Detección de plagio y cálculo del porcentaje de similitud de un código fuente escrito en Java

Rafael Hinojosa López, Keyuan Zhao, Enrique Santos Fraire
A01705777, A01366831, A01705746
Tecnológico de Monterrey

Planteamiento de problema

Según la RAE (Real Academia Española), el plagio se define como la copia de obras ajenas y apropiarse de ellas. En el ámbito académico se considera como una mala práctica que compromete la honestidad y la integridad académica [3]. En los estudios de plagio académicos que se hicieron dentro de las universidades públicas y privadas de siete países de América Latina (Argentina, Brasil, Colombia, Costa Rica, México, Perú y Puerto Rico), reportan que a partir de los resultados de encuestas, más del 70% de los estudiantes admite haber incurrido en prácticas deshonestas a lo largo de su vida estudiantil [4]. Las razones por las cuales los estudiantes llegan a cometer plagio es por la ignorancia de saber citar referencias, mala gestión del tiempo, pereza del estudiante, el deseo de tener buenas calificaciones, presión, falta de conocimiento, entre otras razones que un estudiante puede tener para facilitar cumplir con su deber académico [5].

El plagio de software se considera como utilizar el código fuente de otra persona sin su autorización y adjudicarse como propio [1]. En el mismo sentido, también es considerado como plagio de software las siguientes acciones: copiar e instalar el software para la distribución, descargar y usar el software sin contar la licencia, adquirir el software restringido y destinarlo para uso comercial.

Según el General Report on Software Piracy publicado en BSA (Business Software Alliance), el plagio de software causó pérdidas económicas cerca de 46.3 mil millones de dólares en todo el mundo en 2019. Además, otro dato importante que debemos considerar es que la tasa promedio global sobre el plagio de software en

2019 fue de 39%, lo cual significa que el 39% de los software instalados son plagiados [2].

Uno de los casos estudiados sobre el plagio de software fue el año 2012, en el cual dos informáticos plagiaron un programa de su anterior empresa, en cuyo desarrollo trabajaron, para venderlo a un bajo precio. El resultado de este caso fue la condena de un año de prisión, una multa de €2160 euros y una indemnización de €9000 euros a estos dos individuos [6]. En este caso podemos notar algunas de las consecuencias que conlleva el plagio de software, no por ser parte del equipo desarrollador significa que podemos apropiarnos del código para el uso inapropiado de él.

Teniendo conocimiento sobre el impacto que genera el plagio de software, muchos expertos ya han comenzado a crear soluciones para frenar esta problemática. En el ámbito académico existen herramientas de detección de plagio, tales como Turnitin, Plag.es, Viper, WCopyfind, JPlag, entre otros que detectan la similitud entre trabajos de los estudiantes comparando con diferentes bases de datos (internet, información de la propia universidad, etc.) y generando un informe con el porcentaje de similitudes [7].

Ahora, teniendo en cuenta tales antecedentes, nos planteamos la interrogante: Si utilizamos un sistema de detección de plagio en códigos fuentes escritos en Java, ¿será posible evaluar el porcentaje aproximado de plagio en dichos archivos?

Marco teórico

Con el fin de consultar el estado del arte, hemos investigado trabajos y artículos que se enfocan en la detección de plagio en programas e incluso la detección de múltiples autores. Estos trabajos contribuyeron a delimitar nuestro proyecto, identificar aportes potenciales y objetivos. Además, sentaron las bases para evaluar y analizar las diferentes herramientas y algoritmos computacionales a disposición para lograr nuestros objetivos.

Primeramente, decidimos evaluar el panorama general partiendo del trabajo de Agrawal y Kumar Sharma con su investigación del estado del arte en la detección de plagio en códigos fuente. En esta obra clasificaron los diversos métodos y algoritmos utilizados para la detección de plagio de código fuente, basándose en artículos científicos de otros autores. Entre ellas destacan Machine Learning, Natural Language Processing, la utilización del algoritmo Karp-Rabin Greedy-String-Tiling [15] y el de Winnowing, el modelo *Character N-Grams Comparison*, Minería de Datos [8]. Algunos de estos métodos y algoritmos son utilizados en los siguiente artículos y fueron una base para indagar en el tema de plagio de código fuente.

Comenzando por el trabajo de Bandara y Wijayarathna (2011), proponen el uso de técnicas de Machine Learning para encontrar similitudes en código fuente. Es un sistema basado en una técnica de contado de atributos que utiliza tres algoritmos de aprendizaje para entrenar su modelo: Naïve Bayes Classifier, k-Nearest Neighbor (kNN) y AdaBoost Meta-learning, utilizado como algoritmo de meta-aprendizaje para combinar y complementar los tres algoritmos de Machine Learning utilizados. Su trabajo se enfocó en detectar plagio de archivos de Java, logrando resultados de un 86.6397% de precisión con una muestra de 741 archivos, y 74.1379% con una muestra de 58 archivos. Entre las debilidades encontradas en el modelo destacamos que para una muestra pequeña de datos (58 archivos), la precisión es de un 12.5% menos que cuando se tiene una muestra grande (741 archivos). Además, el algoritmo de Naïve Bayes y el de kNN necesitan del algoritmo AdaBoost para trabajar juntos y óptimamente. [9]

Continuamos por el camino de Machine Learning adentrándonos en el Deep Learning, esta vez dirigido a las redes neuronales, donde nos encontramos con el artículo de Wu y Wang (2021), donde utilizan redes profundas para obtener la similitud entre códigos. El método se basa en la vectorización de código, entrenamiento del modelo utilizando una red siamesa convolucional y el cálculo de la distancia del coseno para obtener la similitud. Cabe mencionar que su trabajo es utilizado para código escrito en Java, lo cual es una fortaleza, pues funcionará como línea base para comparar nuestro trabajo. Finalmente, muestra mejores resultados que aquellos utilizados con word2vec y doc2vec en cuestión de precisión, exhaustividad y valor-F*.* [10]

Adicionalmente, encontramos un acercamiento más simplificado en la investigación de Engels, Lakshmann y Craig (2007), donde se enfocan en identificar plagio de trabajos de estudiantes de primer año de universidad. Utilizan redes neuronales que dan como resultado si un trabajo contiene plagio o no (1 o 0). Para optimizar su solución emplearon el método Quasi-Newton y *boosting* para obtener una mejor precisión en el resultado. La diferencia con respecto a nuestra propuesta es que nosotros deseamos calcular el porcentaje de plagio que contiene un archivo, no solo si es o no plagiado. [16]

Considerando las ventajas que nos proporcionaron los modelos de Deep Learning, el uso de una red neuronal como modelo reafirma su utilidad al detectar segmentos de código con sus autores correspondientes por parte de Abuhamad, Abuhmed, Nyang y Mohaisen (2020), siendo esta un área bastante específica. Para ello utilizaron “Multi-x”, el cual está basado en RNN para identificar múltiples autores de un archivo, identificando a los autores línea por línea en el código fuente, logrando una precisión del 86.41%. A pesar de que su enfoque no es completamente hacia el plagio, sino a la detección de múltiples autores, podemos sacar provecho de sus investigaciones para re-enfocarlas a nuestro proyecto. [12]

En el mismo sentido, otro proyecto que realizó el equipo de Omi, Hossain, Islam y Mittra (2021) con el uso de redes neuronales fue al combinar DNN, SVM y LSTM para detectar múltiples autores en un código fuente. Además, usaron el modelo de code2seq sobre Abstract Syntax Tree para identificar los autores segmentos del

código fuente con más precisión. Para este trabajo usaron 3021 códigos fuente escritos en Java de 40 distintos autores de Github, llegando a una precisión del 96.7% [11], representando así una mejora sustancial del 10.29% con respecto al trabajo anterior.

Cabe destacar que la gran desventaja de detectar a un autor en específico dividido en cada segmento de código, es que se debe usar gran cantidad de códigos con sus autores respectivamente para entrenar el modelo [11] [12].

Finalmente, encontramos una de las obras más completas con el uso de Machine Learning con Ullah, Wang, Farhan, Habib y Khalid (2021). Su trabajo está enfocado en la detección de plagio en códigos escritos en diferentes lenguajes de programación utilizando entropía y tokenización para lograrlo. De igual manera, utiliza el algoritmo PCA y una multi regresión linear (MLR) para detectar el plagio. Luego de evaluar precisión, exhaustividad, tasa TP, tasa FP y mediciones F, se aplica un test z de dos colas para evaluar el modelo. La principal fortaleza de esta herramienta es que funciona para cinco lenguajes de programación (C, C++, Java, C#, Python), llegando a detectar correctamente entre el 73% y el 86% del plagio de código fuente [13], siendo muy parecido al de Bandara y Wijayarathna (75% - 86%), discutido anteriormente. La diferencia que encontramos entre este proyecto y nuestro objetivo es que este está enfocado en varios lenguajes de programación, mientras que el nuestro, en Java.

Consideramos que todos estos trabajos cuentan con fortalezas y debilidades que nos ayudarán a sentar base para identificar aportes potenciales y objetivos en nuestro proyecto. Concluimos que tenemos un amplio número de herramientas identificadas tanto de algoritmos para obtener similitudes, como de Machine Learning, Deep Learning, tokenización, entre otras. Cada una de estas fuentes está relacionada con nuestro objetivo, el cual es detectar plagio en código fuente de Java y serán de ayuda para cumplir exitosamente con nuestro proyecto.

Las fortalezas más relevantes encontradas en los trabajos de investigación y que nos ayudarán a medir nuestra solución son:

- El trabajo de Bandara y Wijayarathna (2011), el cual está enfocado en detectar plagio en archivos escritos en Java, así como el nuestro, y obtuvieron una precisión del 86.6397% con 741 archivos, y 74.1379% con una muestra de 58 archivos. Estas métricas nos servirán para compararlas con la precisión obtenida de nuestra solución.
- El trabajo de Wu y Wang (2021) cuenta con la fortaleza que su trabajo está enfocado en archivos escritos en Java y utilizan métodos diferentes a los empleados por Bandara y Wijayarathna (2011), además, implementan otras métricas a parte de precisión, como exhaustividad y valor-F. Esto nos permitirá tener más resultados y métricas para comparar con los resultados de nuestro trabajo.
- Como última fortaleza se encuentra el trabajo de Ullah, Wang, Farhan, Habib y Khalid (2021), enfocándonos a los resultados que ellos obtuvieron para detectar plagio en código de Java.

Estos artículos nos brindan una serie de métricas para comparar y evaluar el desempeño de nuestros futuros modelos, así como medir con exactitud el éxito de nuestros objetivos.

Posteriormente se explorarán a profundidad estas herramientas, métodos y algoritmos para decidir cuáles son los más aptos tanto para el set de datos como para su evaluación.

Objetivos

A partir de la descripción sobre la problemática que estamos enfrentando, definimos como objetivo para nuestro proyecto **determinar si un par de códigos fuente escritos en Java contienen plagio o no utilizando métodos matemáticos para determinar su similitud como primera aproximación, para posteriormente refinar dicha predicción con el uso de Machine Learning y buscar resultados más precisos.**

Asegurando que nuestro objetivo principal sea exitoso, nuestros objetivos específicos son:

- **Determinar el porcentaje de plagio que contiene un código de Java y posteriormente dictaminar un veredicto, evitando caer en falsos positivos (detectado como PLAGIO cuando NO lo es) al elevar el porcentaje de similitud a un 85% para considerar el par de códigos como plagiados.**
- **Identificar los segmentos plagiados de los códigos.**

Bibliografía

[1] C. Lee. (2021, Jun 26). “¿Qué es el plagio de código fuente y por qué se da cada vez más?” [Online]. Available: https://www.turnitin.com/es/blog/que-es-el-plagio-de-codigo-fuente-y-por-que-se-da-cada-vez-mas

[2] "Homepage | BSA | The Software Alliance" [Online]. Homepage | BSA | The Software Alliance. Available: https://www.bsa.org/

[3] "Plagio" [Online]. Mondragon Unibertsitatea. Available: https://www.mondragon.edu/es/web/biblioteka/plagio

[4] M. D. Morales and I. Lujano, “Entre la Integridad Académica y el Plagio Estudiantil ¿Qué Dicen las Universidades Públicas Mexicanas en su Normatividad”, *Archivos Analíticos de Políticas Eductativas*, vol. 29, no. 166, Sep 2021.

[5] Paperrater. (2014, Feb 24). “研究生学术剽窃的原因与遏制” [Online]. Available: http://www.paperrater.net/20140224/2994.html

[6] Santander. (2012 Jun 11). “Un año de prisión para dos informáticos por plagiar un programa” [Online]. Available: https://www.eldiariomontanes.es/v/20120611/cantabria/tribunales/prision-para-informaticos-plagiar-20120611.html

[7] “Plagio: Programas para la detección del Plagio”. Biblioguías Deusto LibGidak [Online]. Available: https://biblioguias.biblioteca.deusto.es/c.php?g=208480&p=1461352

[8] M. Agrawal and D. Kumar, “A State of Art on Source Code Plagiarism Detection”, *NGCT-2016*, IEEE, pp. 236-241, Oct 2016.

[9] U. Bandara and G. Wijayarathna, “A Machine Learning Based Tool for Source Code Plagiarism Detection”, *International Journal of Machine Learning and Computing*, vol. 1, no. 4, Oct 2011.

[10] W. Yi and W. Wei, "Code Similarity Detection Based on Siamese Network", *2021 IEEE International Conference on Information Communication and Software Engineering (ICICSE)*, pp. 47-51, Apr 2021.

[11] A. M. Omi, M. Hossain, M. N. Islam and T. Mittra, "Multiple Authors Identification from Source Code Using Deep Learning Model," *2021 International Conference*

on Electronics, Communications and Information Technology (ICECIT), pp. 1-4, Dec 2021.

[12] M. Abuhamad, T. Abuhmed, D. Nyang and D. Mohaisen, "Multi-χ: Identifying Multiple Authors from Source Code Files", *Proc. Privacy Enhancing Technol.*, vol. 2020, no. 3, pp. 25-41, Jun 2020.

[13] F. Ullah, J. Wang, M. Farhan, M. Habib and S. Khalid, “Software plagiarism detection in multiprogramming languages using machine learning approach” *Concurrency Computat Pract Exper*, vol. 33, no. 4, Feb 2021.

[14] A. M. Bejarano, L. E. García and E. E. Zurek, “Detection of Source Code Similitude in Academic Environment”, *Comput. Appl. Eng. Educ.*, vol. 23, no. 1, pp. 13-22, Dec 2012.

[15] M. J. Wise, “String Similarity via Greedy String Tiling and Running Karp-Rabin Matching”, Dec 1993.

[16] S. Engels, V. Lakshmanan and C. Michelle. “Plagiarism detection using feature-based neural networks.” *ACM Sigcse Bulletin*, 2007.