“Herramientas para la detección de plagio de software. Un caso de estudio en trabajos de cátedra”

F. Javier Diaz

Claudia M. Tzancoff Banchoff

Anahí S. Rodríguez

{jdiaz, cbanchoff, anahi}@info.unlp.edu.ar

LINTI. Facultad de Informática, Universidad Nacional de La Plata.

La Plata, 1900, ARGENTINA

ABSTRACT

The advance of technology in these last years, together with easy and fast access to the Internet makes ideas, opinions, theories, graphics, drawings, quotes, digital documents, etc available for their use anytime.

This increases the possibility of plagiarism of these types of materials.

The goal of this article is to analyze and compare tools for the detection of plagiarism, emphasizing on “open source” tools, taking into account those which allow for analysis of the source code to be performed, that is, those oriented to the detection of software copies. To test the performance of the tools we will use the works of the students of the subject “Languages Seminar”, in which they develop applications in C Language.

Key words: Plagiarism, Open Code, Copy detection.

RESUMEN

El avance de la tecnología en estos últimos años, junto al fácil y rápido acceso a Internet hace que ideas, opiniones, teorías, gráficos, dibujos, citas, documentos en formato digital, etc., se encuentren disponibles para su uso en cualquier momento.

Esta situación incrementa la posibilidad de que ocurran casos de plagio de este tipo de materiales.

El objetivo de este artículo es analizar y comparar un conjunto de herramientas "open source" para detección de plagio de software. Es decir, que permitan analizar código fuente y la detección de copias de software.

Para probar el rendimiento y las características de las herramientas se realizaron pruebas con trabajos de alumnos de la cátedra de "Seminario de Lenguajes", en la cual se desarrollan aplicaciones en Lenguaje C.

Palabras Claves: Plagio, Código Abierto, Detección de copia. Programación. Lenguaje C.

INTRODUCCIÓN

El objetivo principal de los docentes es transmitir todos sus conocimientos a sus alumnos, y que los mismos, a partir de ellos, muestren sus propias creaciones, para luego poder evaluar los conocimientos que han adquirido.

La gran disponibilidad de documentos, ideas, imágenes, programas, y todo tipo de material en Internet hace que los docentes estén más alertas y preocupados por la evaluación de los trabajos de sus alumnos.

En este contexto, y priorizando un marco académico, se hace indispensable el uso de herramientas que ayuden al docente a detectar copias y, de esta manera, poder evaluar en forma apropiada los trabajos e ideas de sus alumnos.

Muchas universidades internacionales han adoptado medidas estrictas con respecto a la detección de trabajos plagiados. Algunas de ellas, incluso, se refieren al plagio como causal de expulsión. En este marco, desarrollan campañas antiplagio y en muchas de ellas se han desarrollado herramientas y documentación que sirven como base del presente estudio.

El objetivo de este artículo es analizar y comparar un conjunto de herramientas para la detección de plagio de software. Se hicieron varias pruebas piloto con algoritmos escritos en

Lenguaje C, partiéndose de una caso sencillo como ser el algoritmo de búsqueda binaria obtenido del sitio de la Universidad de las Palmas de Gran Canaria [1] y, como caso piloto en el ámbito académico, se tomaron los trabajos y publicaciones de estudiantes de la cátedra "Seminario de Lenguajes", la cual desarrolla aplicaciones en Lenguaje C.

Se trabajó con varias herramientas, teniendo en cuenta aquellas que permiten analizar código fuente, es decir, aquellas que están orientadas a la detección de copias de software. Es importante aclarar que este análisis es mucho más engorroso que la detección de copias en documentos en texto natural, ya que hay que marcar bien la diferencia entre códigos "copiados" y "similares". El objetivo de un trabajo de una cátedra donde se enseña a programar, es que los alumnos implementen una solución a un problema planteado y, seguramente, muchos de ellos entregarán códigos similares. Por lo tanto, se buscará una herramienta robusta que sea capaz de identificar cada uno de estos casos.

¿Qué se conoce por plagio?

Antes de comenzar con la descripción de herramientas y situaciones, se dará un pequeña definición de lo que significa "plagio".

Según la Wikipedia *(The Free Enciclopedia* - http://en.wikipedia.org/), *"..plagio es reclamar la autoría, o darla a entender, de materiales que uno realmente no ha creado, como propios. Dentro de la academia, el plagio se considera como falta de honradez académica, y es una ofensa académica seria y castigable. El plagio no es necesariamente lo mismo que violar el copyright, lo que ocurre cuando uno viola la ley del copyright. La copia de algunas pocas frases para citarlas está dentro del buen uso del copyright, pero si no se las atribuye al verdadero autor, es plagio".*

*El Oxford English Dictionary Online. 2005 (*http://dictionary.oed.com/*)", define plagio como "La acción o la práctica de plagiar; la apropiación ilícita o el robo, y publicación como propios, de las ideas, o de la expresión de las ideas (literario, artístico, musical, mecánico, etc.) de otros.*

El derecho de autor es un conjunto de normas a través de las cuales se regulan tanto los derechos morales como patrimoniales concedida al autor, por el solo hecho de la creación de una obra literaria, artística o científica, ya sea que la misma ya esté o no publicada [2].

El copyright comprende la parte patrimonial de los derechos de autor, esto es tomado en el derecho anglosajón. Es la protección que se les brindará a los autores incluyendo obras literarias, dramáticas, musicales, artísticas e intelectuales[3].

En Argentina el derecho de autor está establecido en el articulo 17 de la Constitución, establece que: "Todo autor o inventor es propietario exclusivo de su obra, invento o descubrimiento, por el término que le acuerde la ley". A si mismo, la Ley 11723 regula el régimen de Propiedad Intelectual, el artículo 5 de esta ley expresa lo siguiente: "La propiedad intelectual sobre sus obras corresponde a los autores durante su vida y a sus herederos o derechohabientes hasta setenta años contados a partir del 1 de Enero del año siguiente al de la muerte del autor" [2].

El copyright le da al dueño del derecho de autor el derecho exclusivo para hacer y para autorizar a otros a hacer lo siguiente:

- Reproducir la obra en copias o fonogramas;
- Preparar obras derivados basados en la obra;
- Distribuir copias o fonogramas de la obra al público vendiéndolas o haciendo otro tipo de transferencias de propiedad tales como alquilar, arrendar o prestar dichas copias;
- Presentar la obra públicamente, en el caso de obras literarias, musicales, dramáticas y coreográficas, pantomimas, películas y otras producciones audiovisuales;
- Mostrar la obra públicamente, en el caso de obras literarias, musicales, dramáticas coreográficas, pantomimas, obras pictóricas, gráficas y esculturales, incluyendo imágenes individuales de películas u otras producciones audiovisuales;
- En el caso de grabaciones sonoras, interpretar la obra públicamente a través de la transmisión audiodigital.

La utilización indebida de algunos de los puntos anteriores, por parte de algún individuo que no tenga el derecho otorgado por el autor de la obra, estaríamos en un caso de violación del copyright.

Características funcionales a evaluar

Para realizar el análisis de las herramientas se plantearon una serie de características que permitieron evaluar y clasificar cuál es la más apropiada para tomar como base. Estas son:

- La interfaz de usuario, que puede ser por línea de comando o visual, (es decir textual o gráfica).
- La búsqueda de similitudes de código dentro de un mismo archivo, o entre varios archivos fuentes.
- El motor de búsqueda de código copiado puede ejecutarse localmente o puede trabajar en un servidor dedicado para esta tarea en alguna universidad que brinde este servicio, como por ejemplo la Universidades de Karlsruhe de Alemania y la Universidad de Stanford.
- Las similitudes se pueden buscar dentro de un conjunto de archivos locales, o bien la comparación se puede realizar con información publicada en Internet.

A nuestro juicio, una herramienta que se pueda adoptar y utilizar fácilmente en un ámbito educativo debería contar con una interfaz gráfica, una búsqueda de similitudes entre distintos archivos de un directorio local y un motor de búsqueda local.

Técnicas de detección de plagio

El tratamiento realizado por cada una de las herramientas varía dependiendo del método que la misma esté utilizando, Muchas herramientas realizan el trabajo de detección de copias en dos fases. En la primer fase se realiza la "*tokenización*", en la cual se convierte el código fuente en una secuencia de "tokens" o marcas, y por ejemplo varias sentencias que indican un iteración se reemplazarán por el mismo token.

Luego se realiza la etapa de evaluación de estas secuencias de tokens, a la cual se debe seleccionar una buena definición de métricas [4] [5].

La propuesta adoptada por Fintan Culwin [6] define varios criterios en los cuales se pueden caracterizar luego las métricas utilizadas, los cuales son:

- *Servidor – Escritorio*: Según el sistema requiera infraestructura Cliente – Servidor o no.
- *Local – Remoto*: Según el sistema requiera una estructura en red o no.
- *Basado en documentos – Basado en corpus*: Según el sistema se aplique a un documento o a un conjunto de ellos.
- *Intracorpus – Intercorpus*: Aplicado a un sistema *basado en corpus*, determina si el sistema sólo utiliza información interna al corpus o no.
- *Gratuito – Comercial:* Según el sistema es gratuito o no.
- *Orientado a base de datos – No orientado a base de datos:* Según el sistema utilice bases de datos o no.
- *Documentos con estilo – Texto plano:* Según el documento tenga algún tipo de estilo o sea texto plano.
- *Código fuente abierto – Código fuente reservado*: Según el código fuente sea libre o no.
- *Multilingüe – Monolingüe:* Según el sistema use técnicas específicas de varios lenguajes o no.

Las métricas utilizadas en herramientas para detección de plagio pueden ser derivadas de métricas utilizadas para medir el rendimiento en un software, teniendo en cuenta los siguientes puntos:

- El número de error en el programa
- El tiempo requerido por el programa
- El tiempo para depurar el programa
- El algoritmo puro de un programa

Herramientas analizadas

Las herramientas que se analizaron en el presente trabajo son las siguientes: Sherlock, Simian, Tester SIM y Jplag. A continuación daremos una breve explicación sobre las mismas.

***Sherlock*[7] :**

Es una herramienta de código abierto, que trabaja con código escrito en los lenguajes Java, C y texto natural. Esta herramienta no cuenta con una interfaz gráfica. Fue desarrollada por la Universidad de Sydney.

Los resultados arrojados se basan en un porcentaje que se corresponde con las similitudes encontradas. El porcentaje "0%" significa que no hay similitudes, y "100%" significa que hay muchas posibilidades de que tengan partes iguales. Al no utilizar el documento en su totalidad, no se puede afirmar que sean completamente iguales. Sólo trabaja con archivos locales, no busca similitudes en Internet.

***Simian*[8]**

Esta una herramienta no es de código abierto. Identifica la duplicación de códigos escritos en JAVA, C#, C, C++, COBOL, Ruby, JSP, ASP, HTML, XML, Visual Basic y texto natural.

Fue desarrollada por una consultora de Australia llamada REDHILL. No cuenta con una interfaz gráfica, por lo tanto, el ingreso de los parámetros es a través de línea de comando.

Sólo trabaja con archivos locales, no busca similitudes en Internet.

***Jplag*[9]**

Esta herramienta no es de código abierto, sólo permite la programación del cliente, pero el servidor es el que realiza las operaciones de comparación; se encuentra en la Universidad de Karlsruhe de Alemania, en la cual fue desarrollada. Trabaja con los siguientes lenguajes C, C++, Java, Scheme y texto natural. Fue desarrollada por la Universidad de Karlsruhe de Alemania.

Analiza la estructura y sintaxis del código, no comparando texto. Este sistema sólo trabaja con archivos locales, no busca similitudes en Internet. Su arquitectura de trabajo es Cliente/Servidor. La interfaz es a través de línea de comando, o, puede implementarse un cliente propio. Para el manejo de los archivos enviados, puede utilizarse el cliente brindado por la universidad que lo desarrolló o bien crear un cliente propio, como mencionamos anteriormente.

***Tester SIM*[10]**

Es una herramienta de código abierto. Trabaja con los lenguajes C, JAVA, Lisp, Modula2, Pascal y texto natural. Está desarrollada por la Universidad de Amsterdam. No cuenta con una interfaz gráfica, se ingresan los distintos parámetros por línea de comando. Los resultados brindados por la misma aparecen separados en dos columnas, mostrando en cada una las porciones de código iguales. Sólo trabaja con búsquedas de similitudes en archivos locales. También se puede procesar un conjunto de archivos viejos contra un conjunto de archivos nuevos.

Primer Análisis comparativo

En una primera etapa se tomó como caso de estudio el algoritmo de búsqueda dicotómica[1]. Este algoritmo se eligió dado que es un ejemplo claro, sencillo y muy utilizado en el entorno académico. El código está escrito en Lenguaje C.

Se confeccionaron tres archivos, con el mismo algoritmo pero con algunos cambios realizados.

Los autores Faidhi y Robinson[3] plantean 6 niveles en los cuales se identifican las partes de un algoritmo en el cual se puede modificar y/o copiar una parte del código, estos niveles son:

1. Nivel 0: es el programa original sin modificaciones
2. Nivel 1: sólo los comentarios son modificados
3. Nivel 2: sólo los identificadores son modificados
4. Nivel 3: se cambian el nombre de las variables y posiciones de las mismas, constantes y procedimientos.
5. Nivel 4: sólo se modifica la combinación en los llamados o definición de los procedimientos o funciones.
6. Nivel 5: se cambian estructuras de control iterativas, por alguna equivalente, por ejemplo cambiando un **while** por un **for**

A partir de estos niveles estos autores establecen como plagio a cualquier cambio que esté contemplado dentro del nivel 1 y 2. Teniendo en cuenta estos niveles, se realizaron cambios al

algoritmo original, de manera tal de poder comprobar el procesamiento eficiente de las distintas herramientas.

Los cambios realizados, teniendo en cuenta los 6 (seis) niveles antes descriptos, son los siguientes:

1. En uno de los algoritmos sólo se realizaron cambios a los comentarios, de aquí en adelante lo llamaremos “busqDic2”
2. En otro de los algoritmos se realizaron cambios en el nombre de las variables, de aquí en adelante lo llamaremos “busqDic3”
3. Y en otro de los algoritmos se cambió la estructura del programa, de aquí en adelante lo llamaremos “busqDic4”.

   Al algoritmo original lo llamaremos de aquí en adelante “busqDic”

Las herramientas que se utilizaron para el procesamiento y análisis de los distintos algoritmos fueron las siguientes: Sherlock, Simian, Jplag y TestSim

Para la evaluación de los resultados obtenidos, se definieron las siguientes hipótesis: la comparación entre busqDic y busqDic2 debería dar una probabilidad alta de copia, la comparación entre busqDic y busqDic3, daría una probabilidad un poco mas baja de copia, pero estaríamos dentro de un caso significativo de que sea plagio o copia, y la comparación entre busqDic y busqDic4 podría mostrar un porcentaje bajo de copia, ya que se le cambió en gran medida la estructura del algoritmo, a lo que se puede entender que es la resolución del problema de una forma parecida o bien una copia.

El proceso de comparar las distintas herramientas se llevó a cabo de la siguiente manera, se comparó el algoritmo original busqDic, contra cada una de las distintas versiones del mismo, busqDic2, busqDic3 y busqDic4.

Tabla comparativa

La Tabla 1, muestra una comparación de los resultados obtenidos en los distintos procesamientos, entre los algoritmos propuestos.

Herramienta	busqDic vs busqDic2	busqDic vs busqDic3	busqDic vs busqDic4
Sherlock	Baja probabilidad de ser copia (30%)	Probabilidad media de ser copia (62%)	Alta probabilidad de ser copia (73%)
Simian	Probabilidad media de ser copia (30 líneas de un total de 78 líneas aproximadamente)	No es copia	Baja probabilidad de ser copia (6 líneas de un total de 78 líneas aproximadamente)
Jplag	Alta probabilidad de ser copia (100%)	Alta probabilidad de ser copia (100%)	Probabilidad media de ser copia (40.6%)
TestSim	Alta probabilidad de ser copia (52 líneas de un total de 78 líneas aproximadamente)	Alta probabilidad de ser copia (52 líneas de un total de 78 líneas aproximadamente)	Probabilidad media de ser copia (18 líneas de un total de 78 líneas aproximadamente)

Tabla 1: Resultados del primer análisis comparativo.

Como se observa en la tabla anterior según el criterio mencionado anteriormente, las herramientas más eficientes son Jplag y TestSim. Los resultados brindados por la herramienta Jplag son los más intuitivos de todos, por ejemplo TestSim sólo muestra las líneas de código copiadas; por lo que se hace un poco más difícil el análisis de los resultados.

Durante el procesamiento de las distintas herramientas, se pudo observar que Sherlock fue la más rápida en cuanto a la muestra de los resultados, pero el procesamiento efectivo se pudo observar en la herramienta Jplag, como mencionamos anteriormente.

Segundo Análisis comparativo

En una segunda etapa se analizaron las herramientas con un caso real con 8 trabajos de la cátedra “Seminario de Lenguaje C” de la Facultad de Informática de la UNLP. Los mismos consistían en realizar un “Crucigrama (acróstico)” y una “Sopa de Letras” utilizando una interfaz textual. Cabe aclarar que los trabajos están en distintos archivos fuentes .c y .h, los cuales fueron unidos en un sólo archivo fuente para su mejor procesamiento. La cantidad de trabajos no es elevada ya que no se busca la comparación de los trabajos en si, si no solamente analizar el funcionamiento de las herramientas.

Se realizaron las pruebas con las siguientes herramientas: Sherlock, TestSim, Simian y Jplag, utilizando distintos tipos de configuración en cada una de ellas, teniendo un resultado en las comparaciones con un porcentaje poco significativo.

Por lo tanto, para poder evaluar el rendimiento de las herramientas antes nombradas, se tomaron dos trabajos al azar, y se les pidió a un conjunto de alumnos que realicen algunos cambios de manera tal de obtener copias similares basadas en estos trabajos originales. Al trabajo1.c se le realizaron los siguientes cambios:

- Factorización del código (cambiar varias bloques de código con la misma funcionalidad en uno solo)
- Cambios de estructuras.
- Clarificar más el código.

Estos cambios fueron realizados en el archivo con nombre “nuevo1.c”

En el trabajo8.c, el cambio consistió sólo en renombrar las variables y nombres de funciones, este es el caso del archivo con nombre “nuevo2.c”

La Tabla 2, muestra los resultados obtenidos de la comparación de los trabajos originales escogidos con los trabajos modificados según las consideraciones antes mencionadas.

Herramienta	**Trabajo8.c - Nuevo2.c**	**Trabajo1.c - Nuevo1.c**
Sherlock	100%	41%
Simian	0%	0%
TestSim	100%	68%
Jplag	100%	63,1%

Tabla 2: Resultados del segundo análisis comparativo.

Comparación de características funcionales

Además de evaluar la eficiencia de las herramientas, se realzó una evaluación sencilla de características funcionales. En la Tabla 3 se sintetiza el análisis comparativo efectuado.

Herramienta	Interfaz de usuario	Código disponible	Configurable	Fácil interpretación de resultados	Fácil de utilizar	Procesamiento local
Sherlock	Línea de comando	Si	Si	Fácil	No es sencilla la interpretación de los parámetros a configurar	Si
Simian	Línea de comando	No	Si *	Sin Datos**	No es sencilla la interpretación de los parámetros a configurar	Si
Testsim	Línea de comando	Si	Si	Fácil***	Los parámetros a configurar son fáciles de interpretar	Si
Jplag	Gráfica	No	Si	Muy Fácil	Muy intuitiva su utilización	No

Tabla 3: Características funcionales

* En esta herramienta se realizaron las pruebas utilizando los distintos parámetros, pero en los resultados mostrados se ven los parámetros seteados por defecto,

** En esta herramienta ante cualquier configuración utilizada no se obtuvieron resultados.

*** En esta herramienta la interpretación de los resultados, depende del parámetro que se ha configurado, ya que por ejemplo, si se utiliza el parámetro -p la salida se muestra resumida en porcentajes.

Conclusión

La preocupación sobre esta temática en nuestro país ya fue abordada anteriormente en [13]. En el presente artículo se ha querido describir en forma detallada casos de estudio en los cuales se hace uso de herramientas para la detección de plagio en software. Si bien las herramientas utilizadas en las pruebas permiten detectar copias en códigos fuente, el uso de la mayoría de las mismas, la forma de representar y visualizar los resultados no son intuitivos ni fáciles de interpretar.

La herramienta más completa, más fácil de usar, de interpretar sus resultados y la que mejor se adecúa a las necesidades del mundo académico es Jplag. Pero el mayor problema encontrado con Jplag es que su motor de comparación se encuentra en la Universidad de Karlsruhe de Alemania haciendo difícil la incorporación de la misma en un módulo en alguna herramienta utilizada en el ámbito académico.

Cabe mencionar que existen varias herramientas más que realizan detección de copia de código fuente, las cuatro herramientas seleccionadas son populares y fueron las que en principio se adecuaron a los requerimientos buscados. Por ejemplo la herramienta Moss [11] resultó muy difícil su configuración y tiene poca documentación, en tanto Ccfinder[12] exige muchos recursos de hardware.

Hasta el momento, no se ha podido impulsar el uso en el ámbito académico, de ninguna de las herramientas analizadas en este artículo, dadas las dificultades mencionadas. Se prevee incorporar al análisis otros productos propietarios que no han sido incluidos en este estudio, y durante este período analizar la evolución de productos open source mencionados.

Bibliografía

[1] http://www.ulpgc.es/otros/tutoriales/mtutor/ej-a.html
[2] http://es.wikipedia.org
[3] http://www.copyright.gov/circs/circ1-espanol.html
[4] "Classifications of Plagiarism Detection Engines". Thomas Lancaster, Fintan Culwin, http://www.ics.heacademy.ac.uk/italics/Vol4-2/Plagiarism%20-%20revised%20paper.pdf
[5] "Shared Information and Program Plagiarism Detection". Xin Chen; Francia, B.; Ming Li; McKinnon, B.; Seker, A. http://ieeexplore.ieee.org/iel5/18/29003/01306552.pdf
[6] "Interfaz de usuario en sistemas de detección de plagio" Diego Campo Millán; Pedro M. Latorre; Andrés Colin Higgins
http://xmariachi.rastafurbi.org/publicaciones/Interfaz_de_usuario_en_SDP__CEDI2005.pdf
[7] http://www.cs.usyd.edu.au/~scilect/sherlock/
[8] http://www.redhillconsulting.com.au/products/simian/index.html
[9] https://www.ipd.uni-karlsruhe.de/jplag/
[10] http://www.cs.vu.nl/~dick/sim.html
[11] http://theory.stanford.edu/~aiken/moss/
[12] http://www.ccfinder.net
[13] "Primeras Experiencias en Detección de Plagio en el Ambiente Educativo". Bordignon, Tolosa, Rodriguez, Peri. http://cs.uns.edu.ar/jeitics2005/Trabajos/pdf/19.pdf