UNIVERSIDAD: Universidad Nacional de La Plata

NUCLEO DISCIPLINARIO/COMITÉ ACADEMICO/OTROS TEMAS: Redes Académicas.

TITULO DEL TRABAJO: **HERRAMIENTAS PARA EL ANÁLISIS Y DETECCIÓN DE PLAGIO.**

AUTOR(ES): Anahí Soledad Rodríguez

CORREOS ELECTRÓNICOS DE LOS AUTORES: anahi@mail.linti.unlp.edu.ar

PALABRAS CLAVES: Plagio, Código Abierto, Detección de copia

Plágio, Código Aberto, Detecçao de cópia

INTRODUCCIÓN

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. Esto hace mucho más factible el plagio de este tipo de materiales.

En el ámbito académico, este tema es más preocupante. 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.

En este contexto, y priorizando un marco académico se hace indispensable el uso de herramientas para detección de plagio. De esta manera, se podrá evaluar en forma apropiada los trabajos e ideas de los alumnos.

¿Qué se conoce por 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 esta 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. [1]

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. [2]

En argentina el derecho de autor esta 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". [1]

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.

OBJETIVO

El objetivo de este artículo es el análisis y comparación de herramientas para detección de plagio en trabajos y publicaciones de estudiantes, así como la puesta en práctica de algunas de ellas; brindando un apoyo a lo docentes para una mejor evaluación de sus alumnos.

En primer lugar se analizarán varias herramientas existentes, haciendo énfasis en herramientas “*open source*”.

Como paso inicial se analizarán herramientas para detección de plagio que analicen código fuente. Cabe aclarar que este análisis es mucho más engorroso que detectar copias en documentos en texto natural ya que hay que marcar bien la diferencia entre lo que es copiar un código o que los mismos sean similares, ya que los que se busca en una cátedra es que los alumnos realicen un cierto trabajo, para el cual los mismos entregarán códigos similares. Por lo que se buscará una herramienta robusta que sea capaz de identificar cada uno de estos casos.

Como mencionamos anteriormente, el análisis se centró en herramientas “*open source*”, que puedan instalarse en un equipo local, de manera tal de poder adecuarlas más fácilmente a las necesidades planteadas.

MATERIAL Y MÉTODOS

Para el análisis de herramientas se pueden plantear una serie de características que permitirán analizar, clasificar y evaluar cuál es la más apropiada para tomar como base. A continuación nombraremos algunas de ellas:

- La interfaz de usuario 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.
- 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.

Muchas herramientas están dedicadas a encontrar similitudes dentro del mismo código, mientras que otras comparan distintos archivos. En este trabajo también trataremos de buscar que las herramientas realicen la comparación entre distintos archivos.

Para poder determinar si un programa o algoritmo ha sido copiado en su totalidad o en partes, hay que tener en cuenta que se pudo haber transcripto parte del código original, así como haberse realizado pequeñas modificaciones. Entonces es importante detectar hasta que punto se considera una copia. Los autores Faidhi y Robinson [4] establecen 6 niveles, en los cuales se define las distintas partes que componen un programa, como ser:

1. Nivel 0: es el programa original sin modificaciones
2. Nivel 1: solo los comentarios son modificadas
3. Nivel 2: solo los identificadores son modificadas
4. Nivel 3: se cambian el nombre de las variables y posiciones de las mismas, constantes y procedimientos.
5. Nivel 4: solo 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 este contemplado dentro del nivel 1 y 2.

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, y por ejemplo varias sentencias que indican un iteración se reemplazarán por el mismo tokens. Esta técnica es muy eficiente en la evaluación de código fuente, no siendo tan eficiente en textos en lenguaje natural.

Luego se realiza la etapa de evaluación de estas secuencias de tokens, a la cual se debe seleccionar una buena definición de métricas. [5] [6]

En términos generales se pueden definir a las métricas como "*un conjunto de reglas, las cuales son usadas para convertir una o mas entradas en un valor numérico, después pudiendo así encontrar generalidades entre los algoritmos, para después poder hacer mejor las comparaciones entre los mismos*".

La propuesta adoptada por Fintan Culwin [7] 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.

Los autores Thomas Lancaster y Fintan Culwin [5], realizan una caracterización de algunas métricas, utilizando algunos criterios anteriormente nombrados, así pudiendo agrupar por diferentes características, como ser:

- *Disponibilidad de las herramientas:* Las herramientas pueden trabajar tanto local como remotamente, basadas en una plataforma WEB. Dentro de esta sección se puede clasificar como Públicas o Privadas, de acuerdo a si la herramienta está disponible sólo para la institución o para todo el público
- *Por número de trabajos procesados y por la métrica usada:* podemos encontrar distintos tipos de métricas, las cuales son:
    - Las métricas simples son aplicadas a 1 o 2 documentos al mismo tiempo para generar un valor numérico. Las métricas apareadas, son usadas para obtener

más información para ser usada para un simple computo y combinando 2 métricas simples.

- Las métricas corpus y métricas multidimensionales cada una opera simultáneamente en un conjunto de documentos. Una métrica corpus trabaja en un conjunto de documentos simultáneamente, para encontrar características generales de los mismos. Una métrica multidimensional opera en un número de documentos previamente seleccionados, entonces una métrica simple será llamada de 1-dimensión, una métrica apareada será de 2-dimensiones y una métrica corpus será n-dimensional donde n es el numero de recopilaciones. Las métricas multidimensionales pueden ser útiles para encontrar cluster de similitudes.
- Las métricas superficiales y métricas estructurales: estas métricas están basadas en la complejidad del cómputo. Las métricas superficiales son aquellas que surgen de una mirada superficial a un conjunto de trabajos, sin necesidad de tener ningún conocimiento de la estructura del lenguaje ni de las lingüísticas del lenguaje natural. Al contrario que las anteriores las métricas estructurales son aquellas en las cuales es necesario el conocimiento de la estructura del lenguaje. Esto puede significar el parceo del código fuente, y para el texto natural significará reducir las palabras en las raíces de la lingüística de la palabra. Encontrar un límite entre las métricas superficiales y las estructuradas, es muy complicado ya que si un documento es tokenizado y se aplicó una métrica superficial, se podría pensar en una métrica de estructura ya que el tokenizado trabaja sobre la estructura. Por lo tanto en algunos casos estas definiciones están abiertas a la interpretación individual.

Las métricas utilizadas en herramientas para detección de plagio pueden ser derivadas de métricas utilizadas para medir perfomance 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 son las siguientes: Sherlock, Simian, DetectaCopias 1.0, Tester SIM, Jplag, Moss, CCFinder y CloneDR.

A continuación daremos una breve explicación sobre las mismas.

Sherlock [8] Es una herramienta 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 el cual 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. Solo trabaja con archivos locales, no busca similitudes en Internet.

Simian [9] Es una herramienta la cual identifica duplicación en códigos escritos en JAVA, C#, C, C++, COBOL, Ruby, JSP, ASP, HTML, XML, Visual Basic y textos natural.

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

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

Jplag: [10] 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 solo trabaja con archivos locales, no busca similitudes en Internet. Su arquitectura de trabajo es Cliente/Servidor. La interfaz con la cual trabaja esta herramienta es a través de línea de comando, o a través de un navegador WEB. Para el manejo de los archivos enviados, puede utilizarse el cliente brindado por la universidad que lo desarrolló o bien crear un cliente propio.

Moss: [11] Esta herramienta se utiliza para detectar código similar escrito en C, C++, Java, Pascal, Ada, ML, Lisp. Fue desarrollada por la Universidad de Stanford.

Esta herramienta no cuenta con una interfaz gráfica, se ingresan los distintos parámetros por línea de comando a través de un script desarrollado en PERL.

Tester SIM: [12] Trabaja con los lenguajes: C, JAVA, Lisp, Modula2, Pascal y texto natural. Está desarrollada por la Universidad de Ámsterdam. 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. Solo trabaja con búsquedas de similitudes en archivos locales. También se puede procesar un conjunto de archivos viejos contra un conjunto de archivos nuevos.

CCFinder: [13] fue desarrollada por el autor T. Kamiya. El foco de esta herramienta es analizar a grandes sistemas con un límite de dependencias del lenguaje. Transforma el código con tokens. Se basa principalmente en identificar porciones de interés (pero con una igualdad sintáctica, no de unas estructuras idénticas).

Análisis Comparativo

Herramienta	Compara Archivos Locales	Compara Archivos en la Web	Compara Doc. En Leng Natural	Cliente/Servidor	Local	Multilenguaje	Comercial	Codigo Fuente Abierto
Sherlock	Si	No	Si	No	Si	Si (c, java)	No	Si
Simian	Si	No	Si	No1	Si	Si (JAVA, C#, C, C++, COBOL, Ruby, JSP, ASP, HTML, XML, Visual Basic)	No	No
Tester SIM	Si	No	Si	No	Si	Si (C, Java, Pascal, Modula-2, Lisp, Miranda)	No	Si
Jplag	Si	No	No	Si	No	Si (C, C++, Java, Scheme)	No	No
Moss	Si	No	No	Si	No	Si (C, C++, Java, Pascal, Ada, ML, Lisp)	No	No
CCFinder	Si	no	Si	No	Si	si(C, C++, Cobol, Java, Emacs Lisp)	No (Para uso académico)	No

RESULTADOS Y DISCUSIÓN

Para realizar las primeras pruebas con estas herramientas se eligió el algoritmo de búsqueda dicotómica, dado que es un es un ejemplo claro, sencillo y muy utilizado. El código está escrito en Lenguaje C [14], siendo el siguiente:

```
/*
+---------------------------------------+
| BUSQUEDA BINARIA (BUSQUEDA DICOTOMICA) |
+---------------------------------------+
#include <stdio.h>  /* printf (), getchar () */
void main (void)
{
  int vector [100];     /* declaración de un vector de 100
                           elementos int */
  int x,                /* contiene valor a buscar */
  encontrado,           /* variable que sólo toma dos valores:
                        cierto (1) o falso (0) */
  i, centro, izquierda, derecha; /* índices de vector */
  /* Escritura de la cabecera. */
  printf ("BUSQUEDA BINARIA EN UN VECTOR ORDENADO:");
  /*  Relleno aleatorio  de los  100 componentes  del vector. A cada
  componente se le  asigna un valor equivalente al  doble del índice
  que  ocupan  en   el  vector.  A  la  variable   x  se  le  asigna
  arbitrariamente la mitad de lo que vale la variable i al salir del
  bucle. */
  for (i = 0; i <= 99; i++)
    vector[i] = i * 2;
  x = i / 2;
  /* Escritura en pantalla del vector ordenado y valor a buscar. */
  printf ("\n\nVector:\n");
  for (i = 0; i < 100; i++)
    printf ("%d\t", vector[i]);
  printf ("\nValor a buscar: %d", x);
  /* Búsqueda binaria en vector ordenado. */
  encontrado = 0;
  izquierda = 0;
  derecha = 99;
  while (! encontrado && izquierda <= derecha)
    {
      centro = (izquierda + derecha) / 2;
      if (x < vector[centro])
        derecha = centro - 1;
      else if (x > vector[centro])
        izquierda = centro + 1;
      else
        encontrado = 1;
    }
  /* Escritura en pantalla del vector ordenado. */
  if (encontrado)
    printf ("\n\nValor %d encontrado en índice %d (recordar que"
            " empieza en 0).", x, centro);
  else
    printf ("\n\nNo existe el valor %d en el vector.", x);
  /* Espera la pulsación de la tecla RETURN para finalizar el
  programa. */
  getch ();
}
```

Se confeccionaron tres archivos, con el mismo algoritmo pero con algunos cambios realizados, los mismos mencionados anteriormente propuesto por los autores Faidhi y

Robinson [2], para poder llevar a cabo la comparación entre algoritmos similares, dichos cambios realizados fueron pensados para poder comprobar el procesamiento eficiente de las distintas herramientas. Los cambios realizados son los siguientes:

1. En uno de los algoritmos solo 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 definió el siguiente criterio: 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

A continuación se muestra una tabla comparativa con 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	Alta probabilidad de	Probabilidad media

Herramienta	busqDic vs busqDic2	busqDic vs busqDic3	busqDic vs busqDic4
	ser copia (52 líneas de un total de 78 líneas aproximadamente)	ser copia (52 líneas de un total de 78 líneas aproximadamente)	de ser copia (18 líneas de un total de 78 líneas aproximadamente)

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 solo 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.

CONCLUSIÓN

Con este trabajo se analizaron y evaluaron distintas herramientas, para la evaluación de copia en códigos fuente, utilizando herramientas “Open Source”, para poder tener el código fuente a disposición por una eventual posibilidad de incorporar dicha herramienta como un módulo para cursos dictados a través de un sistema LMS como puede ser Moodle.

Si bien el análisis se centró en un único código, se puede ver que las herramientas que trabajan sobre una comparación textual del código, o sea comparando el texto y no realizando un análisis más exhaustivo de la estructura del código, no proporcionan resultados tan exactos.

Las herramientas que trabajan con un análisis textual son Simian y Sherlock, esta dos herramientas tiene una interfaz no GUI, pero Sherlock tiene a disposición el código fuente, mientras que Simian no.

Las herramientas que no realizan un análisis textual del código son TestSim y Jplag, dado como resultado más acertados. Ya que TestSim realiza un análisis de léxico del código procesado. Jplag realiza una búsqueda de similitudes en la sintaxis y estructura del programa.

No obstante la herramienta TestSim los resultados brindados por la misma no son muy cómodos a la hora de comparar varios archivos, seria muy difícil obtener una visión general de los mismos. No siendo así los resultados obtenidos por la herramienta Jplag, son los más intuitivos del resto de las herramientas.

El siguiente paso en la investigación es realizar el análisis comparativo sobre trabajos prácticos entregados por alumnos de una cátedra de la facultad de Informática de la UNLP, probando la efectividad de las herramientas en casos reales.

REFERENCIAS

[1] http://es.wikipedia.org

[2] http://www.copyright.gov/circs/circ1-espanol.html

[3] http://www.cs.utsa.edu/~wagner/pubs/plagiarism0.html

[4] http://ieeexplore.ieee.org/iel1/13/1154/00028038.pdf?arnumber=28038

[5] http://www.ics.heacademy.ac.uk/italics/Vol4-2/Plagiarism%20-%20revised%20paper.pdf

[6] http://ieeexplore.ieee.org/iel5/18/29003/01306552.pdf

[7] http://xmariachi.rastafurbi.org/publicaciones/Interfaz_de_usuario_en_SDP__CEDI2005.pdf

[8] Sherlock: http://www.cs.usyd.edu.au/~scilect/sherlock/

[9] Simian: http://www.redhillconsulting.com.au/products/simian/index.html

[10] Jplag: https://www.ipd.uni-karlsruhe.de/jplag/

[11] Moss: http://theory.stanford.edu/~aiken/moss/

[12] Tester Sim: http://www.cs.vu.nl/~dick/sim.html

[13] CCfinder: http://www.ccfinder.net

[14] http://www.ulpgc.es/otros/tutoriales/mtutor/ej-a.html