Análisis del plagio de código fuente en Java: Detección de similitudes de código en el contexto académico

Leonardo Alvarado, Adriana Millares, Diego Padilla
Tecnológico de Monterrey
Querétaro, México
a01705998@tec.mx, a01705674@tec.mx, a01552594@tec.mx

Planteamiento del problema

El significado literal de plagio es “la práctica de tomar el trabajo o las ideas de otra persona y hacerlas pasar por propias” [6].

Esto también ocurre en el ámbito de software, definiéndose como plagio de software, en donde [5] se refiere a él como desarrollo de un código copiado de manera ilegal del código de otra persona o de alguna fuente de código abierto, para luego disfrazarlo como de autoría original.

El plagio de software ha traído consigo numerosos problemas, principalmente a la economía en ese sector, se encontró en que a nivel mundial el porcentaje de código sin licencia era de un 37%, este porcentaje representa una pérdida de más de 46 mil millones de dólares, tomando en cuenta el valor comercial del software. En el caso de México, los datos registran un porcentaje del 49%, que significa una pérdida de 760 millones de dólares [7].

Entre las principales causas del problema se incluye el incremento de profesionales expertos en el diseño, implementación y funcionamiento de plataformas digitales para abordar los desafíos actuales, en México este sector experimenta un aumento del 7.5% y posee un potencial económico de $147 mil millones de pesos.[8]. En este contexto, algunas organizaciones pueden verse tentadas a plagiar software para obtener una ventaja competitiva, ya sea copiando características o funcionalidades de otros productos exitosos.

De acuerdo con el estudio [9] demuestra que a nivel secundaria y superior identificaron que en promedio a nivel mundial el plagio en los trabajos es del 10.37%.

Los elementos del problema incluyen la falta de conciencia sobre la importancia de la originalidad en la programación y la facilidad de acceso a código fuente en línea.

Actualmente, existen distintas herramientas que analizan el plagio de software, pero no hay ninguna que se centralice en Python, ya que la mayoría de estas son multilenguaje o se enfocan en algún otro [10] por ello el problema específico que se abordará en este contexto es cómo detectar y prevenir el plagio del código fuente en Python.

Este problema es importante porque el plagio del código fuente puede dañar la integridad académica y la calidad de la educación en programación, además de tener un impacto negativo en la industria del software al permitir la distribución de código no autorizado y potencialmente defectuoso.

Marco teórico

El plagio de código fuente en ambientes académicos es un problema de gran relevancia. Las técnicas de modificación de código han ido evolucionando a la par de las herramientas de detección. Esto ha llevado a la necesidad de generar herramientas robustas para la detección de las similitudes de código aunque este haya sufrido modificaciones.

En el primer estudio, se evalúan 11 herramientas de detección de plagio para analizar su capacidad para identificar el plagio en casos donde se realizan modificaciones en el código, para el entrenamiento de estos modelos, se usa la herramienta SimPlag, la cual genera modificaciones en el código similares a las que realiza un estudiante. Existen dos tipos de modificaciones: la transformación, donde se realizan cambios cosméticos o estructurales y la inyección, donde se agrega nuevo código a un programa, el cual puede o no ser funcional.[1]

Entre los problemas identificados se encuentran la falta de disponibilidad de herramientas, la falta de robustez en su funcionamiento, la dificultad para detectar las modificaciones realizadas por los estudiantes y la complejidad de obtener conjuntos de datos reales, debido a que esto atenta a la privacidad de los propietarios del código. Los resultados muestran que los enfoques estructurados

son más efectivos y que las herramientas JPlag y Plaggie son las más robustas en este contexto.[1]

En el segundo estudio, se aborda la problemática de la detección del plagio entre diferentes lenguajes de código, lo cual representa un problema tanto en el ámbito académico como en la industria. Identificar la similitud lógica en programas escritos en diferentes lenguajes de programación plantea dificultades debido a las diferencias en la sintaxis. Además, se presentan otros desafíos como la adición u omisión de comentarios, el renombramiento de métodos o variables, y el cambio de estructuras de datos. Como resultado, se resalta la necesidad de contar con herramientas de detección de similitudes de código más robustas, especialmente para aquellos escritos en diferentes lenguajes de programación.[2]

En la siguiente investigación, se busca detectar plagio en códigos, específicamente en estudiantes, debido a que consideran que al realizar esta práctica, los alumnos no aprenden o al menos no de la misma manera. Se busca encontrar el código base del cual los alumnos se basan o copian, usando diferentes herramientas, como Decision Tree, REP Tree, Random Forest, Neural Network, K-nearest Neighbor, y Naïve Bayes y compara los resultados obtenidos para seleccionar el mejor modelo.[3]

Tras la creación y el entrenamiento de los modelos, se llegó a la conclusión de que los que tenía mejores resultados fueron los árboles de decisión usando J48 y el REP tree, los cuales tuvieron una precisión de 82.61%[3]

Finalmente, durante otra investigación, se usó la herramienta CodEX, la cual es un motor de búsqueda de código fuente que permite a los usuarios buscar un repositorio de fragmentos de código fuente, utilizando pedazos de código fuente como consulta. Esta herramienta nos da la apertura de uso en ámbitos académicos para la detección de plagio, utiliza Árboles de Sintaxis Abstracta (AST) con los cuales, representa archivos de código fuente y esto, se combina con el análisis de similitudes de los fragmentos presentados, permitiendo a los usuarios buscar fragmentos de código fuente sospechosos de plagio. El sistema se evalúa haciendo

uso de casos de modificación del código, con un 95% de casos de prueba identificados con éxito. [4]

Estudio	Enfoque	Lenguaje	Herramientas utilizadas	Resultados
1	Estructurado	Java	SimPlag, JPlag, Plaggie	JPlag y Plaggie son las herramientas más robustas
2	Multi lenguaje	Multi lenguaje	No se especifica.	Las herramientas más robustas son mejores para detectar el plagio entre lenguajes
3	Aprendizaje automático	Multi lenguaje	Decision Tree, REP Tree, Random Forest, Neural Network, K-nearest Neighbor, Naïve Bayes.	Los árboles de decisión (J48 y REP Tree) obtuvieron los mejores resultados con una precisión del 82.61%.
4	Árboles de Sintaxis Abstracta	Muti lenguaje	CodEX	El sistema basado en CodEX tuvo un 95% de casos de prueba identificados con éxito.

Tabla 1: Comparación entre estudios revisados.

En los estudios analizados se exploran distintos enfoques de detección de plagio de código fuente, incluyendo sus limitaciones y retos. Las herramientas expuestas muestran que a pesar de las modificaciones en el código fuente, se puede llevar a cabo un análisis de similitud para la detección del plagio. Dentro de las herramientas destacadas se encuentran codEX, JPlag y Plaggie.

El éxito de la detección de plagio se da al tener un modelo robusto, capaz de detectar las modificaciones en el texto. En los distintos estudios se vieron enfoques

a distintos lenguajes de programación, pero en el caso del estudio 1 este se enfoca en java. Hacer uso de herramientas como SimPlag para la ampliación de un conjunto de datos es fundamental para asegurar que el modelo es eficiente.

Para medir la eficiencia de un modelo es necesario usar métricas. Se usarán las siguientes métricas:

- Precisión: Mide la proporción de casos positivos correctamente identificados sobre el total de casos clasificados como positivos. Se calcula como:
  $Precisión = (Verdaderos\ positivos) / (Verdaderos\ positivos + falsos\ positivos)$

- Sensibilidad (Recall): Mide la proporción de casos positivos correctamente identificados sobre el total de casos positivos reales. Se calcula como:
  $Sensibilidad = (Verdaderos\ positivos) / (Verdaderos\ positivos + Falsos\ negativos)$

- Especificidad: Mide la proporción de casos negativos correctamente identificados sobre el total de casos negativos reales. Se calcula como:
  $Especificidad = (Verdaderos\ negativos) / (Verdaderos\ negativos + Falsos\ positivos)$

- Exactitud (Accuracy): Mide la proporción de casos correctamente clasificados sobre el total de casos. Se calcula como:

  $Exactitud = (Verdaderos\ positivos + Verdaderos\ negativos) / (Total\ de\ casos)$

- Valor F1: Combina la precisión y la sensibilidad en una única métrica que equilibra ambas medidas. Se calcula como:
  $Valor\ F1 = 2 * ((Precisión * Sensibilidad) / (Precisión + Sensibilidad))$

Objetivos

- Objetivo general
  - Desarrollar un modelo de predicción basado en un enfoque estructural, para la detección de plagio en código

- Objetivo específico
    - Detectar plagio en el código de los estudiantes de primeros cursos en el lenguaje de java en un 60%

Referencias

[1] H. Cheers, Y. Lin, and S. P. Smith, "*Evaluating the robustness of source code plagiarism detection tools to pervasive plagiarism-hiding modifications*" *Empirical Software Engineering*, vol. 26, no. 5, Jun. 2021.

[2] M. Agrawal, DK. Sharma, "*A State of Art on Source Code Plagiarism Detection*". 2016 2nd International Conference on Next Generation Computing Technologies (NGCT-2016) Dehradun, India 14-16 October 2016.

[3] C. Saoban, S. Rimcharoen, *"Identifying an original copy of the source codes in programming assignmen*ts". 2019 16th international joint conference on computer science and software engineering (jcsse 2019). 2019

[4] M. Zheng, X. Pan, D. Lillis, *"CodEX: Source Code Plagiarism DetectionBased on Abstract Syntax Trees",* 17th International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP) / 3rd International Conference on Artificial Intelligence and Internet of Things (AIoT 2022). 2022

[5] F. Ullah, J. Wang, M. Farhan, M. Habib, S. Khalid, "*Software plagiarism detection in multiprogramming languages using machine learning approach*". Concurrency and computation-practice & experience: Wiley111 River ST, Hoboken 07030-5774, NJ August 2018

[6] Cambridge Dictionary, "plagiarism," @CambridgeWords, May 17, 2023. https://dictionary.cambridge.org/es/diccionario/ingles-espanol/plagiarism (accessed May 17, 2023).

[7] SN, "Gestión de software: obligación de seguridad, oportunidad de negocios". *The software Alliance BSA*, June 2018.

[8] A. Vargas. (2023, Jan 24).” Servicios TI aumenta contribución al crecimiento para 2023”(1st ed.) [Online]. Available: https://selectnet.selectestrategia.net/reporte/servicios-ti-aumenta-contribucion-al-crecimiento-para-2023/

[9] D. Marquez, P. Melgar. *Integridad académica y plagio*. Ciudad de México: Universidad Nacional Autónoma de México, 2020. pp. 49

[10] A. Ahadi and L. Mathieson. “A Comparison of Three Popular Source code Similarity Tools for Detecting Student Plagiarism”. In *Proceedings of the Twenty-First Australasian Computing Education Conference (ACE '19)*. Association for Computing Machinery, New York, NY, USA, 112–117, Jan 2019. https://doi.org/10.1145/3286960.3286974