Nexo Revista Científica
Vol. 29, No. 01, pp. 01-13/Junio 2016

ISSN-E 1995-9516
Universidad Nacional de Ingeniería
http://revistas.uni.edu.ni/index.php/Nexo
http://dx.doi.org/10.5377/nexo.v29i01.4395

UTILIZACIÓN DE UNA PLATAFORMA DE PROCESAMIENTO DISTRIBUIDO PARA LA DETECCIÓN DE POTENCIAL PLAGIO CON INDICADORES DE PROBABILIDAD DE CERTEZA DE LAS TAREAS ENVIADAS A UN SISTEMA DE ADMINISTRACIÓN DE CURSOS

USE OF A DISTRIBUTED PROCESSING PLATFORM FOR PLAGIARISM POTENTIAL DETECTION WITH PROBABILITY INDICATORS OF CERTAIN IN HOMEWORKS SUBMITTED TO A SYSTEM OF COURSE MANAGEMENT.

D. Lavayen, E. Cruz

Escuela Superior Politécnica del Litoral, ESPOL, Facultad de Ingeniería en Electricidad y Computación, Campus Gustavo Galindo Km 30.5 Vía Perimetral, P.O. Box 09-01-5863, Guayaquil, Ecuador
E-mails: {dlavayen;escruz}@espol.edu.ec

RESUMEN

En este artículo se expone el análisis, diseño, implementación y pruebas de un módulo para la detección de potencial plagio de las tareas enviadas a un Sistema de Administración de Cursos, utilizando la plataforma de procesamiento distribuido Hadoop. En el presente trabajo se analiza la problemática del plagio que ocurre en las tareas elaboradas digitalmente por los estudiantes y que son receptadas por los Sistemas de Administración de Cursos. Además, se realiza un análisis conceptual para comprender cómo la necesidad de comparar dos secuencias está presente en otras ramas de la ciencia y cómo la solución ha sido propuesta con el uso de herramientas informáticas. Así mismo, se exponen las tecnologías utilizadas para el desarrollo del módulo; y se detalla cómo se hizo frente a la problemática, dividiendo el proceso en dos partes: el pre-procesamiento de los documentos para generar archivos en texto plano; y la implementación del algoritmo de Smith-Waterman con las mejoras planteadas por PhD. Robert W. Irving. Finalmente, se muestra un resumen con los resultados de las pruebas realizadas sobre el ambiente de procesamiento distribuido.

Palabras Claves: Detección, Plagio, Smith, Waterman, Hadoop, Alineamiento, Secuencias.

ABSTRACT

This paper presents the analysis, design, implementation and testing of a module for potential plagiarism detection of homework sent to a Course Management System, using the distributed processing platform Hadoop. This paper analyze the plagiarism problem that happened in homework done digitally by students and it are receipted by the Course Management Systems. Also, the paper shows a conceptual analysis for understanding how the necessity of comparing two sequences is present in other branches of science and how the solution has been proposed with the use of technological tools. Likewise, the document details how the problems were faced by dividing the process in two parts: the pre-processing of documents to generate plain text files and the implementation of the Smith-Waterman algorithm with the PhD. Robert W. Irving's improvements. Finally, the paper shows a summary of testing results done over the distributed processing platform.

Keywords: Detection, Plagiarism, Smith, Waterman, Hadoop, Alignments, Sequences.

1. INTRODUCCIÓN

Los docentes de colegios y universidades tienen diversas funciones académicas que desempeñan durante el proceso de enseñanza-aprendizaje, entre las actividades académicas que realizan los docentes se encuentra la evaluación de los trabajos que envían a sus estudiantes. Las instituciones educativas han implementado sistemas de administración de cursos para facilitar el envío y recepción de los trabajos que los docentes plantean a sus estudiantes, sin embargo, durante el proceso de evaluación de los trabajos, los docentes se enfrentan a los siguientes problemas:

- El alto índice de plagio en la presentación de tareas elaboradas digitalmente por los estudiantes.
- El esfuerzo que deben emplear los revisores de las tareas para discriminar los trabajos que son copias de otros.
- Determinar el porcentaje del trabajo que ha sido copiado para definir una calificación justa (sujeto obviamente a juicio del profesor).

La gran incidencia de plagio en la presentación de tareas, principalmente a nivel universitario, se debe al uso indiscriminado de fuentes de información que están presentes en internet como por ejemplo: Wikipedia. Estas fuentes en internet facilitan el acceso a la información, sin embargo, los estudiantes no realizan una intervención analítica de los trabajos que presentan a sus revisores y se limitan a realizar una copia directa de la información que se encuentra en el sitio.

Este tipo de copias no sólo representan una infracción a los derechos de autor de cualquier artículo utilizado (penalizado de por sí en muchos países), sino también una falta de madurez y honradez por parte del estudiante cuya conducta debería ser perfilada y corregida por el docente.

Por otro lado, la labor de monitorear sí una tarea ha sido sujeta a copia involucra la comparación sintáctica de un deber contra el resto de trabajos enviados en un paralelo en particular, pero esto puede demandar una cantidad estimable de tiempo dependiendo de la extensión de la tarea y de la cantidad de respuestas a la misma por parte de los demás alumnos en el curso.

Ahora, si se consideran los obstáculos que pueden aparecer durante la revisión de las tareas como: diferencia en extensiones, formatos, imágenes, carátulas y demás artilugios que los estudiantes suelen añadir a sus trabajos para imprimirles su “marca personal”, la simple comparación sintáctica puede convertirse en una tediosa labor que demanda demasiado tiempo.

Estos aspectos son los que han motivado el planteamiento y desarrollo de un módulo que permita agilitar el proceso de detección de plagio y provea indicadores de potenciales copias en los trabajos receptados por un sistema de administración de cursos, determinando así, qué trabajo fue plagio de otro y también en qué porcentaje fue realizada dicha copia, por consiguiente, el docente podrá tomar decisiones de corrección y calificación basadas en datos concretos y extraídos de las mismas tareas de sus estudiantes.

2. MARCO TEÓRICO

Alineamiento de Secuencias: Una secuencia es un conjunto de elementos encadenados uno detrás de otro; en el área de bioinformática las cadenas de ADN, ARN o estructuras primarias proteicas son representadas como una secuencia de caracteres donde cada caracter representa un aminoácido o nucleótido.

El alineamiento de secuencias es utilizado en bioinformática como un proceso en el que se representa y compara dos o más cadenas proteicas en busca de segmentos donde exista coincidencia entre ellos.

A continuación se ilustra proceso de alineamiento de secuencias, para ellos vamos a tomar como ejemplo dos secuencias que no se encuentran alineadas. Cuando existe la coincidencia entre un carácter de ambas secuencias en la misma posición es denominado como un "hit" Figura 1.

Figura 1. Cadenas sin alineamiento con pocos "hits".

El proceso de alinear secuencias consiste en agregar espacios en blancos a las secuencias de tal manera que desplace a los caracteres siguientes. El objetivo de este desplazamiento es lograr coincidencias en otras posiciones de la secuencia. El término con el cual se define a la inserción de un espacio en blanco es el "indel" (llamado así por su correspondiente en inglés insert delet).

Si se agregan "indels" a ambas cadenas, logramos que la cantidad de alineamientos o "hits" sea mayor Figura 2.

Figura 2. Cadenas alineadas con mayor cantidad de "hits".

Cuando no existe una coincidencia en la misma posición de ambas cadenas se lo denomina como un "replace".

El proceso de alinear dos cadenas está sujeto a encontrar cual es el alineamiento más óptimo de entre todos los posibles alineamientos, para ello es necesario definir un Sistema de Puntuación en el que se define una ponderación para cada uno de los términos antes mencionados ("hit", "indel", "replace"). Esta ponderación es multiplicada por cada una de las cantidades resultantes de cada uno de los factores y al final se suma todos estos valores de tal manera que nos da como resultado una puntuación.

Una vez definido el sistema de puntuación, un alineamiento óptimo es aquel que brinde un mayor resultado.

De acuerdo al estudio del PhD. Robert Irving, (Irving, 2004), para la detección de plagio los mejores valores a tomar en el sistema de puntuación son "hit" = 1, "indel" =-1, "replace" = -1, de esta manera se castiga los "indels" y los "replaces" y se premia a los "hits" Figura 3.

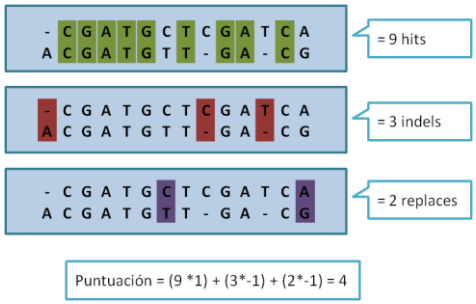

Figura 3. Sistema de puntuación.

Algoritmo de Smith – Waterman: El algoritmo de Smith-Waterman, (Smith y Waterman, 1981), es un método de comparación entre dos secuencias con el objetivo de identificar la mayor cantidad de segmentos similares. Está basado en el uso de algoritmos de programación dinámica, lo cual garantiza que el alineamiento local encontrado es óptimo con respecto a un determinado sistema de puntuación utilizado.

El algoritmo usa una Matriz de puntuación en la cual se define a cada elemento de la matriz Sij como la máxima puntuación obtenida del alineamiento entre el carácter correspondiente a la posición i de la secuencia X y el de la posición j correspondiente a la secuencia Y. La relación de recurrencia estándar para Sij donde h = hit, d = indel y r = replace es la siguiente:

$$S_{i,j}=\begin{cases} S_{i-1,j-1} + h & ; X(i) = Y(j) \\ \max(0,\ S_{i-1,j} - d,\ S_{i,j-1} - d,\ S_{i-1,j-1} - r) & ; \text{para cualquier otro caso} \end{cases}$$

Figura 4. Relación de recurrencia para S.

Teniendo como condiciones iniciales:

$$S_{i,0} = S_{0,j} = 0 \quad ; \text{para todo } i, j$$

Ejemplo: para las cadenas X = “abcdefghij” y Y = “abcxdefghiymzj” la matriz S generada a partir de su relación de recurrencia es la que se muestra en la Figura 5.

	a	b	c	x	d	e	f	g	h	i	y	m	z	j
a	**1**													
b		**2**	1											
c		1	**3**	**2**	1									
d			2	2	**3**	2	1							
e			1	1	2	**4**	3	2	1					
f					1	3	**5**	4	3	2	1			
g						2	4	**6**	5	4	3	2	1	
h						1	3	5	**7**	6	5	4	3	2
i							2	4	6	**8**	**7**	**6**	**5**	4
j							1	3	5	7	7	6	5	**6**

Figura 5. Matriz S generada a partir de X y Y.

El siguiente paso consiste en determinar la cantidad y lugar donde se van a insertar los espacios, para ello se realiza un proceso de recorrido inverso. Para lograr este recorrido se determina el antecesor del elemento y así consecutivamente con el siguiente padre en cuestión de acorde al siguiente criterio Figura 6:

Si $S_{i,j} = 0$ Entonces (i, j) no tiene antecesores

Si $X(i) = Y(j)$ Entonces (i, j) tiene su antecesor en $(i-1, j-1)$

De otro modo

(i, j) tiene su antecesor en $(p, q) \in \{(i-1, j), (i, j-1)\} / S_{i,j} = S_{p,q} - d$

y/o (i, j) tiene su antecesor en $(i-1, j-1) / S_{i,j} = S_{i-1,j-1} - r$

Figura 6. Criterio de alineamiento.

Hay que tener en cuenta que una posición no necesariamente tiene un antecesor único, lo cual puede originar varias rutas. Si el recorrido toma un comportamiento horizontal, esto indica que en la cadena se debe realizar

una inserción de un espacio por cada antecesor que se encuentre en esta posición, mientras que si el antecesor se encuentra de manera diagonal esto puede representar un "hit" o un "replace" dependiendo de la coincidencia entre X(i) y Y(j). Realizando este recorrido se obtiene las cadenas alineadas como se muestra en la Figura 7.

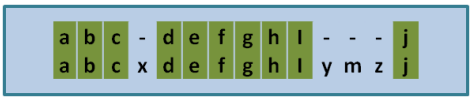

Figura 7. Cadenas alineadas usando el algoritmo de Smit-Waterman.

Teniendo como resultado final 10 hits, 4 indels y 0 replaces, que, de acorde al sistema de puntuación elegido (hit = replace= indel =1) tiene un alineamiento puntuado en: 10*(1) – 4*(1) – 0*(1) = 6.

Propuesta de Robert Irving para detección de plagios y colusiones: El algoritmo anteriormente citado muestra una puntuación determinada, pero esta puntuación puede mejorar si se realizan cortes en la revisión de la cadena, de tal manera que la puntuación total mejore compuesta por cada uno de estos subconjuntos, (Irving, 2004).

A continuación mostramos un ejemplo en la Figura 8, donde podemos notar que al realizar cortes a lo largo de la cadena podemos obtener una mejor puntuación.

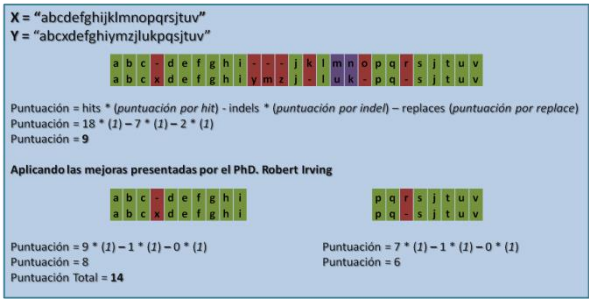

Figura 8. Alineamiento dividido para mejor puntuación.

Para aplicar estos cortes a lo largo de la cadena se debe calcular una nueva matriz denominada M, la cual se la calcula de la siguiente manera Figura 9.

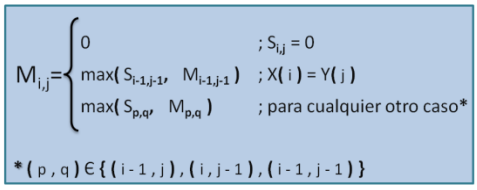

Figura 9. Cálculo de la Matriz M.

Luego de haber calculado la matriz M, se debe realizar un cálculo por cada elemento de S y M y así ir ajustando ambas matrices para tomar las coincidencias más significativas, esto se logra definiendo un valor de umbral o "threshold", el cual sirve como base para determinar cuáles serán tomadas como coincidencias significativas, siendo 1 un valor estricto, esto implicaría que una coincidencia es considerada significativa.

Si (Mij - Sij ≥ threshold) then Mij = Sij = 0;

Si (Sij ≥ threshold y Sij > Mij) Sij es parte de recorrido

3. METODOLOGÍA

Como precedente tenemos un conjunto de directorios distribuidos jerárquicamente que a su vez contienen todas las tareas digitales enviadas por los estudiantes a un sistema de administración de cursos. Además, se debe considerar que se está trabajando sobre un ambiente virtualizado en Amazon Web Services que tiene una granja de servidores, los cuales ya tienen instalado Hadoop con sus respectivas configuraciones en una distribución de Linux.

Considerando los precedentes antes mencionados hemos definidos los siguientes procesos como implementación de la solución:

3.1. Proceso de transformación a texto plano

Consiste en tomar todos los archivos con las extensiones pptx, ppt, xls, xlsx, doc, docx, pdf, html, htm, txt, zip y rar del conjunto de directorios y transformar el contenido del archivo original a texto plano, evitando detalles de: estilo, formato, gráficos, etc.

Todos los archivos y directorios del repositorio de datos deben estar contenidos en una ruta origen, la cual se la especifica en la variable pathSource de tipo String.

Utilizando la clase File de java procedemos a abrir el directorio origen de los datos y obtener todos los archivos y directorios que se encuentran en el primer nivel de la jerarquía. Este arreglo es uno de los parámetros que recibe el procedimiento “convertListOfFiles”, la misma que lleva a cabo el proceso de convertir los archivos a texto plano.

Este procedimiento recorre cada uno de los archivos y directorios que se encuentran en el arreglo, esto es, con la finalidad de buscar archivos que coincidan con el formato establecido, este proceso lo realiza la función “isSupportedFile”, la cual recibe el nombre del archivo y retorna verdadero en caso de que sea uno soportado o falso si no lo es.

Luego se crea un archivo de texto plano el cual contiene como nombre un contador que se incrementa por cada archivo que ha sido transformado a texto plano. Simultáneamente en la variable “buffer” se almacena la relación entre el número del archivo y la ruta completa de donde fue extraído ese archivo incluyendo el verdadero nombre del archivo.

Para transformar cada uno de los archivos a texto plano se usaron librerías que transforman los distintos formatos a texto plano: POI para los archivos de Microsoft y PDFBox para los pdf.

En la variable Text se almacena el contenido del archivo procesado, para motivos de nuestro análisis las palabras tildadas y no tildadas al igual que las mayúsculas y minúsculas representan el mismo valor. Los stopwords, los signos de puntuación, los caracteres especiales, secuencias de escape y los errores en Excel (#REF!, #DIV/0!, etc.), son reemplazados por un espacio que luego serán omitidos por un solo espacio de tal manera que para el análisis en el texto final quedarán palabras separadas por un solo espacio. Para remplazar estos caracteres mencionados usamos una función denominada replaceWords.

Luego de verificar si es un archivo soportado por el sistema, se evalúa si es un paquete soportado, en este caso (rar o zip), para lo cual usamos los paquetes zip que se encuentran en java.util. Para el proceso de extraer los directorios y archivos de un .rar usamos la ejecución del comando unrar.

El proceso de transformación a texto plano se aplica recursivamente a los demás niveles jerárquicos de los directorios hasta cuando se hayan explorado todos los contenidos de los directorios.

Los archivos resultados del proceso de transformación se almacenan en un directorio que se lo define en el procedimiento convertListOfFiles específicamente en la variable pathResult, la cual es una variable de tipo String que almacena la ruta del directorio.

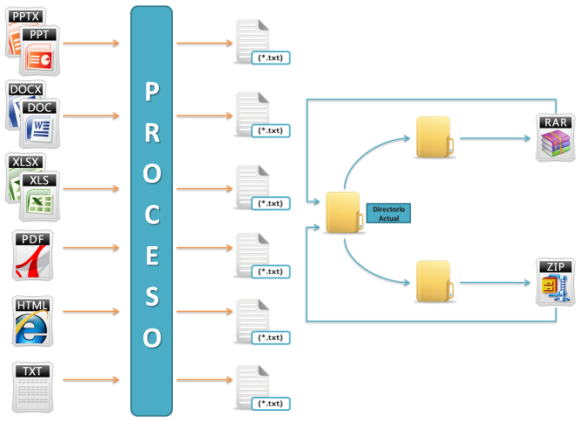

Figura 10. Proceso completo.

3.2. Proceso de identificación de plagio

Consiste en utilizar el procesamiento masivo de datos a través de Hadoop con la implementación del algoritmo de Smith-Waterman, (Smith y Waterman, 1981).

Dicho algoritmo está implementado en la clase mapper, en donde va a recibir el archivo que este procesando Hadoop del repositorio alojado en el HDFS, luego extraerá de la cache del HDFS la tarea a analizar y aplicará el algoritmo generando como resultado el porcentaje de similitud entre esos archivos.

Visualmente podríamos apreciar el proceso Map/Reduce en la Figura 11, donde se observa cómo un solo archivo es tomado como base para las comparaciones sucesivas de la caché distribuida.

Dentro de la implementación del modelo Map Reduce hay que crear una clase main, en donde se configuran los parámetros necesarios para el funcionamiento del proceso distribuido, tales como la clase Map, Reduce, Main, los tipos de parámetros que van a recibir tanto el Mapper como el Reducer; para nuestra implementación estos van a ser de tipo texto.

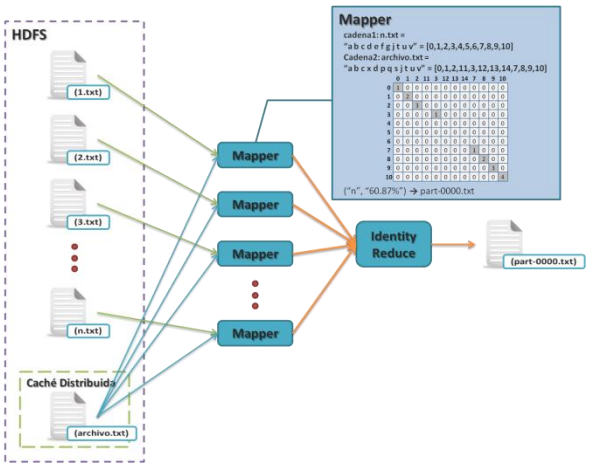

Figura 11. Modelo Map/Reduce.

En la clase main también se configura qué archivo sujeto a análisis va a alojar nuestra cache distribuida.

El siguiente paso es la recuperación del archivo que se encuentra alojado en la cache distribuida. Este proceso se lo puede realizar en la clase Mapper mediante un método denominado configure que permite recuperar el archivo de la cache distribuida y setear una variable de tipo String denominada “cadena1” que va a tener como contenido el archivo recuperado de la caché distribuida y será disponible para todos los Mappers.

En la clase mapper definimos como variables nuestro sistema de puntuación, ponderado en de 1 tanto para los hits, indels, replaces haciendo estricto el porcentaje de similitud entre los archivos comparados, además de este sistema definimos el threshold que nos denota las coincidencias significativas para lo cual hemos tomado el valor de 1, es decir el simple hecho de que coincida una palabra ya es una coincidencia significativa.

La función map recibe como uno de sus argumentos una variable de tipo Reporter, la cual tiene como parte de su contenido un archivo extraído del HDFS durante el procesamiento distribuido, de este archivo extraemos el contenido y lo almacenamos en la variable “cadena2”. El objetivo es crear dos arreglos de números a partir de “cadena1” y “cadena2” respectivamente en donde las palabras que se encuentren tanto en la primera cadena como en la segunda deberán tener el mismo valor numérico en ambos arreglos.

Para implementar esta asociación se utiliza la ayuda de una estructura HashMap que será un diccionario de las palabras que hayan sido recorridas con un respectivo identificador numérico.

Para poder asociar la mayor cantidad de palabras con números tomamos en cuenta los números negativos y como inicio el mínimo numero negativo hasta el máximo entero positivo permitido.

Conociendo la cantidad de palabras en base a la dimensión de cada uno de los arreglos se crean la matriz S y la matriz M utilizando la librería JAMA que permite un manejo óptimo de matrices.

Luego de este cálculo de ambas, usamos la forma revisada del algoritmo de Smith-Waterman, (Smith y Waterman, 1981), la cual nos da como resultado las coincidencias significativas definidas por el threshold y almacenadas en la cadena C, de donde podemos obtener un resultado de cuantas coincidencias significativas existieron en esa comparación de un total de palabras y poder calcular en que porcentaje se parecen dos archivos.

El Reducer se encarga de recolectar la información generada por los mappers y generar un archivo part con los resultados.

4. RESULTADOS Y DISCUSIÓN

4.1. Pruebas de eficacia

Para ilustrar paso a paso el procedimiento realizado, tomaremos un par de archivos de nuestro conjunto de datos; aunque sus nombres de archivos son originalmente “Trabajo de Ecología.doc” y “Deber de Ecología y educación Ambiental.pdf” los llamaremos por simplicidad: original y copia (en ese orden respectivamente).

En la copia los cambios realizados obedecen a modificaciones en formato, tipo de texto, inserción de imágenes, cabeceras, pies de página y cambio en el formato de presentación (pdf). Además los párrafos fueron dispuestos en un orden diferente con la mínima modificación en su contenido.

Después del proceso de transformar los documentos a texto plano, parte de los resultados son los siguientes como se muestra en la Figura 12:

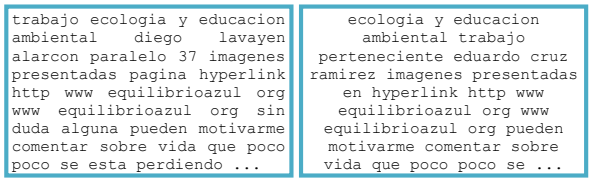

Figura 12. Parte inicial de los archivos en texto plano.

Se puede observar como en los archivos planos se discrimina por completo cambios puntuales en el contenido, como por ejemplo: poner en mayúscula cierto texto, capitalizar y/o tildar palabras, signos de puntuación, etc.

Como se explicó anteriormente, en la fase Map los contenidos de los archivos a analizarse en cada instancia son convertidos automáticamente a un vector de números que representan de manera única a cada palabra.

Después de haber preparado este par de vectores, se realizar el algoritmo mejorado de Smith-Waterman, (Irving, 2004), y como resultado obtenemos la matriz de puntos S modificada que, por cuyo tamaño, lo hemos simplificado a una representación gráfica, en la que cada palabra similar entre el documento original y la copia es representada por un punto negro.

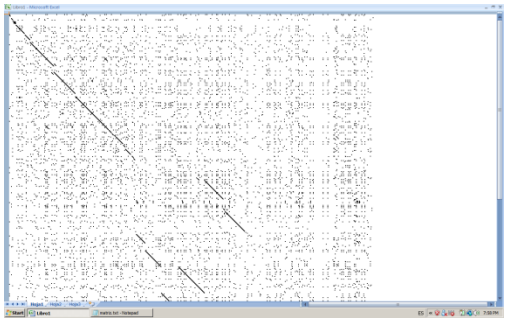

Figura 13. Representación gráfica de la Matriz S.

En la Figura 13 se puede apreciar cómo estos puntos forman diagonales segmentadas que representan coincidencias secuenciales de las palabras en ambos documentos.

La razón por la que dichas diagonales no son continuas y no se encuentran totalmente alineadas es debido a que las coincidencias existen sólo en ciertas secciones de ambos documentos, secciones que no necesariamente están presentes a la misma altura (debido al movimiento de los párrafos en la copia).

Al finalizar este ejemplo el sistema retornó un resultado de 87.86 %, indicando que de las 1120 palabras del documento original, 984 coinciden con el documento copia no sólo en ocurrencia sino también en una secuencia determinada.

4.2. Pruebas de eficiencia

Gracias a los datos de tiempo que se tomaron en cada procesamiento de la clase principal, los resultados son las siguientes tablas y gráficas comparativas:

4.2.1. Cantidad de archivos fijos y cantidad de nodos variables

Para el primer grupo de pruebas, se estableció constante la cantidad de archivos (volumen de datos), variando la cantidad de nodos utilizados por prueba.

Tabla 1. Resultados del primer grupo de prueba

	2 Nodos	**6 Nodos**	**10 Nodos**	**20 Nodos**
1era. Prueba	1:53:49	1:03:34	0:47:59	0:48:53
2da. Prueba	1:46:26	1:08:31	0:52:51	0:34:07
3era. Prueba	2:10:11	1:12:49	0:45:43	0:38:31
Media	1:56:49	1:08:18	0:48:51	0:40:30

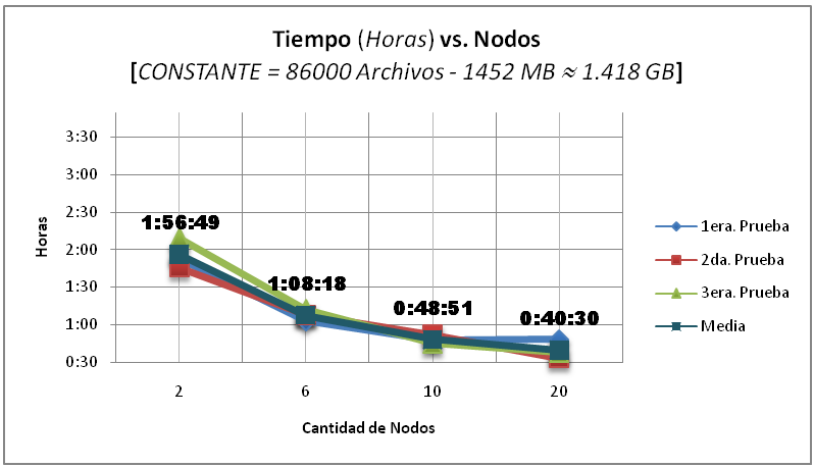

Figura 14. Resultados del primer grupo de pruebas.

Observamos en la Figura 14, cómo el tiempo requerido para el procesamiento de una cantidad constante de datos (según las pruebas realizadas) se ve reducido con el aumento de nodos asociados a cada procesamiento.

Podemos ver cómo la media se va comportando de manera poco variable a partir del uso de aproximadamente 10 nodos.

Al finalizar la prueba 1 observamos cómo el tiempo tomado aumenta en referencia al resto de pruebas con 20 Nodos, esta anomalía se presenta debido a diversos aspectos como: el tráfico en la red interna en las granjas de clústeres de Amazon, la tolerancia a fallos (caída inadvertida de un nodo y su inmediata reposición), la demanda de los servicios de Amazon, entre otros…

Inicialmente es evidente cómo dos clústeres no son suficientes para la cantidad de datos que se están analizando, ya que aproximadamente 2 horas de procesamiento para 1.418GB de datos es considerablemente mucho tiempo.

4.2.2. Cantidad de nodos fijos y cantidad de archivos variables

Para este grupo de pruebas se estableció constante la cantidad de nodos utilizados para el procesamiento, manteniendo variable el volumen de datos para cada prueba.

Tabla 2. Resultados del segundo grupo de pruebas

	221MB 25000 Archivos	**634MB** 45000 Archivos	**1126MB** 65000 Archivos	**1452MB** 86000 Archivos
1era. Prueba	0:08:16	0:13:06	0:26:37	0:49:47
2da. Prueba	0:11:15	0:16:36	0:33:35	0:46:16
3era. Prueba	0:09:09	0:14:51	0:29:11	0:45:17
Media	0:09:33	0:14:51	0:29:48	0:47:07

Es evidente y comprobado con la siguiente gráfica, que, a medida que aumenta el número de archivos analizados, considerando una cantidad constante de nodos levantados (para este caso 20 clústeres), el tiempo requerido se acrecienta proporcionalmente; llegando a un máximo local, para estas pruebas, de 47 minutos y 7 segundos.

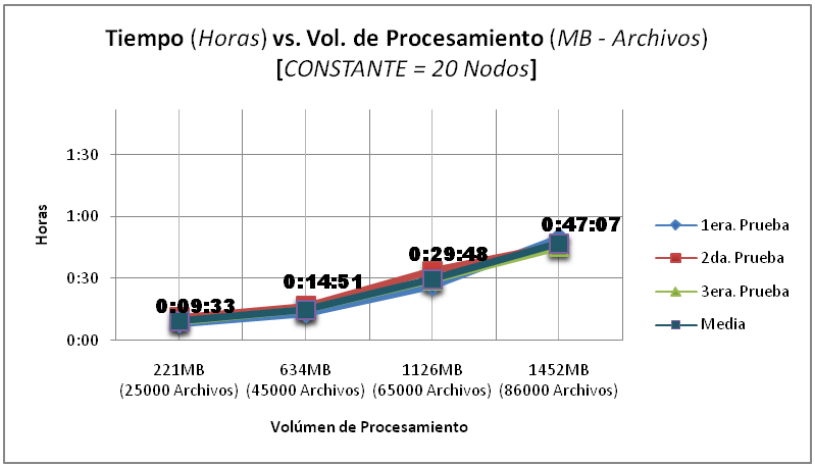

Figura 15. Resultados del segundo grupo de pruebas.

De acuerdo a lo mostrado en la Figura 15, con un conjunto de datos de 86000 archivos, aproximadamente el sistema demoraría menos de una hora en terminar su procesamiento, comparando un subconjunto de éstos con el total.

5. CONCLUSIONES

La eliminación de palabras vacías en el proceso de transformación de documentos binarios a archivos de texto plano, optimiza la comparación con el algoritmo de Smith-Waterman, debido a que se considera una menor cantidad de palabras y por ende matrices de menor tamaño en memoria.

La representación de los documentos en archivos de texto plano resulta la forma más conveniente para realizar comparaciones, debido a que se utiliza una implementación optimizada para su parseo.

El algoritmo de Smith-Waterman implementado con las modificaciones propuestas por Robert Irving presentan una mayor exactitud a la hora de establecer secuencias similares (no necesariamente en el mismo orden) entre dos cadenas sujetas a evaluación.

La implementación de este algoritmo en un ambiente distribuido no representa mayor demanda en cuanto a cambios en la estructura lógica inicial diseñada para un contexto mono-procesador. Debido a que, en este y en muchos casos, el análisis se realiza de manera no secuencial por archivo.

El comportamiento de los nodos, trabajando por demanda de volumen de datos, fue el esperado, presentando una tendencia a la baja, cuando se aumentaba la cantidad de clústeres utilizados con una cantidad constante de datos, que se estabiliza en la utilización de aproximadamente 10 nodos; y presentando una tendencia a la alta cuando el volumen de datos aumenta en un escenario constante de Nodos.

RECOMENDACIONES

En la implantación del módulo de detección de plagio en un Sistema de Administración de Cursos ya operativo, se debe considerar:

Realizar la transformación de documentos a archivos de texto plano automáticamente cuando un archivo es subido al Sistema de Gestión de Cursos.
Realizar la copia de este archivo de texto plano al S3 de manera síncrona.
Registrar el código de este archivo con su ubicación en una tabla de la base de datos utilizadas en el sistema general.
El consumo del archivo resultante después del procesamiento debería ser capturado y almacenado en una tabla de la base de datos utilizada en el sistema general, para llevar un histórico del análisis.

AGRADECIMIENTOS

A la MSc. Cristina Abad Robalino, por sus constantes consejos y ayuda a lo largo del proyecto.
A la MBA. Ana Tapia Rosero, por su gentil ayuda en la culminación del presente.
Al MSc. Federico Raue, por sus muy útiles recomendaciones que mejoraron el performance de nuestra propuesta.

REFERENCIAS

Altschul, S.F., Gish, W., Miller, W., E.W. Myers, y Lipman, D.J. A basic local aligment search tool. Journal of Molecular Biology, 215:403 – 410. Octubre 1990.

Apache. Hadoop. “http://hadoop.apache.org/”. Febrero 2010

Huang, X., Hardison, R.C., y Miller, W. A space-efficient algorithm for local similarities. CABIOS, 6:373 – 381. Marzo 1990.

Irving, R. Plagiarism and Collusion Detection using the Smith-Waterman Algorithm, University of Glasgow. Julio 2004.

Lancaster, T. Effective and Efficient Plagiarism Detection. PhD thesis, School of Computing, Information Systems and Mathematics, South Bank University. Enero 2003.

Smith, T.F., Waterman, M.S. Identification of common molecular subsequences. J Mol Biol. 147 (1): pp. 195-7. PMID 7265238. Marzo 1981.

White, T. Hadoop: The Definitive Guide. O'Reilly Media. pp. 524. ISBN 0596521979. Mayo 2009.

Zhang, Z., Berman, P., y Miller, W. Alignments without low scoring Regions. Journal of Computational Biology, 5:197 – 210. Febrero 1998.