VOL. 1, NO. 14, Agosto 2021



# Detección de plagio entre codigos C++

Johann F. Huaypuna Huanca<sup>1</sup>

<sup>1</sup> Escuela Profesional de Ciencias de la Computación, Universid,ad Nacional de San Agustín de Arequipa, Arequipa, Perú

#### Introducción

oy en día el plagio es abundantemente tratado en la literatura científica como clasificación en tipos de plagio además de diferentes definiciones acerca del plagio, estudios sobre las causas, efectos, consecuencias, posturas morales entre otros como se menciona en (1). En este trabajo nos concentramos en la detección de plagio a nivel de código c++ ya que a diferencia del plagio textual que se da principalmente en documentos, el plagio a nivel de código es un poco mas complicado de detectar ya que al igual que los documentos uno puede hacer plagio a través del parafraseado a nivel código también es posible hacer este parafraseado, cambiando principalmente nombres de variables, números, pero realizando el procedimiento exactamente igual que el código fuente las herramientas textuales no serian de gran ayuda en este escenario ya que no podría detectarse el plagio. En este trabajo se aborda ese problema haciendo uso de técnicas de compiladores principalmente a través del análisis léxico a través de la generación de tokens con sus respectivos tipos y realizando la similitud de los mismos usando técnicas usadas en computación molecular los cuales son usados para detectar similitudes entre cadenas de ADN de proteínas.

# TERMINOLOGÍA Y CONCEPTOS BÁSICOS

#### Análisis Léxico

Es la primera fase de un compilador y consiste en un programa que recibe como entrada el código fuente de otro programa (secuencia de caracteres) y produce una salida compuesta de tokens (componentes léxicos) o símbolos. supongamos que tedemos la expresion:  $int \ a = (b+3*c) - max$ 

TOKEN	TIPO		
int	ENTERO		
a	VARIABLE		
II	IGUAL		
(	APERTURA_PARENTESIS		
В	VARIABLE		
+	OPERADOR		
3	NUMERO		
	:		
max	VARIABLE		

# Algoritmo Needleman-Wunsch

El algoritmo Needleman-Wunsch realiza un alineamiento global de dos secuencias (aquí llamadas j y i). Es comúnmente usado en bioinformática para alinear secuencias de nucleótidos o proteínas. Por ejemplo sea las secuencias a comparar:

Secuencia j: AAG

Secuencia i: AGC

Sea la matriz de sustitución S(i, j) con penalización  $\gamma = -5$ 

$$S(i,j) = \begin{bmatrix} A & C & G & T \\ A & 2 & -7 & -5 & -7 \\ C & -7 & 2 & -7 & -5 \\ G & -5 & -7 & 2 & -7 \\ T & -7 & -5 & -7 & 2 \end{bmatrix}$$
(1)

La definición de puntaje se determina por la siguiente ecuación:

$$S(i,j) = \begin{cases} S(i-1,j-1) + S(x_i, y_j) \\ S(i-1,j) + \gamma \\ S(i,j-1) + \gamma \end{cases}$$
 (2)

Escoger el máximo valor de puntaje luego realizamos la matriz de programación dinámica

		Α	Α	G
	0 -	<b>→</b> -5 →	-10-	-15
Α	-5	2		
G	-10			
С	-15			

$$S(i-1,j-1) + s(x_i, y_j) = 0 + 2 = 2$$

$$S(i-1,j) + \gamma = -5 + (-5) = -10$$

$$S(i,j-1) + \gamma = -5 + (-5) = -10$$

Una vez completada la tabla realizamos un traceback para

obtener el puntaje de alineamiento óptimo

1



Fig. 1: Matriz de programación dinámica completada

## METODOLOGÍA

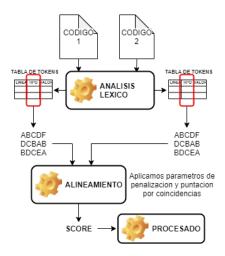


Fig. 2: Metodología

 Lo primero que haremos sera generar la tabla de tokens haciendo el análisis léxico de cada uno de nuestros códigos que evaluaremos.

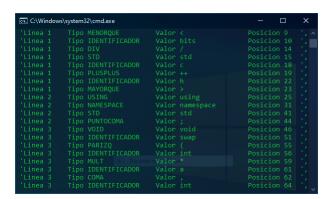


Fig. 3: Tabla de tokens

2. De nuestra tabla de tokens extraemos los tipos y generamos 2 vectores de tipos.



Fig. 4: Vector de tipos

 Antes de realizar el alineamiento renombramos nuestros elementos de nuestro vector de tipos para todas las coincidencias.



Fig. 5: Cambio de variable coincidencias

- 4. Luego realizamos el alineamiento mediante programación dinámica y puntuamos las coincidencias y asu vez penalizamos los gaps y aperturas
- 5. Terminado este proceso, tendremos un *score* el cual representa el numero de coincidencias entre uno y otro lo expresamos en términos de porcentaje para saber cuanta coincidencia hay entre los códigos.

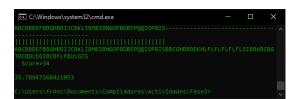


Fig. 6: Similitud

# RESULTADOS

Para la parte de resultados se tomaron en cuenta varios escenarios de prueba y se muestran los resultado de similitud para cada situación ver la siguiente tabla.



ESCENARIO DE COPIA	SIMILITUD
Total	100%
Total con modificación	100%
Una parte de código	52 %
Diferentes códigos	30%

# **CONCLUSIONES**

- La detección de plagio a nivel de código no siempre se puede realizar a nivel textual ya que es posible que la persona que realiza el plagio cambie los nombres de las variables, funciones,etc.
- Como se ha visto en este trabajo se puede realizar mediante programación dinámica a partir de los vectores de tipos dándoles un tratamiento de secuencias.
- Seria posible mejorar este método realizando las comparaciones por ventanas es decir segmentos de códigos.
- Para trabajos futuros se considera trabajar métodos añadidos para mejorar la precisión.

#### VIDEO Y GITHUB

- Video Demostrativo Youtube
- Repositorio Github

### REFERENCIAS

[1] M. F. Abad-García, "El plagio y las revistas depredadoras como amenaza a la integridad científica," *Anales de Pediatría*, vol. 90, no. 1, pp. 57.e1–57.e8, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1695403318305265