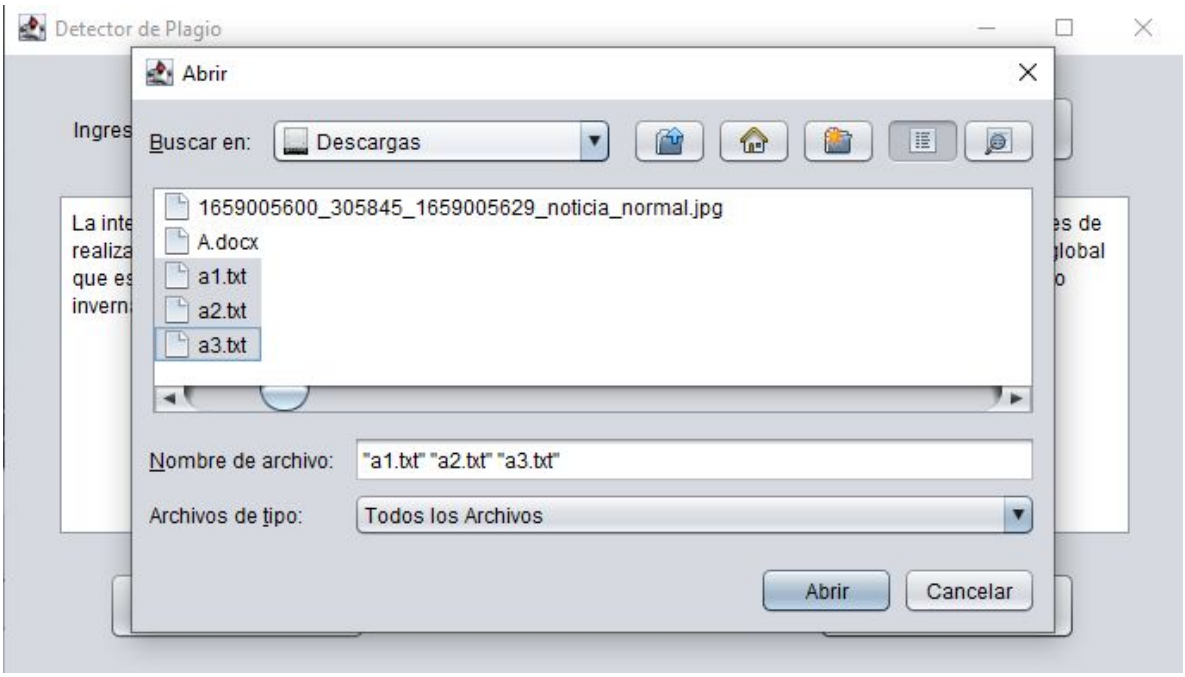
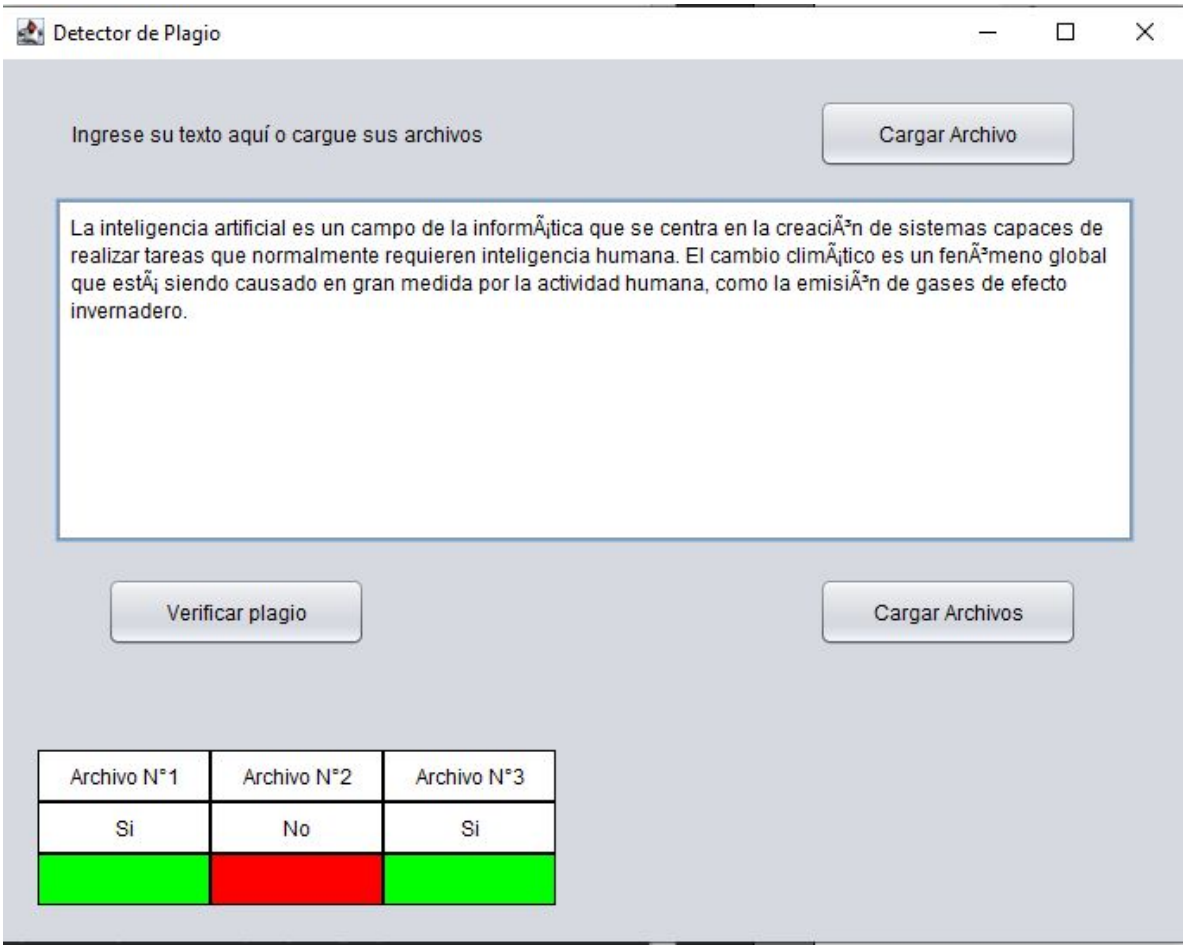


Figura 5: Ejecucion Completa



8. Rúbricas

8.1. Entregable Informe

Tabla 1: Tipo de Informe

Informe	
Latex	El informe está en formato PDF desde Latex, con un formato limpio (buena presentación) y facil de leer.

8.2. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumplio con el ítem correspondiente.
- Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos lo items.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Tabla 2: Niveles de desempeño

	Nivel			
Puntos	Insatisfactorio 25 %	En Proceso 50 %	Satisfactorio 75 %	Sobresaliente 100 %
2.0	0.5	1.0	1.5	2.0
4.0	1.0	2.0	3.0	4.0

Tabla 3: Rúbrica para contenido del Informe y demostración

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X		
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X		
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X		
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X		
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X		
6. Fechas	Las fechas de modificación del código fuente estan dentro de los plazos de fecha de entrega establecidos.	2	X		
7. Ortografía	El documento no muestra errores ortográficos.	2	X		
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4			
Total		20			

9. Referencias

- Presentación TRIE hecha en clase por la Dra. Karim Guevara Puente de la Vega
- `https://www.youtube.com/watch?v=m9zawMC6QAI&ab_channel=ApnaCollege`
- `https://www.educba.com/trie-data-structure-in-java/`