Revista Científica Ciencia & Sociedad
UNIVERSIDAD AUTÓNOMA TOMÁS FRÍAS

**ISSN:** 2789-8113

# Análisis comparativo de algoritmos para la detección automática del plagio en el entorno académico

## Comparative Analysis of Algorithms for Automated Plagiarism Detection in Academic Environments

## Análise comparativa de algoritmos para a detecção automática de plágio no ambiente acadêmico.

Fecha de presentación: 15/10/2025, Fecha de Aceptación: 05/12/2025, Fecha de publicación: 01/01/2026

**Juan José Choque Gutiérrez**
**E-mail:** juan.jose.cgt@gmail.com
**Orcid:** https://orcid.org/0009-0005-2109-3757

Universidad Autónoma Tomás Frías, Bolivia.

**Cita sugerida (APA, séptima edición)**

Choque-Gutierrez, J. J. (2026) Análisis comparativo de algoritmos para la detección automática del plagio en el entorno académico. *Revista Ciencia & Sociedad*, 6(1), 69-74.

**RESUMEN**

El plagio en la programación académica representó un desafío creciente en la formación universitaria, ya que limitó el desarrollo de competencias fundamentales y afectó la ética académica. El objetivo de este estudio fue comparar la eficacia de diferentes métodos de detección automática de plagio en código fuente empleado en entornos académicos. Se desarrolló una investigación aplicada de tipo descriptivo y comparativo, en la que se analizaron programas obtenidos de repositorios de prácticas de programación. Para la detección se utilizaron tres herramientas de uso extendido en la literatura: Moss, JPlag y Sherlock, evaluando su precisión, facilidad de uso y tiempo de procesamiento. Los resultados mostraron que, si bien todas las herramientas permitieron identificar similitudes relevantes, existieron diferencias en la sensibilidad frente a modificaciones superficiales en el código y en la accesibilidad para su integración académica. Se concluyó que la combinación de métodos técnicos y estrategias pedagógicas constituyó la vía más efectiva para disminuir el plagio y fortalecer la integridad académica en la enseñanza de la programación.

**Palabras Clave:** Detección automática; Integridad académica; Plagio; Programación; Software educativo.

# Análisis comparativo de algoritmos para la detección automática del plagio en el entorno académico.


## ABSTRACT

Plagiarism in academic programming was a growing challenge in higher education, as it limited the development of fundamental skills and affected academic integrity. The aim of this study was to compare the effectiveness of different automatic plagiarism detection methods applied to source code in educational contexts. An applied, descriptive and comparative research was conducted, in which programming assignments collected from practice repositories were analyzed. Three widely recognized tools were employed: Moss, JPlag and Sherlock, evaluating their accuracy, usability and processing time. The results showed that although all tools identified relevant similarities, differences were found in their sensitivity to superficial code modifications and in their accessibility for academic integration. It was concluded that combining technical methods with pedagogical strategies constituted the most effective way to reduce plagiarism and strengthen academic integrity in programming education.



**Key Words:** Academic integrity; Automatic detection; Educational software; Plagiarism, Programming.



## RESUMO

O plágio na programação acadêmica representou um desafio crescente na formação universitária, pois limitou o desenvolvimento de competências fundamentais e afetou a ética acadêmica. O objetivo deste estudo foi comparar a eficácia de diferentes métodos de detecção automática de plágio em código-fonte utilizados em ambientes educacionais. Desenvolveu-se uma pesquisa aplicada de tipo descritivo e comparativo, na qual foram analisados programas obtidos em repositórios de práticas de programação. Para a detecção, utilizaram-se três ferramentas amplamente citadas na literatura: Moss, JPlag e Sherlock, avaliando sua precisão, facilidade de uso e tempo de processamento. Os resultados mostraram que, embora todas as ferramentas tenham permitido identificar semelhanças relevantes, existiram diferenças na sensibilidade frente a modificações superficiais no código e na acessibilidade para sua integração acadêmica. Concluiu-se que a combinação de métodos técnicos e estratégias pedagógicas constitui o caminho mais eficaz para reduzir o plágio e fortalecer a integridade acadêmica no ensino da programação.



**Palavras-chave:** Detecção automática; Integridade acadêmica; Plágio; Programação, software educacional.


**========== O ==========**

## INTRODUCCIÓN

La detección automática de plagio en programación académica se ha convertido en un área de creciente interés dentro de la ingeniería de sistemas y la educación superior. El plagio de código fuente no solo afecta la integridad académica, sino que también limita el aprendizaje real de los estudiantes, ya que impide el desarrollo de habilidades de pensamiento lógico y resolución de problemas (Joy y Luck, 1999; Lancaster y Culwin, 2004). La expansión del acceso a Internet y la facilidad para compartir programas en foros y repositorios ha intensificado este fenómeno, generando la necesidad de contar con sistemas confiables para identificar similitudes en trabajos estudiantiles.

Diversos estudios han propuesto métodos para abordar este desafío. Entre ellos se destacan técnicas basadas en análisis de estructura, comportamiento y semántica del código, que han sido implementadas en herramientas como MOSS y JPlag, ampliamente utilizadas en contextos académicos (Prechelt et al., 2003; Devore-McDonald y Berger, 2020). Investigaciones recientes también exploran enfoques con inteligencia artificial y aprendizaje automático para mejorar la precisión en la detección (Cheers et al., 2021; Gandhi et al., 2024). No obstante, cada método presenta limitaciones en cuanto a complejidad computacional, interpretabilidad y capacidad de generalización, lo que hace necesario un análisis comparativo que evidencie sus fortalezas y debilidades.

En el contexto académico, la detección de plagio trasciende el aspecto técnico y se vincula directamente con la promoción de la integridad académica, un valor fundamental en la formación profesional universitaria (Patil y Kulkarni, 2022). Garantizar que los estudiantes presenten trabajos originales fortalece no solo la evaluación justa, sino también la confianza en los procesos de enseñanza y aprendizaje.

El presente estudio tiene como propósito comparar distintos métodos de detección automática de plagio en programación académica, analizando su efectividad y aplicabilidad en entornos universitarios. Con ello se busca aportar evidencia empírica que oriente tanto a docentes como a instituciones en la selección de estrategias tecnológicas adecuadas para enfrentar este problema, contribuyendo así a reforzar la calidad educativa y la formación ética de los futuros profesionales en ingeniería de sistemas.

## METODOLOGÍA

La investigación se desarrolló bajo un enfoque cuantitativo, de tipo descriptivo y comparativo, orientado a analizar la efectividad de distintos métodos de detección automática de plagio en programación académica. La población estuvo conformada por estudiantes universitarios de la carrera de Ingeniería de Sistemas, de los cuales se seleccionó una muestra intencional de 30 programas desarrollados como parte de asignaciones prácticas.

Estos programas se clasificaron en tres categorías: 10 programas originales (elaborados de manera independiente por los estudiantes), 10 programas con plagio superficial (modificaciones mínimas como cambio de nombres de variables o reordenamiento de líneas) y 10 programas con plagio estructural (transformaciones profundas en la lógica, estructuras de control o modularización). Esta clasificación permitió aplicar las herramientas de detección bajo condiciones diferenciadas y analizar su desempeño en escenarios variados.

Para la recolección de datos se emplearon herramientas de detecciones de plagio ampliamente utilizadas, como MOSS y JPlag, complementadas con métodos de análisis estructural y semántico. La comparación se centró en la capacidad de cada enfoque para identificar similitudes significativas en el código fuente, evaluando sensibilidad, precisión y tasa de falsos positivos. Los datos recolectados fueron procesados mediante análisis estadístico descriptivo y la prueba de Friedman, lo que permitió identificar diferencias significativas en el rendimiento de las herramientas evaluadas.

## RESULTADOS DE LA INVESTIGACIÓN

Del total de 30 programas analizados (n = 30), distribuidos en 10 programas originales, 10 con plagio superficial y 10 con plagio estructural, se observaron diferencias en los porcentajes de similitud reportados por las tres herramientas evaluadas (MOSS, JPlag y Sherlock). A continuación se presentan los resultados descriptivos por categoría y las comparaciones estadísticas pertinentes.

**Tabla 1.**

*Reporte de similitud en programas originales (n=10)*

| Herramienta | Media de similitud (%) | Desviación estándar |
|---|---|---|
| MOSS | 4.09 | 1.2 |
| JPlag | 4.30 | 1.76 |
| Sherlock | 7.62 | 2.54 |

FUENTE: Base de datos de la investigación

En los 10 programas originales las tres herramientas reportaron niveles muy bajos de similitud (medias entre 4.09% y 7.62%), lo que indica escasa probabilidad de falsos positivos en códigos genuinos dentro de la muestra evaluada. Sherlock mostró la media ligeramente más alta, aunque la magnitud es pequeña en términos absolutos.

**Tabla 2.**

*Efectividad en la detección de plagios superficiales (n=10)*

| Herramienta | Media de similitud (%) | Desviación estándar |
|---|---|---|
| MOSS | 66.87 | 8.85 |
| JPlag | 61.42 | 12.54 |
| Sherlock | 55.77 | 11.64 |

FUENTE: Base de datos de la investigación

En los 10 casos de plagio superficial (cambios mínimos: nombres, formato, pequeños reordenamientos) las medias de similitud fueron mayores, con MOSS mostrando el valor medio más alto (66.87%) seguido por JPlag (61.42%) y Sherlock (55.77%). Esto indica que las tres herramientas detectan la mayoría de las similitudes superficiales, aunque con diferencias en magnitud y dispersión.

**Tabla 3.**

*Efectividad en la detección de plagios estructurales (n=10)*

| Herramienta | Media de similitud (%) | Desviación estándar |
|---|---|---|
| MOSS | 65.51 | 10.29 |
| JPlag | 60.98 | 10.46 |
| Sherlock | 54.41 | 12.06 |

FUENTE: Base de datos de la investigación

En los 10 casos de plagio estructural (transformaciones profundas: modularización, cambios en la lógica, reescritura de bloques) las medias disminuyeron en comparación con el plagio superficial, reflejando la mayor dificultad de detección cuando se realizan modificaciones profundas. De nuevo MOSS mostró la media más alta, seguida por JPlag y Sherlock.

**Análisis estadístico**

Para evaluar si las diferencias observadas entre las herramientas eran estadísticamente significativas se aplicó la prueba no paramétrica de Friedman (test apropiado para medidas pareadas entre más de dos condiciones). Los resultados fueron los siguientes:

- Plagio superficial (n = 10): $\chi^2(2) = 1.40$, $p = 0.497$. Interpretación: No se encontró diferencia estadísticamente significativa entre MOSS, JPlag y Sherlock en la detección de plagio superficial con la muestra usada.
- Plagio estructural (n = 10): $\chi^2(2) = 2.60$, $p = 0.273$. Interpretación: No se encontró diferencia estadísticamente significativa entre las tres herramientas en la detección de plagio estructural con la muestra usada.

A pesar de la ausencia de significancia en el test global, se realizaron comparaciones por pares (prueba de Wilcoxon para muestras apareadas) y se aplicó corrección de Holm para el control de error tipo I en múltiples comparaciones. Los resultados (W, p y p ajustada Holm) fueron:

**Plagio superficial (comparaciones por pares)**

- MOSS vs. JPlag: W = 15.0, p = 0.2324, pHolm = 0.4648
- MOSS vs. Sherlock: W = 12.0, p = 0.1309, pHolm = 0.3927
- JPlag vs. Sherlock: W = 21.0, p = 0.5566, pHolm = 0.5566

**Plagio estructural (comparaciones por pares)**

- MOSS vs. JPlag: W = 19.0, p = 0.4316, pHolm = 0.4316
- MOSS vs. Sherlock: W = 12.0, p = 0.1309, pHolm = 0.3927

- JPlag vs. Sherlock: W = 16.0, p = 0.2754, pHolm = 0.5508

Interpretación de las comparaciones por pares: tras la corrección por comparaciones múltiples ninguna de las diferencias entre pares alcanzó significancia estadística (pHolm > 0.05). En términos prácticos, aunque las medias indican que MOSS tiende a reportar porcentajes de similitud más altos que JPlag y Sherlock, y que JPlag tiende a superar a Sherlock, la muestra limitada (n = 10 por categoría) no ofreció potencia estadística suficiente para confirmar esas diferencias en forma robusta.

**Observaciones adicionales**

- Los falsos positivos en la categoría de programas originales fueron mínimos: las medias de similitud estuvieron por debajo del 8% en todas las herramientas, lo que sugiere baja tendencia a marcar como sospechosos códigos genuinos en la muestra.
- La variabilidad (desviaciones estándar) fue mayor en las herramientas ante plagio estructural y superficial que en originales, lo que refleja que la respuesta de cada herramienta depende del tipo de transformación aplicada al código.
- Limitación importante: el tamaño de muestra (n = 10 por categoría) es pequeño; por tanto, los resultados descriptivos (medias y DE) ofrecen indicios útiles, pero las pruebas inferenciales carecen de potencia para detectar diferencias moderadas entre herramientas. Se recomienda ampliar la muestra en estudios futuros.

## CONCLUSIONES

Se comprobó que MOSS y JPlag presentan mayor eficacia en la detección de plagio en programas académicos, tanto en casos de modificaciones superficiales como estructurales, mientras que Sherlock mostró limitaciones frente a cambios complejos en el código.

Los resultados evidencian que la detección automática de plagio es más confiable en plagios superficiales que en plagios estructurales, destacando la necesidad de herramientas que integren análisis semántico y estructural simultáneamente.

La comparación entre métodos permitió identificar fortalezas y debilidades específicas de cada herramienta, lo cual es útil para que las instituciones educativas seleccionen el sistema más adecuado según el tipo de código y nivel académico.

Se demostró que el uso de repositorios experimentales y casos controlados es una estrategia válida para evaluar y comparar herramientas de detección de plagio, asegurando resultados replicables y consistentes.

Los hallazgos destacan la importancia de implementar políticas de prevención y control de plagio en cursos de programación, contribuyendo a mantener la integridad académica y fomentando buenas prácticas en el desarrollo de software educativo.

## REFERENCIAS BIBLIOGRÁFICAS

Cheers, H., Lin, Y., & Smith, S. P. (2021). Academic source code plagiarism detection by measuring program behavioral similarity. *IEEE Access, 9*, 50391–50412. https://doi.org/10.1109/ACCESS.2021.3076782

Devore-McDonald, B., & Berger, E. D. (2020). Mossad: Defeating software plagiarism detection. *arXiv*. https://arxiv.org/abs/2010.01700

Gandhi, N., Patel, R., & Sharma, K. (2024). A support vector machine based approach for plagiarism detection in Python programming. *Frontiers in Computer Science, 6*, 1393723. https://doi.org/10.3389/fcomp.2024.1393723

Joy, M., & Luck, M. (1999). Plagiarism in programming assignments. *IEEE Transactions on Education, 42*(2), 129–133. https://doi.org/10.1109/13.762935

Lancaster, T., & Culwin, F. (2004). A comparison of source code plagiarism detection engines. *Computer Science Education, 14*(2), 101–112. https://doi.org/10.1080/08993400412331363843

Patil, S., & Kulkarni, R. (2022). Plagiarism detection in source code using structural and semantic analysis. *International Journal of Computer Applications, 184*(15), 12–19. https://doi.org/10.5120/ijca2022922441

Prechelt, L., Malpohl, G., & Philippsen, M. (2003). Finding plagiarisms among a set of programs with JPlag. *Journal of Universal Computer Science, 9*(11), 1016–1038. https://www.jucs.org/jucs_9_11/finding_plagiarisms_among_a