PONTIFICIA **UNIVERSIDAD CATÓLICA** DEL PERÚ

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

SISTEMA DE INFORMACIÓN DE DETECCIÓN DE PLAGIO EN DOCUMENTOS DIGITALES USANDO EL MÉTODO DOCUMENT FINGERPRINTING

Tesis para optar por el Título de Ingeniero Informático, que presenta el bachiller:

Fernando Emilio Alva Manchego

ASESOR: Ing. Manuel Tupia Anticona

Lima, enero del 2010

Resumen

Actualmente, el plagio de documento digitales es un problema que afecta, en diferentes dimensiones, a la sociedad en su conjunto y, muy especialmente, al ámbito académico. Conociendo las graves consecuencias que puede traer su expansión, se decide realizar un trabajo que consiste en la implementación de un sistema de información que permita la detección automática de plagio en documento en formato electrónico.

Se incurre en plagio cuando se indica que una persona es autora de “algo” (ideas plasmadas en documentos) que en realidad no ha elaborado y que ha copiado de otra persona o fuente sin realizar alguna modificación sustancial o un aporte personal y sin referenciar convenientemente a la autora original.

Debido a que este es un problema de carácter universal, existe una gran variedad de esfuerzos alrededor del mundo motivados por el objetivo de combatir el plagio en todas sus formas. Esto ha hecho posible que se desarrollen diferentes métodos y herramientas que permiten alcanzar el objetivo de la comparación de similitudes entre documentos, que es, en esencia, el problema que se pretende solucionar.

Uno de los métodos que se emplea es el *Document Fingerprinting,* el cual es un algoritmo que permite extraer un conjunto de valores numéricos del documento que representan a varias porciones del mismo. El conjunto de estos valores recibe el nombre de *fingerprint* del documento. Mediante la comparación de las *fingerprintgs* de distintos documentos es que se puede detectar si es que lo documentos poseen secciones comunes y, por tanto, se ha incurrido en plagio.

En este proyecto, se implementa un sistema de información que emplea una instancia específica del algoritmo *Document Fingerprinting,* llamada *Winnowing,* para obtener un mejor resultado en la comparación. Asimismo, se provee al sistema de una interfaz de usuario apropiada para el análisis de documentos en búsqueda de plagio.

A Dios,

A mis padres Emilio y Magdalena,

A mi hermano Daniel,

A mis amigos,

A todos quienes creen en crear y no solo usar tecnología.

AGRADECIMIENTO

A mi familia, por darme su amor, confianza y apoyo en todo lo que me propongo.

A mis amigos, por todos los buenos momentos y las lecciones aprendidas.

A mi asesor, por todo lo que me ha enseñado de esta especialidad tan apasionante.

A todas aquellas personas que de alguna u otra manera me han sabido dar su apoyo y sus buenos deseos.

A Dios, por poner a todas estas personas maravillosas en mi camino y por llenarme de bendiciones a mí y a todos mis seres queridos.

Tabla de Contenidos

Índice de Figuras

Introducción

Actualmente, el plagio documentario es un problema que afecta, en diferentes dimensiones, a la sociedad en su conjunto. La gravedad de las consecuencias que puede traer su expansión, motiva a que investigadores traten de entender a mayor profundidad este problema para así desarrollar herramientas que ayuden a combatirlo, sobre todo en ámbitos académicos. Estos esfuerzos se ven reflejados en diferentes áreas académicas y, como tal, la ingeniería informática no se ha quedado atrás, desarrollando aplicaciones que permitan la detección del plagio entre diferentes tipos de documentos. Es con este incentivo que se decide realizar un trabajo consistente en el desarrollo de un sistema de información que permita la detección de plagio en documentos digitales.

Como se menciona, existe una gran variedad de esfuerzos alrededor del mundo motivados por el objetivo de combatir el plagio en todas sus formas. Es así, que existe vasta información sobre métodos y herramientas ya implementadas que permiten alcanzar el objetivo de la comparación de similitudes (solapamiento) entre documentos. Esto permite que la herramienta que se plantea desarrollar tenga un fuerte soporte teórico y tecnológico que respalde los métodos que se emplean en su desarrollo. Así mismo, la prueba de que herramientas similares ya hayan sido implementadas y disfruten de gran éxito en distintas parte del globo, evidencia la alta necesidad que existe en la sociedad actual por aplicaciones que ayuden a combatir el plagio; esto permite estimar el gran nivel de utilidad que el sistema a desarrollar tendría en el contexto actual.

La estructura del documento es como sigue: en el capítulo 1 se revisan los principales conceptos relacionados con el problema del plagio y el marco legal que lo envuelve; asimismo, se hace una revisión profunda del estado del arte de las herramientas y los métodos que se emplean para combatir dicho problema. Al final de este capítulo se realiza una breve descripción de la solución que se plantea implementar. En el capítulo 2 se describen las principales funcionalidades que el sistema posee, así como se

presentan las arquitecturas de software e información que se emplearán. En el capítulo 3 se describe a profundidad el algoritmo principal a emplear, así como procesamientos adicionales que se deben llevar a cabo para la obtención de una mejor respuesta. En el capítulo 4 se detallan algunas características de la implementación de la herramienta y, finalmente, en el capítulo 5 se indican las conclusiones y observaciones obtenidas luego de la implementación de la herramienta; de la misma manera, se brindan pautas para incrementar la precisión en la detección del sistema, dejando las mismas como posibles trabajos futuros.

"No es lo que obtenemos sino lo que contribuimos lo que le da un significado a nuestras vidas"

Anthony Robbins

1 Generalidades

En este capítulo se presentan los conceptos principales necesarios para poder comprender el problema que se atacará, así como un análisis detallado del estado del arte, considerando tanto herramientas que cumplan una función similar a la que se desea implementar, como métodos propuestos por investigadores a nivel internacional. Finalmente, se presenta la sustentación que justifica la implementación del sistema de información que se plantea.

1.1 Definición del Problema

Para poder entender el problema del plagio en la actualidad, es menester comprender a cabalidad lo que implica este término. La Real Academia Española de la lengua define al *plagio* como "***copiar en lo sustancial obras ajenas, dándolas como propias***" [**1**]. En un artículo publicado por el vicerrectorado académico de la Pontificia Universidad Católica del Perú se indica que "***el plagio consiste en hacer pasar como nuestras ideas, los textos que pensaron otros y que nos fueron transmitidos por ellos, bien por escrito, bien oralmente o por algún otro mecanismo de comunicación***" [**2**]. En pocas palabras, se incurre en plagio cuando se indica que una persona es autora de "algo" (ideas plasmadas en documentos) que en realidad no ha

elaborado y que ha copiado de otra persona o fuente sin realizar alguna modificación sustancial o un aporte personal y sin referenciar convenientemente a la autora original.

El reflejo de este problema se puede ver en diferentes ámbitos de nuestra sociedad: desde el plagio de letras de canciones populares hasta el plagio documentario dentro de las esferas académicas y de investigación. Es precisamente el ámbito académico universitario el que mayores consecuencias negativas conlleva, pues es de sus miembros de quienes se esperaría una formación ética más completa.

De acuerdo con lo expuesto en [**3**], el plagio se puede realizar de manera tanto intencional como no intencional. Se realiza de forma *intencional* cuando se copian ideas directamente tal y como fueron expuestas por el autor, estando completamente consciente de lo realizado. Por otro lado, se realiza de forma *no intencional* cuando sucede por desconocimiento; por ejemplo, cuando no se colocan las referencias adecuadas de las fuentes consultadas en la realización de un documento. En este último caso, se consideraría, por ejemplo, el parafraseo o traducción de documentos sin indicar la fuente de la cual se obtuvieron los mismos.

El plagio de documentos a nivel académico no es realmente un problema moderno: se ha vuelto más común en la actualidad con la gran expansión del uso de las tecnologías de la información y, en particular, del Internet. Como lo menciona Urbina: “siempre ha habido alumnos que han copiado de diversas fuentes como libros, enciclopedias, otros trabajos; lo realmente novedoso consiste en la facilidad que supone la *Web* para hacerlo. En apenas unos pocos clics de ratón podemos disponer de un trabajo con bastantes posibilidades de éxito académico” [**4**].

Este aumento en el uso de la vasta información que provee Internet y que los estudiantes emplean para sus trabajos de manera indiscriminada e incurriendo en plagio, ha llevado a la creación del término: *ciber – plagio académico*. Rubén Comas lo define como “el uso de las TIC (principalmente Internet y los recursos asociados a ésta –sobre todo el WWW-) para el plagio (total o parcial) de trabajos académicos por parte del alumnado” [**5**]. El ciber-plagio es una práctica cada vez más común y esto se ve evidenciado en el gran número de visitas que reciben sitios Web dedicados a la publicación de trabajos académicos. De acuerdo a un ranking de las 100 páginas Web

en español más visitadas elaborado por la compañía de información Web Alexa, conocidos sitios web del género como son www.monografías.com y www.rincondelvago.com ocupan los puestos 26 y 30, respectivamente, entre los sitios Web con más visitas dentro de la comunidad hispana [**6**].

Sin embargo, a pesar del actual auge del ciber-plagio, los estudiantes aún cometen delitos de plagio de la manera *tradicional* (se entiende por tradicional a la falta de referencias a consultas realizadas a libros texto y no a información obtenida desde Internet). De acuerdo a un estudio presentado en la revista *Ethics & Behavior*, el 42.6% de los estudiantes encuestados reportaron el uso de tanto métodos convencionales como digitales para cometer delitos de plagio, mientras que sólo el 4.2% empleaba exclusivamente métodos tradicionales [**3**]. Basados en éstas estadísticas, los autores concluyen que Internet y otras herramientas digitales son en realidad conductos y no causas del problema de deshonestidad académica y plagio.

Adicionalmente, se debe mencionar que el plagio a nivel académico, y especialmente en las áreas de las Ciencias Naturales y la Investigación, tiene efectos altamente negativos para el desarrollo correcto del Saber Humano. Esto se debe a que, en palabras de Lucio Mendieta, "la investigación y la especulación científica y filosófica forman, desde el origen de la humanidad, una cadena infinita que se sucede a través de las generaciones, cada una parte de lo ya realizado por las anteriores para agregar, después de ímprobos esfuerzos que requieren, a veces, años de trabajo, algo nuevo o diferente que aumenta de manera positiva el acervo de la cultura y de la ciencia universales" [**7**]. Por tanto, si se comete plagio en la investigación científica y académica no se está cumpliendo con el objetivo de aportar algún tipo de conocimiento nuevo o combinación diferente de conocimientos preexistentes, de tal forma que sirva para el desarrollo de toda la sociedad.

Por ende, se puede decir que el plagio es en realidad el gran problema de los ámbitos académicos y científicos: los estudiantes e investigadores, muchas veces sin proponérselo, extraen para sus labores y asignaciones, fragmentos de trabajos realizados por otros autores sin su debido reconocimiento. Las consecuencias, dependiendo del marco legal de cada país, que el cometer plagio trae consigo no están meramente referidas a penalizaciones a nivel judicial o administrativo, sino que atentan contra todo el futuro de la ciencia y la sociedad en su conjunto. Es por todo ello, que es

necesario hacer frente a este problema de forma inmediata y contundente mediante el empleo de las mismas tecnologías de información que favorecen, en cierta medida, al plagio mismo.

1.2 Marco Conceptual

Es necesario poder entender claramente los conceptos y términos relacionados con el manejo de grandes cantidades de documentos, su almacenamiento y procesamiento, con el fin de poder llegar a determinar el grado de similitud o diferencia que poseen en comparación con otros o entre ellos mismos. Así mismo, se debe de poder entender a cabalidad conceptos relacionados con el reconocimiento de patrones que es, algorítmicamente hablando, el problema que finalmente se desea solucionar.

Por otro lado, también, es necesario poder comprender conceptos relacionados con el plagio documentario y así entender mucho mejor el gran impacto que la herramienta a desarrollar tendrá.

Con esto en mente, se procede a definir ciertos conceptos cuyo entendimiento será vital para poder comprender claramente lo expuesto en los siguientes capítulos.

1.2.1 Bases de Datos de Texto (BDT)

Una base de datos de texto (BDT) es un sistema que mantiene una gran colección de documentos relacionados que se encuentran ensamblados en una única unidad en la cual se pueden realizar búsquedas. Los documentos individuales pueden ser de cualquier tamaño, pero deben de mantener una relación entre ellos. Como se sabe, una base datos está compuesta de pequeñas unidades llamadas registros; en el caso de las bases de datos de texto un registro puede ser un documento por completo, una sección dentro de un documento, una página o un fragmento del mismo [**8**]. Al igual que una base de datos convencional, una BDT debe proveer acceso rápido y seguro a los textos que almacena [**9**].

Las BDT se pueden clasificar en *Bases de Datos de Texto Estructurado* y *Bases de Datos de Texto No Estructurado*. En las primeras, los elementos básicos son documentos con estructura como HTML o XML; mientras que en las segundas, los

documentos no tienen estructura definida, no contienen palabras clave que permitan diferenciar unas partes de otras en el documento. Un ejemplo de estas últimas son los repositorios de información, los cuales almacenan grandes volúmenes de datos que cambian constantemente y no están estructurados.

Como se puede apreciar, debido a la naturaleza de las BDT las búsquedas que se realizan en ellas no son como las que se llevan a cabo en una base de datos relacional convencional. Las búsquedas que se pueden realizar en una BDT son *semánticas* y *sintácticas* [**9**]. En las primeras, el usuario da la información que desea recuperar y el sistema retorna todos los documentos que son relevantes. Por otro lado, en las búsquedas sintácticas, se posee un patrón de búsqueda (cadena o expresión regular) y se realizan búsquedas aproximadas

1.2.2 Reconocimiento de Patrones

El reconocimiento de patrones consiste en el etiquetado (clasificación o asignación de nombres) de los elementos del universo (objetos, fenómenos, conceptos). Para poder comprender estos elementos, se llevan a cabo *procesos perceptuales*, los cuales, en el ser humano, pueden ser modelados como un sistema de tres estados: adquisición de datos sensoriales, extracción de características y toma de decisiones [**10**]. En la Figura 1.1 se puede apreciar la interacción entre estos tres estados.

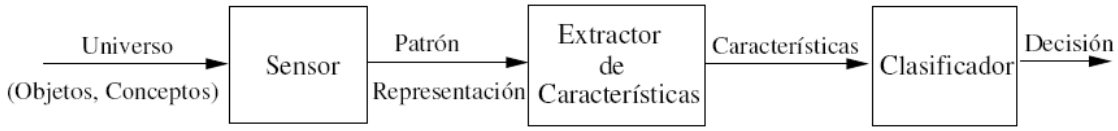

Figura 1.1 - Etapas en un sistema de reconocimiento de patrones

El *sensor* es el encargado de proporcionar una representación factible de los elementos del universo que se desea clasificar. Para ello, debería de estar diseñado de tal manera que pueda medir directamente todas las propiedades físicas que distinguen a los elementos en las diferentes clases; sin embargo, en la práctica, no siempre es posible conseguirlo por diferentes factores, tales como la falta de conocimiento suficiente para lograrlo, incapacidad para realizar una medición directa de las propiedades o inviabilidad económica.

El *extractor de características* actúa como un filtro, eliminando información redundante o irrelevante, y entregando sólo aquella información discriminatoria, partiendo del patrón de representación.
Finalmente, el *clasificador* se encarga de asignar a la categoría apropiada los patrones de clase desconocida a priori.

1.2.3 Árboles de Decisión

Un árbol de decisión es una estructura de datos que permite representar condiciones y acciones, muestra qué condiciones se consideran en primer lugar, en segundo lugar y así sucesivamente. Este método permite mostrar la relación que existe entre cada condición y el grupo de acciones permisibles asociado con ella [**11**].

Como se expone en [**12**], un árbol de decisión está formado por una serie de nodos, los cuales representan, individualmente, a un atributo. Las ramas que salen de los nodos corresponden a los posibles valores del atributo correspondiente. De este modo, los árboles de decisión permiten clasificar cada instancia de modo descendente, comenzando por el nodo raíz y bajando por la rama del árbol correspondiente al valor de ese atributo. Este proceso se repite para cada nodo, llegando finalmente a un nodo hoja, que corresponde a la clasificación buscada. Una vez creado el árbol de decisión, para un nuevo caso únicamente sería necesario recorrer dicho árbol para establecer la clasificación a la que pertenece.

A continuación, se presentan unos tipos especiales de árboles de decisión que son empleados especialmente para el reconocimiento de patrones en textos.

- **Árbol Digital o Trie**

De la misma forma como se explica en [**9**] y [**13**], sea:

A un alfabeto, $|A| = m$
S un conjunto de cadenas, $S \subseteq A^*$
s una cadena dada, $s \in A^*$

El objetivo es determinar si la cadena dada s pertenece al conjunto de cadenas S.

Un árbol digital o trie es aquel en el cual cada una de sus ramas está rotulada con un único carácter de A; cada hoja contiene una cadena de S; por tanto, dado s, se usan los caracteres que lo conforman para direccionar la búsqueda. Es así que, $s \in S$ si se puede obtener *s* concatenando los rótulos en un camino desde la raíz a un nodo terminal.

Ejemplo:

Sea S = {oso$, saco$, saca$, sal$}

El árbol digital o trie correspondiente al conjunto de cadenas dado se presenta en la Figura 1.2:

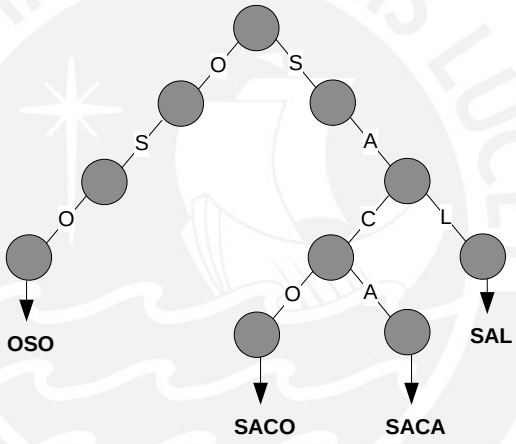

Figura 1.2 - Ejemplo de árbol digital o trie

- **Árbol PATRICIA**

Las siglas PATRICIA corresponden a **P**ractical **A**lgorithm **t**o **R**etrieval **I**nformation **C**oded **i**n **A**lphanumeric. Es una versión más comprimida de un árbol trie puesto que los nodos pueden etiquetarse con más de un carácter, tal y como se muestra en la Figura 1.3.

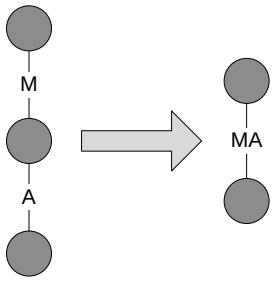

Figura 1.3 - Reducciones realizadas por un árbol PATRICIA [13]

Gracias a esta característica, el trie PATRICIA elimina las ramas no esenciales que corresponden a nodos internos que tienen un solo hijo. En la Figura 1.4 se muestra un ejemplo de cómo se ve reducido un árbol trie al aplicar la transformación PATRICIA.

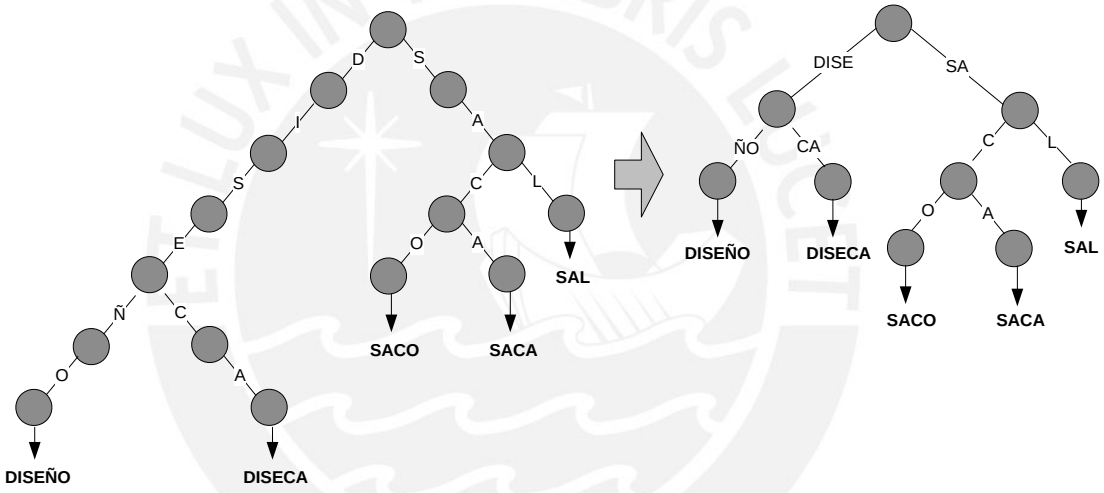

Figura 1.4 - Transformación de un trie normal a un trie PATRICIA [13]

Sin embargo, una mejor manera de representar un árbol PATRICIA es indicando en los nodos la posición en la cadena que hace la diferenciación entre dos palabras [**9**]. Por ejemplo, si tenemos el siguiente conjunto de palabras:

Sea S = {animar$, animo$, animal$, casa$, caza$, cazador$, cosa$}

El árbol PATRICIA correspondiente sería el que se muestra en la Figura 1.5.

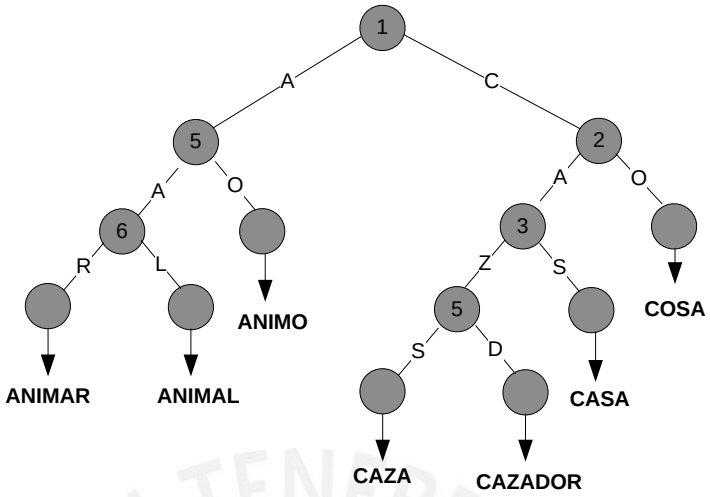

Figura 1.5 - Ejemplo de árbol PATRICIA [9]

- **Árbol de Sufijos (Suffix Tree)**

Un árbol de sufijos es una estructura de datos que permite almacenar todos los posibles sufijos de una cadena, y por tanto todas las sub-cadenas posibles de la cadena original. Este tipo de estructura es empleado para resolver muchos problemas de comparación de cadenas debido a su versatilidad, además que puede ser construido usando diferentes algoritmos.

Tomando como referencia a [**9**]:

Sea *T* una cadena de longitud *n* (denotamos $T_{1...n}$) entonces:
$T_{1...i}$ es un **prefijo** de T ($\forall$ i = 1...n)
$T_{i...n}$ es un **sufijo** de T ($\forall$ i = 1...n)
Denotaremos con **SUF (*T*)** al conjunto de todos los sufijos de T.

"Sea **T** un texto y **P** un patrón de búsqueda, entonces **P** ocurre en **T** si **P** es prefijo de algún sufijo de **T.** [**9**]"

Un ejemplo de este tipo de árboles se puede apreciar en la Figura 1.6.

	1	2	3	4	5	6	7	8
T =	a	b	c	c	a	b	c	a	$

SUF (T)	Posición	
a b c c a b c a $	1	
b c c a b c a $	2	P = bc
c c a b c a $	3	
c a b c a $	4	
a b c a $	5	
b c a $	6	
c a $	7	
a $	8	

Figura 1.6 - Ejemplo de árboles de sufijos (Parte A). Construcción del árbol de sufijos [9].

Luego de construir el árbol de sufijos correspondiente, se puede transformar en un árbol de PATRICIA para aprovechar sus ventajas al momento de realizar búsquedas.

Primero se procede a obtener los códigos binarios que identificarán a cada sufijo. En este caso, se emplean los códigos ASCII en notación binaria para poder hallar dichas representación numérica, tal y como se muestra en la Figura 1.7.

	1	2	3	4	5	6	7	8	
T =	a	b	c	c	a	b	c	a	$	a = 00 b = 01 c = 10 $ = 11

SUF (T)	En binario	Posición
a b c c a b c a $	00 01 10 10 00 01 10 00 11	1
b c c a b c a $	01 10 10 00 01 10 00 11	2
c c a b c a $	10 10 00 01 10 00 11	3
c a b c a $	10 00 01 10 00 11	4
a b c a $	00 01 10 00 11	5
b c a $	01 10 00 11	6
c a $	10 00 11	7
a $	00 11	8

Figura 1.7 - Ejemplo de árboles de sufijos (Parte B). Primera parte de la transformación del árbol de sufijos a un árbol de PATRICIA. Se obtienen los códigos binarios correspondientes a cada sufijo [9].

Luego, se emplean dichos códigos binarios para poder armar la estructura de árbol PATRICIA para el conjunto de sufijos correspondiente (Figura 1.8).

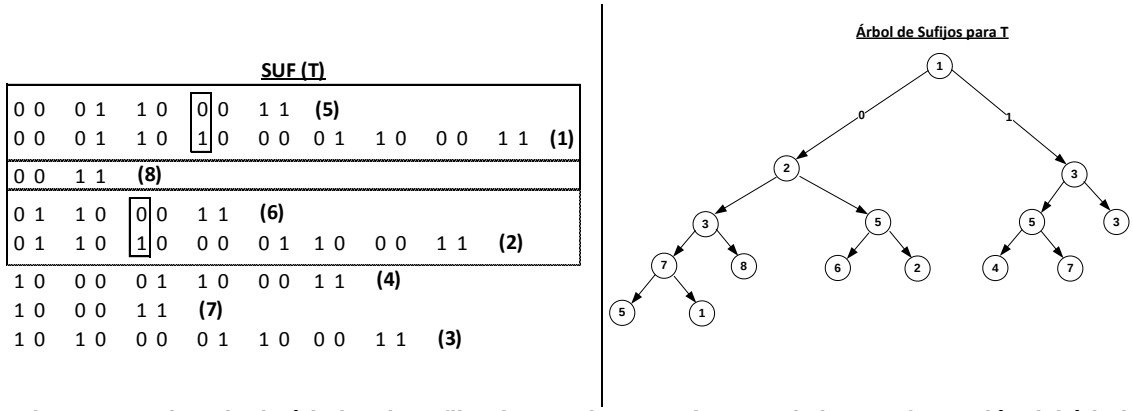

Figura 1.8 - Ejemplo de árboles de sufijos (Parte C). Segunda parte de la transformación del árbol de sufijos a un árbol de PATRICIA. Se arma la estructura de árbol con los códigos hallados [9].

De esta manera, se puede emplear este tipo de representación más versátil como índice de la base de datos de texto y realizar las búsquedas pertinentes.

1.2.4 Redes Bayesianas

Según [**14**], una red bayesiana es un grafo acíclico dirigido en el cual cada nodo representa una variable y cada arco una dependencia probabilística, en la cual se especifica la probabilidad condicional de cada variable dados sus padres. La variable a la que apunta el arco es dependiente (causa-efecto) de la que está en el origen de éste.

Para un mayor entendimiento de este concepto, se presenta uno de los ejemplos mostrados en [**15**]:

«Se tiene una alarma antirrobo instalada en una casa. La alarma se activa normalmente con la presencia de ladrones, pero también cuando ocurren pequeños temblores de tierra. Se tiene además, dos vecinos en la casa, Juan y María, que ha prometido llamar a la policía si oyen la alarma. Juan y María podrían no llamar aunque la alarma sonara (por tener música muy alta en su casa, por ejemplo). Incluso podrían llamar aunque no hubiera sonado (por confundirla con un teléfono)»

En la Figura 1.9 se muestra la red que corresponde a este ejemplo.

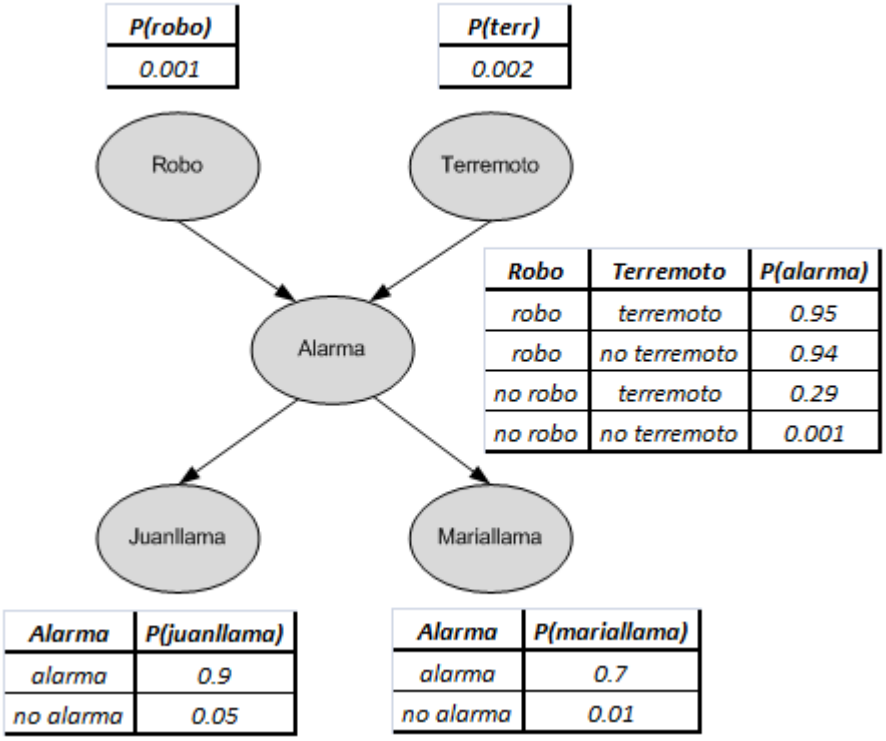

Figura 1.9 - Ejemplo de Redes Bayesianas [15]

Como se explica en [**15**], la topología de la red expresa que *Robo* y *Terremoto* son causas directa para *Alarma*. También, *Robo* y *Terremoto* son causas para *Juanllama* y para *Mariallama*, pero esa influencia sólo se produce a través de *Alarma*: ni Juan ni María detectan directamente el robo ni los pequeños temblores de tierra. En la red no se hace referencia directa, por ejemplo, a las causas por las cuales María podría no oír la alarma: éstas están implícitas en la tabla de probabilidades *P(Mariallama|Alarma)*.

1.2.5 Redes Neuronales

Según [**16**], desde el punto de vista computacional, una Red Neuronal Artificial (RNA) puede ser descrita como un conjunto de autómatas celulares (*neuronas*), por el cual se establece un flujo de información mediante una topología de interconexiones (*sinapsis*), tal como se aprecia en la Figura 1.10.

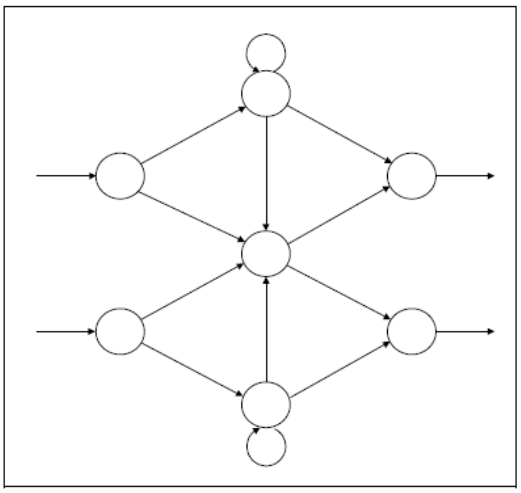

Figura 1.10 - Autómatas Celulares [16]

En la mayoría de las RNA propuestas, cada neurona procesa la información según el modelo propuesto por McCulloch-Pitts.

Como muestra la Figura 1.11, cada neurona obtiene su salida como la suma ponderada de cada una de sus entradas previa multiplicación por su correspondiente peso. Si dicho peso es positivo, se dirá que la entrada es excitadora; en caso contrario se hablará de inhibidora de acuerdo con el modelo biológico.

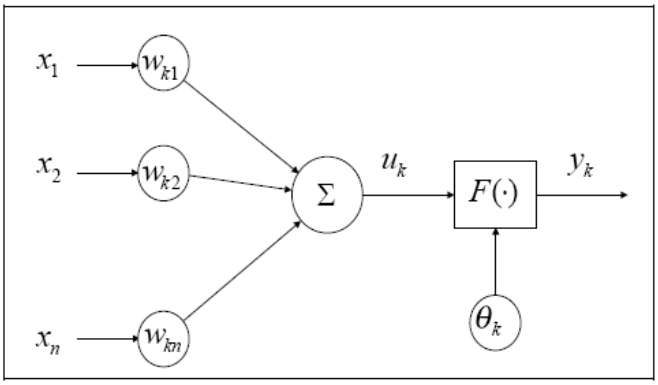

Figura 1.11 - Neurona Artificial (McCulloch - Pitts) [16]

Sobre las redes neuronales cabe destacar los siguientes tres aspectos: el aprendizaje mediante la adaptación de sus “pesos” sinápticos a cambios en el entorno; el manejo de información imprecisa, difusa, con ruido, y, en ciertas ocasiones basada en distribuciones de probabilidad; y la generalización de tareas o ejemplos desconocidos, a partir de otros que sí los son. Es gracias a estas características que las redes neuronales son ampliamente utilizadas en problemas de reconocimiento de patrones.

Asociado a cada tipo de red existe al menos un algoritmo de aprendizaje, el cual es básicamente un método sistemático para encontrar un valor adecuado de los pesos. En términos generales, se basan en definir, implícita o explícitamente, una función objetivo que represente de forma global el estado de la red. A partir de ella, los pesos asignados inicialmente van evolucionando a unos valores que lleven a dicha función a un mínimo (estado estable de la red). El aprendizaje, por tanto, es algo característico del tipo de red.

En el caso del reconocimiento de patrones, la tarea que se desea realizar consiste en poseer un número fijo de categorías en las cuales las muestras de entrada deben clasificarse. Para ello primero se requiere una fase de entrenamiento en la que se presenta a la red los patrones que debe aprender y la categoría en cual clasificarlo. Entonces se le presenta a la red un patrón nuevo y desconocido pero que pertenece a alguna de las categorías aprendidas y esta debe decidir a qué categoría se parece más. La ventaja de usar redes neuronales está en el hecho que se pueden separar regiones no lineales de decisión tan complicadas como se desee dependiendo del número de neuronas y capas. Por lo tanto, las redes neuronales artificiales sirven para resolver problemas de clasificación de alta complejidad.

Dentro de las redes neuronales, las más utilizadas son las redes con múltiples capas que funcionan hacia delante. Esta red está compuesta por un conjunto de nodos de entrada que componen la capa de entrada, un conjunto de una o más capas ocultas de neuronas y una capa de neuronas de salida. La señal de entrada se propaga hacia adelante desde la capa de entrada, por la oculta y hasta la salida; este tipo de configuración se conoce como MLP o “MultiLayer Perceptrons”.

El hecho de que este tipo de red se aplique para resolver con éxito multitud de problemas se debe a la utilización del algoritmo de aprendizaje que actualmente está más extendido, el algoritmo o regla back propagation, el cual es una generalización de la regla LMS “Least Mean Square”, por lo tanto también se basa en la corrección del error.

Básicamente el proceso *back propagation* consiste en dos pasadas a través de las diferentes capas de la red, una pasada hacia adelante y una pasada hacia atrás. En la

pasada hacia adelante, se aplica en la capa de entrada un patrón o vector de entrada, este propaga su efecto a través de las diferentes capas y como consecuencia produce un vector de salida. Durante este proceso, los pesos sinápticos de la red son fijos y no se modifican.

Durante la pasada hacia atrás en cambio, los pesos si se modifican de acuerdo con la regla de corrección del error.

La señal de salida real se compara con la señal deseada y como resultado se obtiene una señal de error, que se propaga en dirección contraria a través de la red modificando los pesos, de forma que, al volver a pasar el vector de entrada hacia adelante, la respuesta obtenida se asemeje más a la salida deseada.

Un ejemplo muy común de aplicación de este tipo de red es el que se presenta en [**17**] y que consiste en un sencillo problema de reconocimiento óptico de caracteres:

«Dado un panel de entrada compuesto por una matriz de 7x5 puntos, se consideran 12 clases diferentes donde se pretenden clasificar las muestras que se introducen. Los patrones que definen correctamente a cada una de las clases son los números del 0 al 9, el punto y el guión (Figura 1.12). Cuando a la entrada se presente una muestra distinta de los patrones correctos, el sistema presentará a su salida la información decodificada de la clase a la que pertenece la muestra, o bien, de la clase a la cual se aproxima más.»

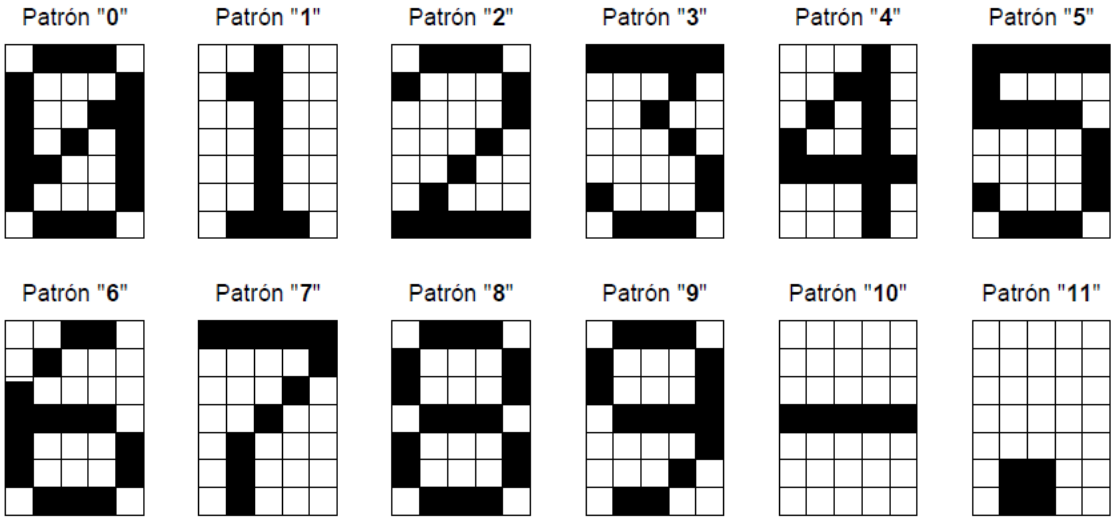

Figura 1.12 - Representación gráfica de los patrones de los caracteres numéricos en el formato de matriz de puntos 7x5 [17**]**

En base a este planteamiento, la red neuronal dispone de 35 entradas que se corresponden con los puntos de la matriz numerados en la Figura 1.13. El valor de cada entrada puede ser 0 si el punto es blanco y 1 si el punto es negro. Por otro lado, dispone de 12 salidas, una por cada clase. Cuando se introduzca una muestra a la entrada únicamente se activará la salida de la clase a la que pertenezca, permaneciendo las 11 restantes desactivadas con valores próximos a cero. Se considera que una salida está activada cuando su valor es próximo a la unidad.

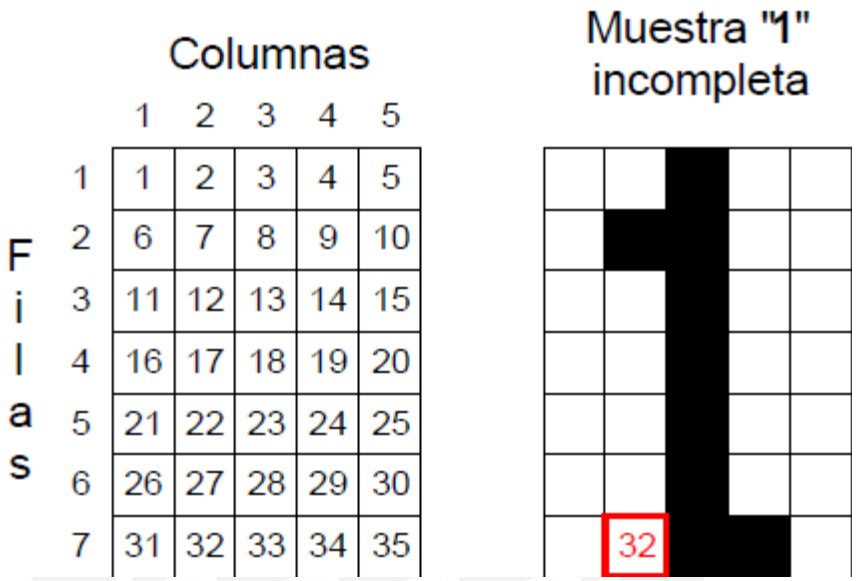

Figura 1.13 - Numeración de los puntos del display 7x5 y ejemplo de una muestra del patrón "1" con un punto erróneo [17]

Luego del entrenamiento correspondiente, se obtiene la tabla de resultados mostrada en la Figura 1.14, en la cual se puede apreciar que la red responde de manera casi ideal cuando se introduce un patrón sin error. De la misma manera, cuando se introduce una muestra con error en un punto fijo, la red clasifica perfectamente dicha muestra en la clase correcta. Mayor detalle sobre este ejemplo se puede obtener de [**17**].

Salida ideal.	Patrón "1" sin error.	Patrón "1" con error en el punto 32.
0	0.00000000038246	0.00000000022760
1	0.97094634494005	0.93598496351919
0	0.00483116118854	0.00425679820665
0	0.00000007267857	0.00000003653425
0	0.00001815880785	0.00001549628867
0	0.00000000010405	0.00000000001916
0	0.00000098764700	0.00000340726166
0	0.00038787588951	0.00012976360904
0	0.00016876462031	0.00008967151863
0	0.00127029941580	0.00051591379971
0	0.02144501839329	0.01084408021869
0	0.01863962414026	0.06933193518770

Figura 1.14 - Resultados [17]

1.2.6 Marco Legal del Plagio Documentario

Es necesario entender el marco legal asociado al problema que se aborda en el presente trabajo de tesis. Para ello, se hace una referencia a los conceptos relacionados con la propiedad intelectual y los derechos de autor. Con este objetivo, se empleará el material correspondiente a [**18**], según el cual, la propiedad intelectual comprende:

- **Propiedad Industrial:** marcas, patentes de invención, modelos de utilidad, diseños industriales. *Base legal: Decisión 486 y Decreto Legislativo 823.*

- **Derechos de autor:** protección de los creadores de obras. *Base legal: Decisión 351 y Decreto Legislativo 822.*

Para el caso particular del plagio, se centrará la explicación a la información relacionada con los derechos de autor.

Un documento muy importante en lo que respecto a derechos de autor se refiere, es el **Convenio de Berna para la Protección de las Obras Literarias y Artísticas**. El objetivo del convenio, como se explica en [**19**], es poder establecer las disposiciones que brinden el mínimo nivel de protección a una obra literaria o artística; para lo cual, el convenio se basa en tres principios fundamentales:

- **Principio de trato nacional.** Las obras originarias de uno de los Estados contratantes deberán ser objeto, en todos y cada uno de los demás Estados contratantes, de la misma protección que conceden a las obras de sus propios nacionales.
- **Principio de protección automática.** Para que la protección sea efectiva, no debe de cumplirse formalidad alguna previa.
- **Principio de independencia de la protección.** La protección es independiente de la existencia de protección en el país de origen de la obra**.**

Las condiciones mínimas de protección están referidas a las obras y los derechos que han de protegerse, así como la duración de la protección. Así mismo, los países considerados “países en desarrollo”, de conformidad con la práctica establecida por la Asamblea General de las Naciones Unidas, podrán, para ciertas obras y en ciertas condiciones, apartarse de esas condiciones mínimas de protección en lo que respecta al derecho de traducción y el de reproducción.

En el caso del Perú, el trámite de los derechos de autor se debe realizar ante la Oficina de Derechos de Autor del INDECOPI. A pesar de que no es obligatorio el registro, es altamente recomendable realizarlo, pues constituye un medio probatorio efectivo ante eventuales conflictos.

Algunos conceptos básicos e importantes a tomar en cuenta y que son explicados en [**18**], son:

- **Autor:** Persona natural que realiza la creación intelectual.
- **Obra:** Toda creación intelectual personal y original, susceptible de ser divulgada o reproducida en cualquier forma, conocida o por conocerse.
- **Originalidad de una obra:** La expresión (o forma representativa) creativa e individualizada de la obra, por mínimas que sean esa creación y esa individualidad. La obra debe expresar lo propio del autor, llevar la impronta de su personalidad.

La protección del derecho de autor recae sobre todas las obras del ingenio, en el ámbito literario o artístico, cualquiera que sea su género, forma de expresión, mérito o

finalidad. Dentro de los tipos de obras protegidas se consideran las expresadas en forma escrita: libros, revistas, folletos.

Cabe mencionar, que no se protegen las ideas, sino exclusivamente la forma de expresión mediante la cual las ideas del autor son descritas, explicadas, ilustradas o incorporadas a las obras.

En el mismo documento se explica que un autor posee tanto derechos morales, como patrimoniales. Dentro de los primeros se consideran: derecho de divulgación, derecho de paternidad, derecho de integridad, derecho de modificación o variación, derecho de retiro de la obra del comercio y derecho de acceso. En los segundos se tiene: **derecho de reproducción,** derecho de distribución al público de la obra, derecho de comunicación al público de la obra por cualquier medio y derecho de utilizar la obra bajo otras modalidades. En el caso de las obras ya divulgadas lícitamente, es permitida sin autorización del autor:

- La reproducción por medios reprográficos, para la enseñanza o la realización de exámenes en instituciones educativas. La reproducción por reprografía de breves fragmentos o de obras agotadas.
- La reproducción individual de una obra por bibliotecas o archivos públicos que no tengan directa o indirectamente fines de lucro.
- La reproducción de una obra para actuaciones judiciales o administrativas.
- La reproducción de una obra de arte expuesta permanentemente en las calles, plazas u otros lugares públicos, o de la fachada exterior de los edificios.
- El préstamo al público del ejemplar lícito de una obra expresada por escrito.

1.3 Plan del Proyecto

En la presente sección de describen todos los aspectos referidos a la realización y gestión del proyecto. Se indica cuál es la metodología que se seguirá, justificando adecuadamente el porqué de su elección. Asimismo, se esboza una lista de actividades (WBS) que deben llevarse a cabo para la correcta elaboración del proyecto.

1.3.1 Metodología de desarrollo del proyecto

La metodología seleccionada para el desarrollo del proyecto es SCRUM. Esta metodología ágil es aplicable para todo tipo de proyectos sin importar el tamaño. Asimismo, es fácil de comprender y adecuar a las características particulares de cada proyecto.

En la Figura 1.15 pueden visualizarse los principales componentes de esta metodología: el *product backlog* (que registra todos los requisitos que debe cumplir el sistema), el *sprint backlog* (que indica los requerimientos a implementar en un sprint determiando – más adelante se explicará, en detalle, el término *Sprint*), las *daily scrum meetings* (que permiten elaborar los sprint backlog a partir del product backlog) y los productos entregables (que se muestran luego de cada sprint).

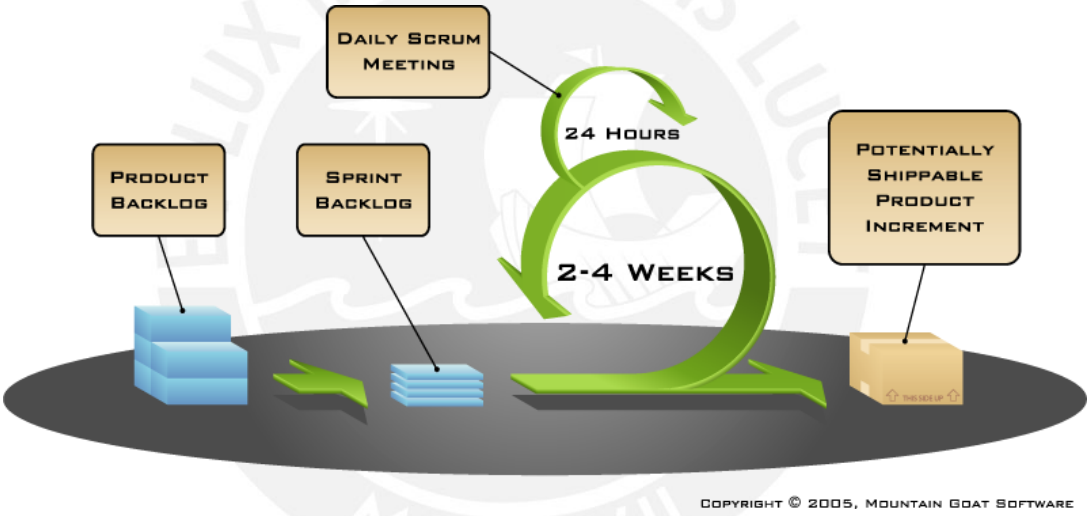

Figura 1.15 - Proceso SCRUM [20**]**

Algunas de las características principales que posee esta metodología y, por las cuales, fue seleccionada para el desarrollo de este proyecto son:

- **Rápida respuesta al cambio.** Por la naturaleza del proyecto, si bien pueden existir requerimientos iniciales bases, a medida que se va investigando tanto sobre el método a aplicar para solucionar el problema, como las tecnologías disponibles para poder implementar la solución, se incorporan requisitos adicionales. Es por ello que se requiere de una metodología que no se encuentre estrechamente fundamentada en la lista de requisitos que se plantean en un inicio y que pueda

responder al cambio fácilmente. SCRUM permite responder de forma casi natural a los cambios constantes en el proyecto.

- **Enfoque centralizado en mostrar resultados constantemente.** Una forma de visualizar el avance del proyecto es a través de la presentación de pequeños prototipos del sistema de forma periódica. Esto no sólo permite avanzar con el desarrollo del proyecto de forma centralizada en implementar funcionalidades, sino que sirve de motivación para quien lo desarrolla, pues puede ver, constantemente, los frutos de su trabajo. Uno de los fundamentos de SCRUM es presentar demos funcionales cada cierto período, pues, como todo proceso ágil, se centra en liberar software utilizable frecuentemente.

Sin embargo, dado que el proyecto que se plantea es individual, no es posible aplicar todos los conceptos que plantea la metodología. A continuación se listan los componentes de SCRUM que serán empleados y de qué manera:

- **Scrum team – Scrum master.** Para el caso del proyecto, el que desarrolla la investigación y la implementación del sistema actuará con ambos roles.

- **Product owner.** Este papel será jugado en parte por quien desarrolla el sistema, dado que por su investigación es quien conoce mejor las características del mismo y, por otra parte, por el asesor, dado que es quien conoce los estándares académicos que el proyecto debe cumplir.

- **Product backlog.** Es la lista priorizada de funcionalidades que el sistema debe cumplir. Se puede visualizar de forma general en el capítulo 2, sección 2.

- **Sprint planning meeting – Sprint backlog – Sprint review meeting.** Un *Sprint* constituye un periodo de tiempo durante el cual se implementarán algunas funcionalidades listadas en el *Product backlog*. Las *Sprint planning meetings* son reuniones en las cuales se identifica cuáles son las funcionalidades que se van a desarrollar en el *Sprint* (se registran en el Sprint backlog) así que se estima cuándo se presentará la Demo del *Sprint*.

Para el caso del proyecto, se plantea que cada *Sprint* tendrá una duración fija de 1 semana (7 días), luego de la cual (día 8), se presentará al demo con el avance realizado. El número de *Sprints* a realizar depende de la complejidad de las funcionalidades, la prioridad de las mismas y los errores que se obtengan. En el mismo día de la presentación de la demo, se realizará el *Sprint review meeting*, que básicamente consistirá en analizar si la estimación de tiempos fue correcta y cómo debe corregirse.

- **Daily scrum.** Reuniones diarias para determinar las tareas que se realizarán cada día. Esto será desarrollado de forma personal por quien implementa la solución.

La metodología permite determinar algunos de los anexos a presentar. En este caso, el *Product backlog* y los *Sprint backlog* constituirán anexos..

1.3.2 Planificación de tareas

Por la metodología a emplear, no es posible estimar desde un inicio la duración de todo el proyecto ni específicamente qué tareas se van a realizar en cada *Sprint*, dado que eso dependerá de las funcionalidades que sean seleccionadas y las estimaciones de tiempo que se realicen en cada *Sprint planning meeting*.
Sin embargo, es posible listar cuáles son las tareas principales que se realizarán a lo largo de todo el proyecto, aunque no se pueda determinar cuándo exactamente. Para ello, se presenta el diagrama de Estructura de Descomposición del Trabajo (EDT o WBS) en la Figura 1.16.

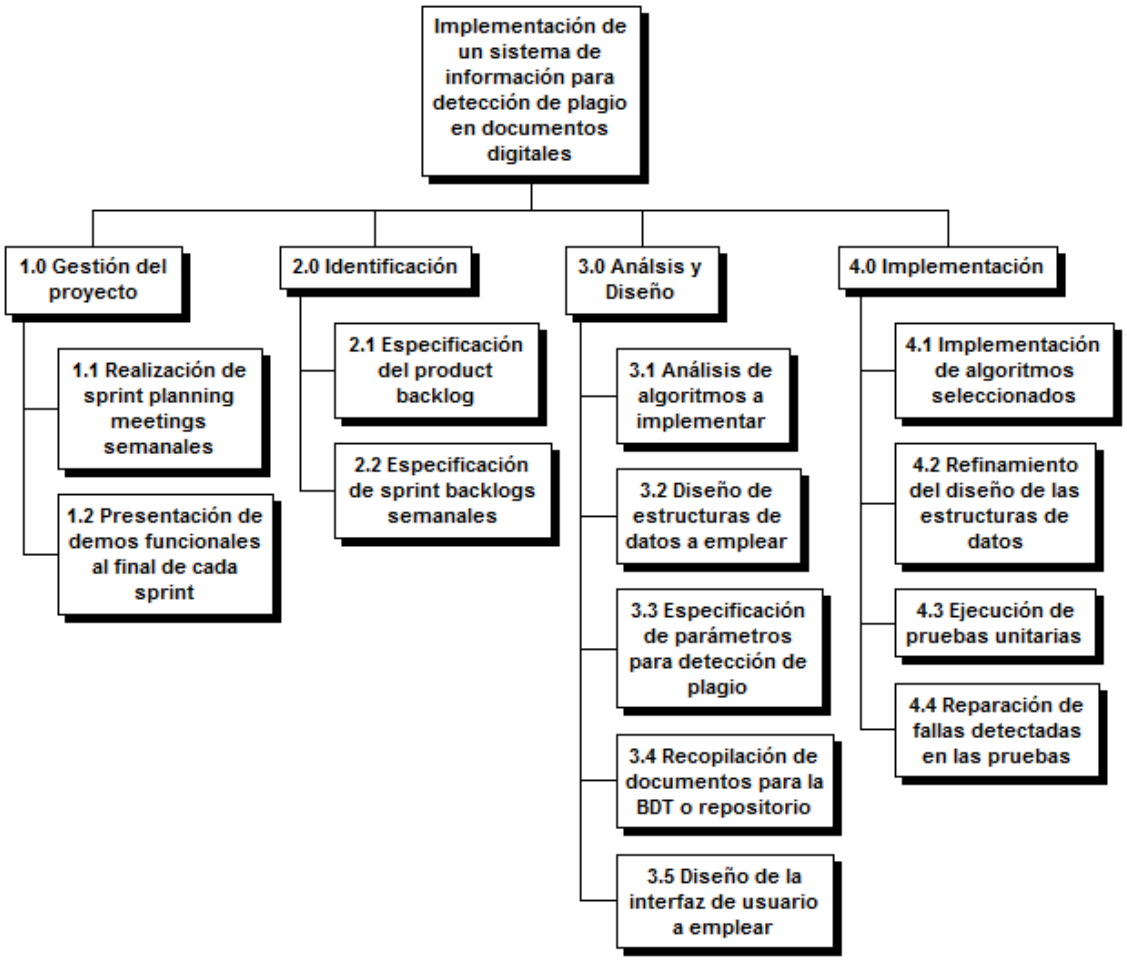

Figura 1.16 - Estructura de Descomposición del Trabajo

La sección relacionada con la Implementación de la solución empleará una metodología apropiada para el mismo, la cual será descrita en el capítulo 2.

1.4 Estado del Arte

En esta sección se presente un análisis detallado del estado del arte, correspondiente a herramientas existentes en el mercado, así como métodos propuestos por investigadores, para poder determinar el nivel de similitud que poseen dos documentos al ser comparados.

1.4.1 Definición del problema de plagio documentario

En una sección anterior, ya se ha analizado el problema del plagio de una manera lo suficientemente detallada para darse cuenta del impacto negativo que puede presentar. De acuerdo al alcance del presente proyecto, el plagio será tratado a nivel

documentario. Se entiende como plagio documentario a la copia parcial o total de documentos de texto, sin mencionar al real autor de los mismos y haciendo pasar las ideas ahí vertidas como propias.

Este tipo de plagio es uno de los más comunes, y en el que se puede incurrir tanto de manera intencional o como no intencional.

Debido a que es el tipo de plagio que mayor se ha extendido en la comunidad, existen una serie de herramientas desarrolladas para combatirlo, las cuales van desde herramientas legales como tecnológicas. En las siguientes secciones se presentará una mayor descripción de las mismas.

1.4.2 Métodos actuales de solución de problema: *Document Fingerprinting*

El método que se aplica de manera más común actualmente, es el que consiste en la obtención de una firma o *fingerprint* de un documento. En esta sección se procederá a explicar en qué consiste este método.

La *fingerprint* (huella digital o firma) de un documento se refiere al conjunto de valores numéricos que representan a varias porciones del mismo [**21**]. El proceso para obtener la *fingerprint* de un documento consiste de los siguientes pasos:

1. Se divide el texto en pedazos o particiones (*chunks*), que se consideren lo suficientemente significativas para poder captar el significado de lo que se está escribiendo. Estas particiones pueden ser oraciones completas, conjunto de palabras contiguas, conjunto de caracteres contiguos, entre otros.
2. A cada una de estas particiones se le aplica una función que permita obtener un valor numérico único. Para ello, generalmente se aplica una función de tipo hash.
3. De todo el conjunto de valores hash obtenidos se selecciona un subgrupo que va a ser el que va a representar a todo el documento en el proceso de comparación.
4. Se elabora un índice invertido con las *fingerprints* obtenidas. Para poder encontrar todos los documentos que poseen alguna similitud con un documento A, primero se leen todas las listas invertidas de las *fingerprints* de A; posteriormente se mezclan estas listas para, finalmente, aplicar alguna regla de verificación especificada.

Como se puede apreciar, existen dos decisiones muy importantes que se deben de realizar para asegurar el buen funcionamiento del método. En primer lugar, se debe de seleccionar el tamaño de la partición que se va a tomar en cuenta en el paso 1. Un tamaño muy grande (como el de todo el documento) permitiría únicamente detectar copias exactas, mientras que un tamaño muy pequeño (como el de una palabra) terminaría indicando que todos los documentos son copias entre sí. En segundo lugar, el seleccionar el subconjunto de valores hash que sean lo suficiente representativos de todo el documento acarrea otro "problema". Se debe seleccionar un número lo suficientemente grande como para que permita comparar un alto número de oraciones y, así, incrementar la efectividad, pero a su vez, debe ser lo suficientemente pequeño como para que pueda ser almacenado apropiadamente y aún así ser significativo.

Como se indica en [**21**], los métodos de *fingerprinting* se pueden clasificar en dos tipos: *fingerprinting* ***con sobre-posición*** y ***sin sobre-posición***. Esta división se realiza dependiendo de cómo los métodos seleccionan los grupos de caracteres en el paso 1. Los métodos con sobre-posición se caracterizan porque emplean una especie de "ventana deslizante", la cual cambia por cada palabra, generalmente, y una secuencia de palabras en la ventana es considerada como una partición; logrando que las particiones consecutivas se sobrepongan entre sí. Uno de estos métodos es el *Winnowing* [**22**]. Por otro lado, los métodos sin sobre-posición, dividen al texto en pequeños segmentos significativos, como frases u oraciones, en lugar de generar muchas secuencias de palabras. Uno de estos métodos es el *Hash-breaking* [**23**].

En lo relacionado al paso 2, cada método emplea una forma diferente de selección. En el caso de *Winnowing*, por ejemplo, se establece un tamaño fijo de ventana sobre el número de particiones que se han generado y se selecciona aquella partición cuyo valor hash sea el menor en la ventana.

El método de *fingerprinting* es eficiente, en el sentido que permite almacenar una pequeña porción del documento para el proceso de comparación, a la vez que, al no guardar el texto en sí, permite que los sistemas que lo implementen no puedan ser empleados como fuente de plagio en sí mismas. Sin embargo, posee la enorme desventaja de ser susceptible a pequeños cambios en la estructura de las oraciones (cambiar la posición del sujeto dentro de la oración, por ejemplo) o en el contenido de las mismas (empleo de sinónimos o cambio de tiempo verbal de la oración).

1.4.3 Herramientas existentes en el mercado – índole comercial y de investigación

Como se explicó en la sección 2, el plagio es un problema que afecta a personas e instituciones de todas partes del mundo. Por tanto, alrededor del globo se han desarrollados diferentes herramientas que permiten detectar este tipo de delito. Debido a que existen una gran cantidad de aplicaciones en el mercado que ofrecen el servicio de comparación de documentos en busca de plagio, en [**24**] los autores realizan una clasificación de las diferentes herramientas disponibles y que son de uso popular. Las herramientas consideradas son tanto las empleadas para detección de plagio en documento de texto libre, como en los de código fuente. Las características tomadas en consideración para la comparación son de tipo técnico: si se emplean para detección en texto libre, si se emplean para detección en código fuente, si son intra-coporales o extra-coporales, si se emplean de forma local o basadas en Web, si son de tipo público o privado, entre otras características que se pueden apreciar en el cuadro mostrado en la Figura 1.17.

	Características															
Nombre de la Herramienta	**Código Fuente**	**Texto Libre**	**Intra-corporal**	**Extra-corporal**	**Local**	**Basada en Web**	**Pública**	**Privada**	**Arreglos Especiales**	**Pasado**	**Actual**	**Tokenización**	**Singular**	**Pareado**	**Estructural**	**Superficial**
Big Brother	X	X	X		X				X		X	X		X	X	
CopyCatch		X	X		X				X		X			X		X
CopyFind		X	X		X				X		X			X	X	
DetectaCopius	X		X		X		X				X			X		X
EduTie		X	X	X		X			X		X			X	X	
EVE2		X		X	X				X		X			X		X
Ferret		X			X		X				X			X		X
Gossip		X	X	X	X				X		X			X		X
iParadigms (TurnItIn.com)		X	X	X		X			X		X			X	X	
Jones	X		X		X			X			X	X	X			X
JPlag	X		X			X				X		X		X	X	
MOSS	X		X			X					X	X		X	X	
MyDropBox (PlagiServe)		X		X		X	X				X			X	X	
Saxon	X		X		X				X		X			X	X	
SHERLOCK	X	X	X				X				X	X		X	X	X
Tetlow	X		X		X				X		X	X		X	X	
TRANKER		X	X		X				X		X			X		X
WORDCheck		X	X		X				X		X			X		X

Figura 1.17 – Clasificación de herramientas para detección de plagio [24**]**

De la misma manera, en [**25**] los autores realizan otro análisis sobre las herramientas más conocidas empleadas para detección de plagio. Este cuadro se puede visualizar en la Figura 1.18

Atributos	Herramientas para Detección							
	Herramientas Específicas							*Herramientas Alternativas*
	Turnitin	Eve2	CopyCatchGold	WordCheck	Glatt	Moss	Jplag	Google Yahoo Altavista
Tipo de herramienta de texto sobre la que opera								
¿Verifica código fuente?	-	-	-	-	-	X	X	-
¿Verifica texto libre?	X	X	X	X	X	-	-	X
Tipo de herramienta de corpus sobre la que opera								
¿Opera intra-coporalmente?	X	-	X	X	-	X	X	-
¿Opera extra-corportalmente?	X	X	-	-	-	-	-	X
Otros atributos								
¿Diseñado para uso de estudiantes?	X	-	-	-	-	-	-	X
¿Diseñado para uso de profesores?	X	X	X	X	X	X	X	X
¿Respuesta automática?	-	X	-	X	-	-	-	X
¿Libre?	-	-	-	-	-	X	X	X

Figura 1.18 - Comparación de atributos de herramientas para detección de plagio [25**]**

En primer lugar, se revisarán las principales características de algunas de las herramientas señaladas arriba, empleando para ello lo presentado en [**26**].Las herramientas a describir son: Findsame (Digital Integrity), Eve2 (CaNexus), Turnitin (iParadigms), CopyCatch (CFL Software Developments) y WordCHECK (WordCHECKsystems).

- **Findsame**

  Herramienta diseñada para detectar material que ha sido “cut and paste” de Internet. Es un servicio provisto por Digital Integrity (DI) y basado en tecnología MOSS. Es una herramienta de búsqueda de contenido basado en Web y genera reportes de manera casi instantánea.

- **Eve2**

  Son las siglas para Essay Verification Engine. Es una herramienta de búsqueda que ejecuta búsquedas complejas para encontrar material de Internet. No compara documentos por pares y no emplea una base de datos propietario. Tampoco permite rastrear ensayos a bancos de ensayos, ni tampoco documentos que no se encuentren en formato html. A pesar de que Eve2 es una herramienta que se

ejecuta en una PC, es una interface que realiza búsquedas en la Web. Esta aplicación no puede ser instalada en red para múltiples usuarios.

- **Turnitin**

  Es una herramienta diseñada para detectar material cortado y pegado de Internet y otros almacenados en una base de datos. Es un servicio ofrecido a la comunidad académica para detectar plagio en asignaciones escritas. Esta herramienta detecta material copiado de Internet, así como una colusión entre estudiantes a través de una verificación cruzada de ensayos registrados unos contra otros, y también comparando con una base de datos interna de textos.

- **CopyCatch**

  CopyCatch es el elemento de diagnóstico de de un set de herramientas de análisis de texto, Vocalyse Toolkit, desarrollado en asociación con miembros del Grupo de Corpus Lingüístico Forense en la universidad de Birmingham. CopyCatch detecta colusión entre estudiantes comparando documentos enviados y calculando la proporción de palabras que se poseen en común. También es posible una comparación a nivel de frases. Esta herramienta no detecta plagio de Internet.

- **WordCHECK**

  Esta herramienta detecta colusión entre estudiantes mediante la comparación de perfiles de palabras clave entre un documento enviado y otros almacenados en un archivo interno. La aplicación no permite la detección de plagio por Internet, ni tampoco comparación a nivel de frases.

A continuación, se revisarán las características de algunas herramientas desarrolladas en universidades y lo métodos que emplean para la detección de plagio.

- **COpy Protection System (COPS)**

  Este sistema puede detectar no sólo copias exactas sino también parciales. Permite que los documentos sean registrados vía email en formato TEX (incluido LATEX), DVI, troff y ASCII. Nuevos documentos pueden ser tanto registrados en el sistema como probados contra los que ya encuentran registrados. Si un nuevo documento es probado, se muestra un resumen que lista todos los documentos

registrados que viola. Los módulos principales de COPS se pueden ver en la Figura 1.19.

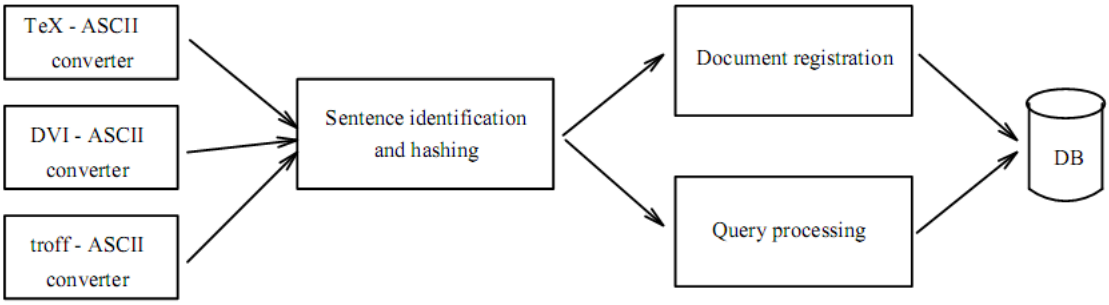

Figura 1.19- Módulos del Sistema COPS [23**]**

Los módulos de COPS son fácilmente reemplazables, permitiendo así experimentar con diferentes estrategias de particionamiento y funciones de decisión. En su forma original, COPS emplea una estrategia de particionamiento basada en oraciones.

Para explicar el funcionamiento de COPS, se empleará lo descrito en [**23**]. A cada documento que es presentado al sistema se le otorga un ID único, el cual se emplea para indexar una tabla con información de los documentos como título y autor. Para que el documento pueda ser registrado primero debe ser convertido a su forma canónica, es decir texto ASCII plano. El proceso mediante el cual esto ocurre depende del formato del documento. Luego de haber transformado el documento se procede a determinar en valor hash correspondiente a las oraciones individuales del documento. Usando puntos, signos de exclamación e interrogación como delimitadores de oraciones, se aplica una función hash a cada oración y se obtiene una llave numérica. El ID único actual del documento es luego almacenado en una tabla hash permanente, una vez por cada oración. Cuando se desea verificar un nuevo documento contra los que ya se encuentran registrados, se emplea un procedimiento bastante similar. Se genera el texto ASCII plano, se determinan las oraciones, y se genera la lista de llaves hash, y se las busca en la tabla hash. Si más de un determinado número de oraciones coincide con un documento determinado se reporta una posible violación.

- **Stanford Copy Analysis Mechanism (SCAM)**

  SCAM es un servidor de registro de documentos desarrollado por un grupo de investigación en la universidad de Stanford. Este servidor puede detectar duplicados en una variedad de formas y permite identificar tanto solapamiento total como parcial, este último empleando un nivel de comparación definido por el usuario. Posee un repositorio en el cual se registran artículos de diferentes fuentes (listas de correo electrónico, grupos de Internet, entre otros) y que se emplea luego para la detección de plagio. Por ejemplo, fue usado para poder determinar si los artículos que se enviaban a un Journal determinado poseían algún tipo de similitud con otros ya registrados en años anteriores y que se encontraban en el repositorio [**27**]. Uno de los principales objetivos del desarrollo de SCAM fue poder mejorar aquellas deficiencias que poseía COPS en su implementación, como es que el empleo de oraciones como unidad de comparación no permite detectar solapamiento parcial de oraciones. En contraposición, SCAM emplea una técnica de particionamiento (*chuncking*) basada en palabras. El sistema funciona básicamente de la siguiente manera: los documentos que van a ser registrados son particionados e insertados en el repositorio; los nuevos documentos que llegan son particionados en las mismas unidades y son comparados con los documentos pre-registrados para verificar solapamiento. Una de las características principales de SCAM es el esquema de almacenamiento que emplea, el cual se base en un índice invertido. Como se indica en [**28**] se emplea una estructura de índice invertido para almacenar las particiones de los documentos registrados. Un índice de particiones en el vocabulario es construido y mantenido en el momento del registro. Cada entrada para una partición apunta a un grupo de registros que indican los documentos donde la partición se ubica. Cada registro para una partición dada w_i posee dos atributos (*numDoc*, *frecuencia*), donde *numDoc* es un identificador único de un documento registrado, y *frecuencia* es el número de ocurrencias de w_i en el documento con ID *numDoc*. Cuando un documento *D* va a ser comparado con los documentos pre-registrados, las particiones de *D* son buscadas en el índice de los documentos registrados. Esto significa que sólo los documentos que se solapan al nivel de las particiones serán considerados usando este mecanismo de índice. Por tanto, el número total de búsquedas en el índice es el número de particiones distintas que ocurren en el documento *D*. Para poder determinar si existe solapamiento entre un documento en el repositorio y el que se está ingresando, se

utiliza una medición denominada *modelo de frecuencia relativa*, el cual emplea frecuencias relativas de palabras como indicadores del uso de palabras similares y lo combina con la medida de similitud del coseno (una medida muy conocida en IR, para más detalle ver [**28**]). El mecanismo empleado de particionamiento por palabras permite reducir el número de falsos positivos.

- **CHECK**

  Es un sistema que permite el reconocimiento de solapamiento total y parcial de documentos. Utiliza un mecanismo similar al de SCAM en el sentido que los documentos son registrados en un servidor central y contra el cual se comparan los documentos prueba que se deseen. Para poder describir al sistema CHECK, se emplea lo escrito en [**29**]. La mayor diferencia de CHECK es que trata de emplear la semántica del documento para poder obtener una mejor y más rápida respuesta. Cuando un documento potencialmente plagiado es comparado contra un documento registrado, se emplean técnicas de recuperación de información para pre-procesar los documentos y determinar el significado semántico de los mismos. Si los documentos son de diferentes temas, futuras comparaciones entre los documentos podrían ser eliminadas. Este razonamiento es empleado de manera recursiva para estructuras organizativas individuales de diferentes granularidades incluyendo secciones, sub-secciones, sub-sub-secciones y párrafos. En otras palabras, se aplican de manera recursiva técnicas de recuperación de información a cada estructura organizativa individual hasta encontrar 2 párrafos que están altamente relacionados semánticamente. Luego estos párrafos son comparados en detalle para determinar si se solapan de manera sustancial. Mediante el estudio de la semántica de los documentos adicionalmente a su sintaxis, se provee un nivel extra de seguridad puesto que los usuarios ya no podrían esquivar el mecanismo de detección simplemente cambiando pequeños detalles a cada oración (como cambiar el tiempo de los verbos de presente a pasado) ya que la semántica de los documentos se mantendrá mayormente intacta.

  Como se puede observar en la Figura 1.20, el sistema está compuesto de tres módulos principales: registro de documentos, comparación de documentos y parseo de documentos. El primer módulo se encarga de registrar un documento original en el servidor de base de datos, el cual mantiene el catálogo de documentos considerados originales. El segundo módulo compara un documento

ingresado contra los registrados para detectar cualquier posible plagio. El módulo de parseo de documentos construye una estructura de índice interna, llamada característica estructural para cada documento a ser usado por los módulos de registro y comparación.

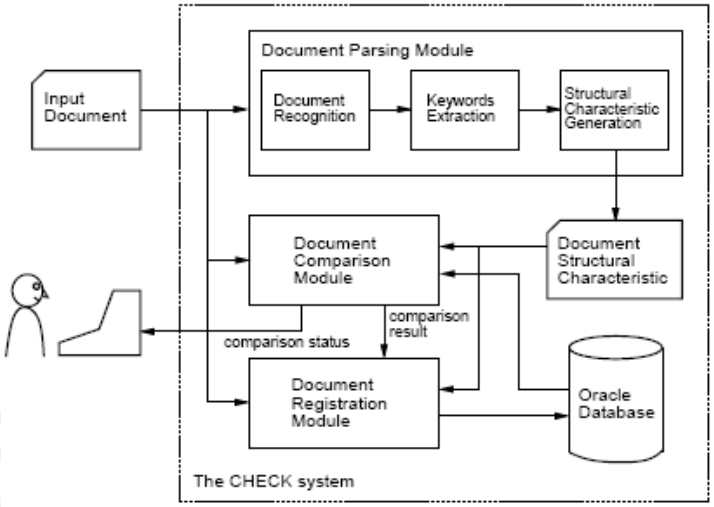

Figura 1.20 - Arquitectura de CHECK [29**]**

El sistema posee tres funciones: registro de documentos originales (no hay comparación con los ya registrados), verificación de documentos (no hay registro del mismo en la BD) y registro normal de documentos (incluye la verificación del documento contra los registrados y en caso no haya señales de plagio se procede a registrarlo con un documento original).

- **MatchDetectReveal (MDR)**

  MDR es un proyecto realizado conjuntamente entre la Universidad de Monash y la Universidad de Alicante. Este proyecto estudia diferentes maneras de identificar solapamiento entre documentos digitales. Diferentes métodos empleados en la actualidad son comparados basándose en diferentes atributos como son precisión, desempeño y grado de protección. Así mismo, se propone un novedoso método que consta de dos etapas. Como se describe en [**30**], en la primera fase se selecciona un grupo de documentos candidatos aplicando métodos de división e indexación para luego, en la segunda fase, emplear métodos de comparación exacta basados en árboles de sufijos.

El proceso, se inicia con la formación de un repositorio de documentos que se será empleado para la comparación. Al momento del registro, los documentos sufren un pre procesamiento para poder ser indexados en el repositorio. En este pre procesamiento, los documentos adquieren el formato adecuado para que puedan ser empleados luego por el mecanismo de comparación. Cuando se ingresa un documento sospechoso que desea ser analizado, el mecanismo de búsqueda selecciona varios documentos candidatos del repositorio para ser comparados con el sospechoso empleando el mecanismo de comparación. Empleando árboles de sufijos y un algoritmo de comparación estadística, el mecanismo de comparación indica el nivel de solapamiento o similitud de los documentos.

Si bien la arquitectura del MDR, que puede apreciarse en la Figura 1.21, está adaptada para aplicaciones de detección de plagio y copias, sus componentes principales pueden ser empleados para otros propósitos como referencias cruzadas y comparación de diferentes versiones de un mismo documento.

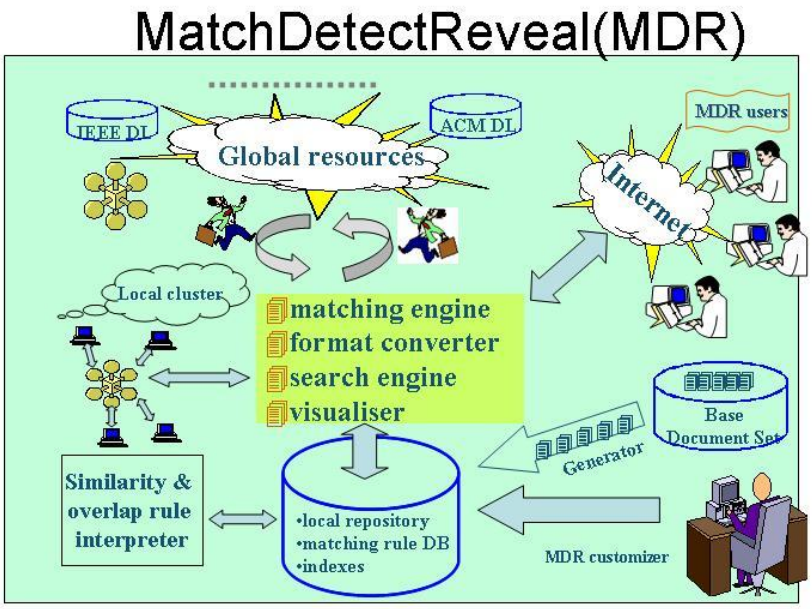

Figura 1.21 – Arquitectura del MDR. Muestra los diferentes componentes involucrados en el sistema. [30]

1.4.4 Investigaciones sobre mejoras en algoritmos o estructuras de datos

Como se ha podido notar en la sección anterior, existen diversas estructuras de datos y algoritmos que son empleados para este tipo de aplicaciones; es así, que también se

realizan investigaciones respecto a mejoras que se pueden realizar para el proceso de búsqueda y detección plagio. Algunas de las investigaciones son:

- **Eficiencia de estructuras de datos para detectar solapamiento en documentos digitales** [**31**].

  En este documento se analiza la eficiencia del uso de funciones de hash empleadas para reducir el espacio en memoria empleado en los índices de las particiones. Debido a que es posible que las funciones de hash devuelvan un mismo valor para diferentes particiones, se puede incurrir en comparaciones falsas. Por tanto, se propone un algoritmo que emplea árboles de sufijos para evitar esas falsas comparaciones. Sin embargo, el empleo de árboles de sufijos conlleva también a un incremento en el espacio empleado para el almacenamiento de los índices. Es por ello, que los autores sugieren el empleo de un tipo especial de árbol de sufijos en el cual sólo se consideran particiones que comienzan al inicio de las palabras. Asimismo, muestran cómo, si se transformara un árbol de sufijos a un grafo dirigido acíclico, los requerimientos para espacio de almacenamiento también se verían reducidos.

- **Reduciendo los requerimientos de espacio de los árboles de sufijos** [**32**]

  En este documento, los autores muestran cómo, a pesar de proveer accesos eficientes a todas las sub-cadenas de una cadena determinada, y que pueden ser construidos y representados en un espacio y tiempo linear, los árboles de sufijos almacenan gran cantidad de información redundante. Los autores explotan estas redundancias para poder obtener representaciones más eficientes en cuanto al uso del espacio de almacenamiento.

Como se puede apreciar, las investigaciones realizadas para mejoras en las estructuras de datos y algoritmos a emplear, permiten establecer una mejor manera de adoptar dichas estructuras (o quizás sus modificaciones) al momento de decidir qué algoritmos o estructuras de datos son los más recomendables para un determinado problema.

1.4.5 Investigaciones realizadas en la PUCP

En el caso de investigaciones realizadas por alumnos de la universidad, no existe alguna que esté directamente relacionada con el problema que se describe; sin embargo, vale la pena mencionar un proyecto de fin de carrera de la especialidad de Ingeniería Electrónica. Este proyecto, si bien no enfrenta el problema del plagio, es de gran ayuda para el establecimiento de posibles mejores estructuras de datos y algoritmos a emplear para contrarrestar el problema que se plantea.

El proyecto en cuestión se titula ***Comprensión y generación de lenguaje natural en un sistema de diálogo usando inteligencia artificial para servicios telefónicos de información de cines*** [**33**]**.**

El proyecto que expone es un CALL-CENTER para atención de clientes de cadenas de cines. Muchas veces, los clientes requieren conocer cierto tipo de información relacionada, no sólo con las películas que van a ser transmitidas y sus horarios, sino también sobre promociones, consulta de precios, reclamos, entre otros. Para poder atender éstos pedidos a través del teléfono, el autor propone el empleo de un sistema que pueda responder adecuadamente a las consultas que los clientes realizan por teléfono. Expone la implementación de un sistema que empleando fundamentos de sistemas expertos y manejo de árboles sintácticos, sea capaz de cumplir tal labor.

Este proyecto guarda cierta relación con la herramienta que se desea implementar, en el sentido que estudia formas para analizar la semántica de oraciones o textos. Este tipo de análisis puede ofrecer una mejora en el método empleado para detección de plagio, tal y como se explica en mayor detalle en la siguiente sección.

1.4.6 Nuevas Tendencias en la Detección Automática del Plagio en Documentos Digitales

Si bien las técnicas actuales permiten detectar la gran mayoría de formas de plagio en documentos digitales mediante la comparación directa de los párrafos y cadenas de texto, poseen aún una deficiencia, y es que centralizan su método de búsqueda en la estructura sintáctica de las oraciones y no en la semántica. Esto no permite que se puedan detectar infracciones de plagio más sutiles como es mediante el parafraseo de oraciones. Es por ello, que tendencias más modernas de métodos para la detección de

plagio se inclinan hacia el uso de técnicas de Procesamiento de Lenguaje Natural (PLN) puesto que permiten emplean la estructura semántica de los documentos.

Una de estas investigaciones la podemos ver en [**34**], en donde el autor describe un método para detección de plagio empleando *Latent Semantic Analysis,* que es una técnica de procesamiento de lenguaje natural, que incorpora SVD (*Singular Value Decomposition*) para analizar relaciones entre un conjunto de documentos y sus términos.

De la misma manera, en [**35**], se indica que la estructura sintáctica subyacente y el significado sintáctico de dos oraciones pueden ser comparados para revelar su similitud. En este documento se describe un método en el cual se emplea la gramática libre de contexto, la gramática de casos y la gramática léxico-funcional para representar la estructura sintáctica y el significado semántico de las oraciones y muestra como este nuevo acercamiento permite detectar casos de plagio que no pueden ser identificados por los métodos tradicionales. Estas recientes investigaciones motivan a la realización de trabajos futuros para el mejoramiento del sistema que se describe en este documento.

1.5 Descripción y Sustentación de la Solución

Para poder solucionar este problema, se han desarrollado varias herramientas que permiten combatir este problema para evitar las nefastas consecuencias que puede traer. El Estado del Arte ha hecho posible notar que ya existen en el mercado distintas herramientas que permiten realizar el tipo de labores que se describe. Sin embargo, cada una de ellas poseen limitaciones propias que pueden no convertirlos en los más apropiados candidatos a ser un utilizados en un contexto universitario como el que se presenta en la PUCP.

Es por ello que se plantea el desarrollo de esta herramienta que permita lograr la detección de plagio en las diferentes asignaturas que se posean en las diferentes especialidades de la universidad. La herramienta permitirá, entre otras cosas, el almacenamiento y/o registro de trabajos o asignaciones de semestres académicos anteriores, los cuales formaran parte del repositorio de documentos, contra los cuales se compararán los nuevos documentos que los estudiantes presenten. La selección

sobre contra qué grupos de archivos se realizará la comparación, quedará enteramente a decisión del usuario que emplee la herramienta. De la misma manera, los criterios de evaluación del nivel de plagio o copia de documentos, será a su vez establecido de manera apropiada dependiendo de cada contexto académico. Estas características son las que permiten la diferenciación entre el sistema que se plantea desarrollar, y otros que ya existen en el mercado. La personalización de acuerdo a las necesidades de cada facultad, curso o profesor es otro elemento diferenciador que posee la aplicación a implementar.

Como ya se ha podido notar en secciones anteriores, el combatir el problema del plagio es una cruzada en la que muchos profesionales se están uniendo, debido a que su expansión no debe ser permitida, y se están desarrollando grandes esfuerzos por contrarrestarla. Entonces no hay duda de que una herramienta que permita combatir dicho problema se encuentra con un gran soporte para el éxito y la utilidad en el contexto académico actual. Es así, que el sistema que el autor plantea realizar no solo se justifica como un proyecto de fin de carrera, sino que también como un elemento para combatir uno de los grandes males del mundo académico actual.

"El éxito no se logra sólo con cualidades especiales. Es sobre todo un trabajo de constancia, de método y de organización."

J.P. Sergent

2 Análisis y Diseño

En este capítulo se desarrollaran los aspectos relacionados con el análisis y diseño de la solución que se propone. En primer lugar, se indicará la metodología a emplear para la implementación de la herramienta, especificando las adecuaciones necesarias para que sea aplicable para el proyecto de tesis. Posteriormente, se realiza el análisis de la solución, teniendo como eje central el estudio de viabilidad del sistema. Finalmente, se señalan los aspectos de diseño que se tendrán en consideración para la elaboración del sistema.

2.1 Metodología Aplicada para el Desarrollo de la Solución

La metodología seleccionada para el desarrollo de la solución es EXtreme Programming (XP). Como se menciona en [**36**]: *"Extreme Programming es una disciplina de desarrollo de software basada en valores de simplicidad, comunicación, retroalimentación y coraje. Funciona al unir a todo el equipo ante prácticas simples, con suficiente retroalimentación para permitirle ver dónde se encuentra y, así, adecuar las prácticas a su situación particular."* Las prácticas de XP se pueden apreciar en la Figura 2.1.

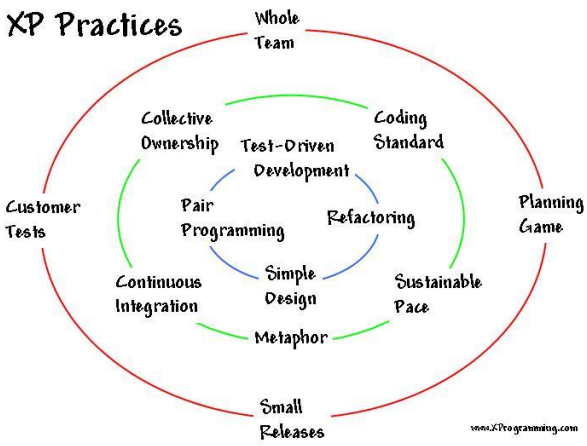

Figura 2.1 - Prácticas de Extreme Programming

Las prácticas que posee esta metodología y que la hacen ideal para un proyecto como el que se describe son:

- **Desarrollo guiado por pruebas.** Como se mencionó anteriormente, uno de los valores centrales de XP es la retroalimentación y no existe mejor manera de obtenerlo que mediante el desarrollo de pruebas. Como se menciona en [**37**]:

  "Desarrollo guiado por pruebas significa que escribes un test automático y, a continuación, escribes el código suficiente para pasar dicho test y después se 'refactoriza' el código, principalmente para mejorar la legibilidad y eliminar duplicaciones. Aclarar y repetir."

  Asimismo, como se enfatiza en [**36**], no es suficiente con escribir las pruebas, sino que también estas deben de ser ejecutadas **en su totalidad** cada vez que se agrega nuevo código a la implementación. De esta manera, el programador puede obtener una retroalimentación inmediata de cómo se está llevando a cabo el desarrollo.
  La ventaja de que se realicen pruebas constantemente es que los errores pueden ser corregidos de forma rápida y, así, incrementar la efectividad del proceso de programación.

- **Diseño incremental.** La metodología emplea un proceso denominado **refactorización** (*refactoring*, en inglés) que permite la mejora continua del diseño.

Esto significa, como se indica en [**37**], mantener el diseño simple desde el principio y mejorarlo continuamente, en lugar de conseguir que todo funcione desde el inicio y entonces congelarlo. Este proceso de diseño incremental es una consecuencia directa del desarrollo guiado por pruebas, dado que asegura que a medida que el diseño evoluciona, todo sigue funcionando correctamente.

Esta característica ofrece la ventaja de que es posible "ponerse manos a la obra" casi desde un inicio, sin tener que pasar por procesos largos de diseño y análisis.

Las características de XP que serán empleadas en la implementación del sistema que se plantea y cómo éstas serán adecuadas para el proyecto, se describen a continuación:

- **Desarrollo guiado por pruebas (TDD en inglés).** Como se ya explicó previamente, ésta es una de las características principales por la que esta metodología fue seleccionada. Para poder lograr que este proceso se lleve a cabo de forma casi automática, se emplearán utilidades (provistas por el lenguaje de programación seleccionado) para la implementación de *unidades de pruebas* dentro del código de la implementación.

- **Diseño simple – Diseño incremental.** Se realizará un diseño inicial de las estructuras de datos a emplear, la arquitectura de software y de información a emplear, el cual se presenta en las secciones siguientes de este mismo capítulo. Estos diseños serán mejorados continuamente a lo largo de proceso de implementación.

- **Integración continua – Propiedad colectiva del código.** Dado que el equipo de trabajo se encuentra formado por solo un integrante, todo el código fuente se encontrará centralizado.

- **Estandarización del código.** Se confecciona un documento de estándar de programación, el cual será incluido como un anexo al presente documento.

Finalmente, cabe mencionar que existe vasta evidencia de que XP se complementa bastante bien con SCRUM, que es la metodología seleccionada para la gestión del

proyecto. Esto se da, principalmente, porque ambas son metodologías ágiles y, por tanto, se rigen bajo los mismos principios del *"Manifest for Agile Software Development"* [**38**]. Asimismo, poseen disciplinas que se solapan, como son la liberación constante de pequeños demos funcionales, integración de todo el equipo de trabajo y planeamiento por iteraciones.

2.2 Identificación de Requerimientos

Como cualquier sistema de información, el sistema de detección de plagio en documentos digitales que se plantea desarrollar debe cumplir con ciertos requerimientos para que pueda ser empleado de manera apropiada por el usuario, siguiendo las buenas prácticas internacionalmente aceptados dentro de la ingeniería de software.

El establecimiento de las principales características que el sistema a implementar debe poseer y los requerimientos que éstas cumplen ha sido posible gracias a la investigación realizada sobre diferentes sistemas similares que ya existen en el mercado, así como también al análisis de estudios como los que se plantean en [**39**] o [**40**], que identifican las principales funcionalidades que un sistema que persiga el objetivo que se plantea en el documento debe brindar.

2.2.1 Requerimientos Funcionales

Se listan los requerimientos que constituyen las principales características sobre funcionalidad que el sistema debe implementar:

- **Módulo de Registro**

1. El sistema debe permitir almacenar una serie de documentos, que servirán a manera de repositorio para la comparación con los documentos sospechosos, de manera individual y/o masiva.
2. El sistema debe permitir al usuario determinar el momento en el cuál desea registrar un documento, pudiendo este ser: al realizar la comparación o en cualquier momento.

3. El sistema debe permitir al usuario remover un documento en el que se han encontrado similitudes parciales o totales incluso luego de haber sido registrado.

- **Módulo de Comparación y Evaluación**

1. El sistema debe permitir detectar el nivel de similitud total o parcial que posean dos documentos al ser comparados.
2. El sistema debe ser capaz de detectar plagio por copia directa y sin modificación alguna.
3. El sistema debe permitir comparar un documento con otro seleccionado de manera individual, en busca de patrones comunes de texto.
4. El sistema debe permitir seleccionar contra qué grupo de documentos que se encuentran en el repositorio se debe comparar el documento sospechoso.
4. El sistema debe poder detectar tanto solapamiento de tipo total o parcial. Es decir, debe poder determinar si un documento se encuentra completamente contenido dentro de otro o si solo parte de éste ha sido plagiada.
5. El sistema debe permitir indicar el nivel de similitud total o parcial que posean los documentos comparados a través de un indicador numérico.

- **Módulo de Visualización**

1. El sistema debe permitir visualizar de forma gráfica el nivel de similitud (parcial o total, medido en cantidad de oraciones comunes) que posee el documento sospechoso contra los que se encuentran en el repositorio.
2. El sistema de permitir visualizar de forma gráfica el detalle de las oraciones comunes que posee el documento sospechoso con los que se encuentran en el repositorio.

2.2.2 Requerimientos No Funcionales

Se listan los requerimientos que corresponden a características no funcionales que el sistema debe implementar:

1. El sistema debe brindar una interfaz gráfica con la cual el usuario pueda interactuar.
2. El sistema debe desarrollarse empleando herramientas de libre disponibilidad y que no acarreen costos de licencias de algún tipo.

2.2.3 Restricciones o Limitantes Funcionales

Así como existen características que el sistema debe poseer para poder cumplir con el objetivo que se ha planteado, existen ciertas restricciones que se tomarán en consideración en su desarrollo:

1. El sistema permitirá la comparación sólo de documentos con extensión PDF.
2. Para la detección del nivel de similitud, no se tendrán en consideración las fórmulas o gráficos que el documento pueda presentar.

Estas restricciones han sido establecidas a manera de requerimientos que el sistema no implementará debido a la complejidad que el cumplimiento de las mismas conlleva, y que se encuentran fuera del alcance del proyecto. Sin embargo, no se descarta que se tomen en consideración para un trabajo futuro.

2.3 Análisis de la Solución

En la presente sección de realiza un análisis de la solución propuesta, indicando la viabilidad de la implementación de la misma, así como la descripción del flujo de actividades que se pueden realizar con el mismo, teniendo en cuenta los requisitos descritos en la sección anterior.

2.3.1 Análisis de la viabilidad del sistema

Para poder realizar el análisis de viabilidad del sistema que se plantea desarrollar, se seguirán las directrices establecidas por la metodología MÉTRICA V3, en su subproceso Estudio de Viabilidad del Sistema (EVS), dentro del proceso Desarrollo de Sistemas de Información. Cabe resaltar, que dada la naturaleza del proyecto que se plantea, no se desarrollarán todas la tareas ni se elaborarán todos los productos sugeridos por la metodología, sino solo aquellos que se consideren pertinentes.

El objetivo del EVS es el análisis de un conjunto concreto de necesidades para proponer una solución a corto plazo, que tenga en cuenta restricciones económicas, técnicas, legales y operativas [**41**]. Las actividades que engloba este proceso se recogen en la Figura 2.2, en la que se indican las actividades que pueden ejecutarse en paralelo y las que precisan para su realización resultados originados en actividades anteriores.

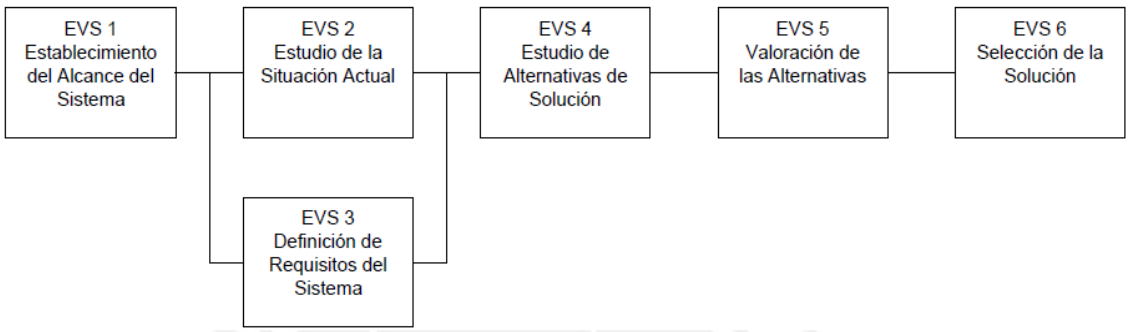

Figura 2.2 - Actividades del proceso EVS

- **Actividad EVS 1: Establecimiento del alcance del sistema**

  Se estudia el alcance de la necesidad que se desea satisfacer, realizando una descripción general de la misma. Considerando esto, se tiene:

  - Objetivo: implementar un sistema de información que permita la detección automática de plagio en documentos digitales.
  - Necesidades: detectar de forma automática señales de plagio en los documentos elaborados por los alumnos, contar con un sistema de fácil uso por los docentes para la detección de plagio en documentos, contar con un repositorio centralizado de documentos “originales”, contra los cuales comparar los documentos sospechosos de plagio.
  - Usuarios: El sistema será empleado por los docentes de una institución educativa (se toma como referencia a la PUCP).
  - Infraestructura: se cuenta con una infraestructura tecnológica óptima, que permite el almacenamiento y transmisión de grandes cantidades de datos a través de redes entre profesores y alumnos. Los docentes cuentan con equipos de cómputo con la capacidad necesaria para soportar software de última generación.

- Antecedentes: se cuenta con una política institucional que promueve el combatir el plagio entre los estudiantes, mas no así con una herramienta propia, adecuada a las necesidades institucionales.

- **Actividad EVS 2: Estudio de la situación actual**

  Se analizan los sistemas de información que se poseen actualmente, y que serán afectados por la herramienta que se plantea construir.

  Dado que la institución no posee un sistema de información con características similares al que se plantea implementar, ni algún sistema que se vea afectado por el mismo, esta actividad no aplica para el análisis que se desarrolla en este proyecto.

- **Actividad EVS 3: Definición de requisitos del sistema**

  Determinación de los requisitos generales y sus prioridades. Véase la sección 2.2, y el *Product Backlog* (Anexo 1).

- **Actividad EVS 4: Estudio de alternativas de solución**

  Se centra en proponer diversas alternativas que respondan satisfactoriamente a los requisitos planteados. Esta actividad ya fue realizada en el análisis del estado del arte, descrito en la sección 1.4 del capítulo 1.

- **Actividad EVS 5: Valoración de las alternativas**

  Se realiza una valoración de las alternativas descritas en la actividad anterior, considerando el impacto de las mismas, tanto desde el punto de vista tecnológico como operacional, así como los posibles beneficios esperados contrastados con los costes asociados. Se realiza también un análisis de riesgos, decidiendo cómo enfocar el plan de acción para minimizar los mismos.

  Como se ha podido observar, en la sección 1.4 se realiza un análisis tanto de herramientas ya construidas, como métodos para poder resolver el problema planteado. En primer lugar, se realizará la valoración de las alternativas de software ya implementado.

En la Figura 2.3 se presenta el cuadro correspondiente al análisis desarrollado, donde se puede ver que sólo se han considerado las herramientas que verifican documentos de texto libre y no las de código fuente. De la misma manera, se han excluido los buscadores de internet, dado que si bien proporcionan, de cierta manera, una forma de verificación, ésta es de forma indirecta y no automática. Adicionalmente, solo se han considerado aquellas herramientas que gozan de mayor popularidad en el mercado. Para poder valorar a las herramientas, se ha empleado una escala de 1 a 5, teniendo en cuenta que un mayor valor indica que la herramienta favorece la característica que está siendo considerada. Por ejemplo, un valor alto en "Costo de implantación", significa que se favorece el costo, ya sea porque la herramienta es libre o el precio de adquisición es bajo. Asimismo, un valor bajo en "Tiempo de implantación", significa que no se favorece dicha característica, quizás porque adquirir el software es complicado o porque su configuración y/o instrucción de uso toma bastante tiempo.

	Herramienta								
Características	**CopyCatch**	**CopyFind**	**EduTie**	**EVE2**	**Ferret**	**TurnItIn**	**MyDropBox (PlagiServe)**	**SHERLOCK**	**Desarrollo a medida**
Facilidad de uso	3	1	4	3	4	4	3	3	4
Costo de implantación	2	5	2	3	5	2	2	4	5
Tiempo de implantación	3	4	4	4	3	4	3	3	4
Personalización	3	1	2	2	3	3	3	3	5
Disponibilidad	4	5	4	4	4	4	4	4	1
Puntaje	**15**	**16**	**16**	**16**	**19**	**17**	**15**	**17**	**19**

Figura 2.3 - Valoración de sistemas de información

Como también se puede observar en la Figura 2.3, se ha agregado una herramienta que representa la posibilidad de elaborar un sistema de información a la medida de las necesidades planteadas.

En el caso del desarrollo a la medida, corresponde, igualmente, realizar una comparación de los métodos existentes y descritos también en el estado del arte, de tal manera que, de ser seleccionada dicha opción, se pueda también elegir qué método se empleará para la detección de plagio.

Siguiendo la misma mecánica de calificación, el resultado de la evaluación se muestra en la Figura 2.4.

	Métodos						
Características	**Redes bayesianas**	**Redes neuronales**	**Document fingerprinting**	**Chunking por oraciones**	**Chunking por palabras**	**Estrucutura característica**	**Arboles de sufijos**
Tiempo de implementación	2	2	4	4	4	3	4
Seguridad en la detección	3	3	4	4	3	4	4
Uso de memoria	3	3	4	4	4	2	3
Uso de espacio en disco	3	3	3	3	3	4	3
Puntaje	**11**	**11**	**15**	**15**	**14**	**13**	**14**

Figura 2.4 - Valoración de métodos de detección de plagio

Cabe mencionar que los valores de las calificaciones otorgadas a las herramientas y métodos, en las diferentes características a evaluar (Figura 2.3 y Figura 2.4), han sido determinados luego de examinar los documentos en los cuales son descritos, así como evaluación empírica (en el caso de las herramientas) realizada por el autor.

- **Actividad EVS 6: Selección de la solución**

  Se analizan las ventajas y desventajas de cada una de las soluciones propuestas con el fin de seleccionar la más adecuada.

  Considerando la valoración de los sistemas de información mostrada en la Figura 2.3, el emplear tanto el sistema Ferret como realizar una implementación a medida obtienen el mayor puntaje. Sin embargo, el nivel de personalización que se obtendría al realizar un sistema a medida es más alto que el de emplear simplemente Ferret. Además, aunque la disponibilidad del sistema a medida es

baja (dado que aún no ha sido implementado), no existe prisa por poseerlo y, por tanto, la diferencia en ese aspecto ya no es tan significativa. Teniendo todo eso en consideración, se selecciona a la implementación de un sistema a medida como la mejor opción a seguir, en este caso.

Ahora, corresponde seleccionar el método que se empleará para la herramienta a implementar. En este caso, observamos los resultados obtenidos en la Figura 2.4 y notamos que tanto *Document Fingerprinting* como *Chunking por Oraciones*, obtienen el más alto puntaje. Sin embargo, debido a que el primero es mayormente utilizado, y su implementación más sencilla, se decide optar por el mismo.

En conclusión, luego del análisis realizado, se justifica la implementación de un sistema de información para detección de plagio en documentos empleando el método de *Document Fingerprinting*.

2.3.2 Definición básica del sistema

Se realiza una descripción general del flujo de información que realiza el sistema que se plantea implementar, considerando todas las funcionalidades que debe poseer y que ya han sido descritas anteriormente. En la Figura 2.5 se presenta un diagrama de flujo de básico que muestra lo señalado.

Como se puede apreciar en el diagrama, la funcionalidad inicial es el registro de un documento en el repositorio. En caso que el usuario desee realizarlo, se lee el documento y se registra junto con los demás. Caso contrario, el usuario puede decidir si comparar un documento o no. Si es que así lo desease, debe seleccionar si leer un nuevo documento y registrarlo en el repositorio para luego compararlo, o directamente seleccionar un documento que ya ha sido registrado previamente en el repositorio. En cualquiera de estos dos casos, se prosigue con la comparación del mismo contra los documentos que se encuentran en el repositorio y, finalmente, se muestra el resultado correspondiente.

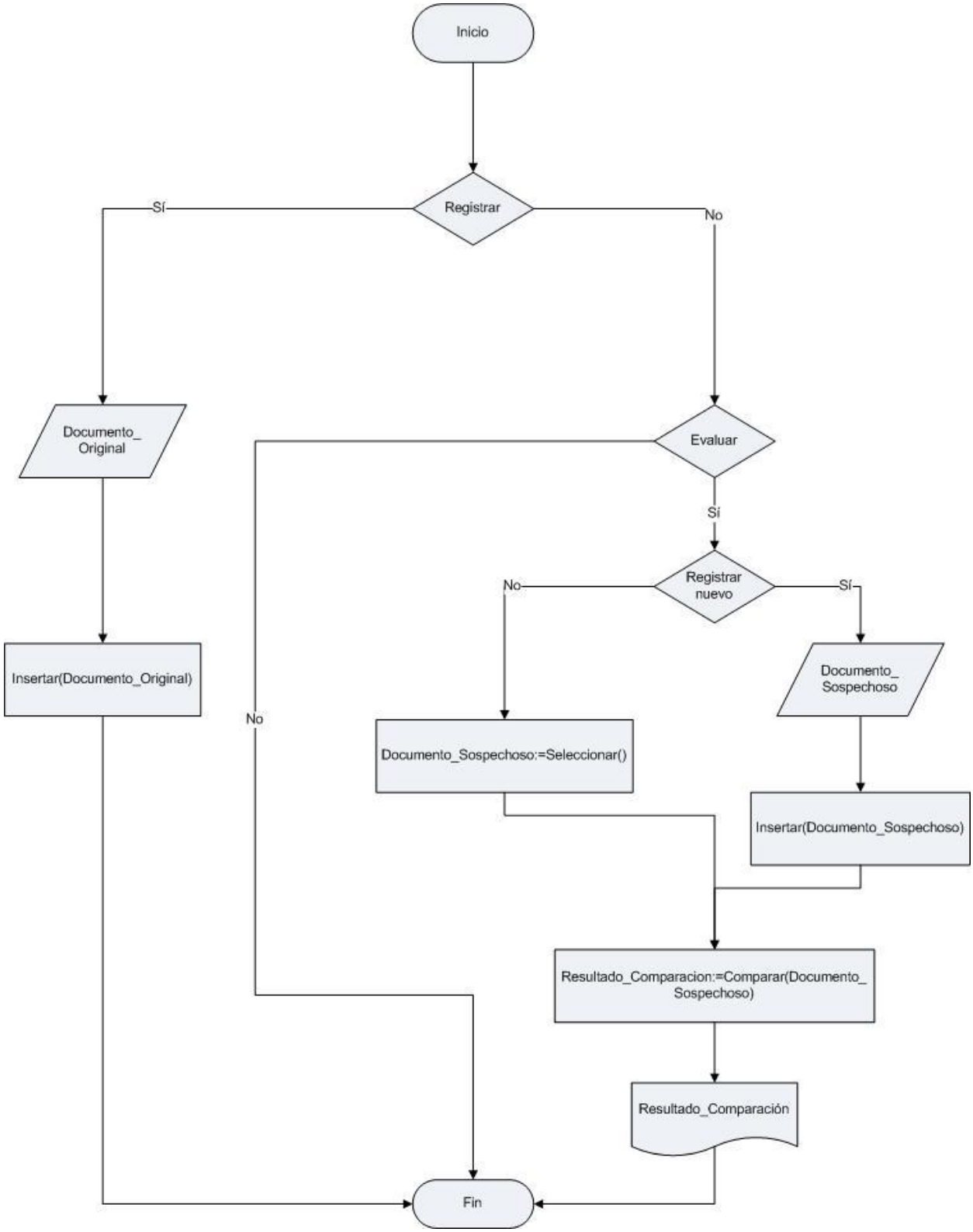

Figura 2.5 – Diagrama de Flujo

2.4 Arquitectura de la Solución

En la presente sección se presenta la forma en que los diferentes componentes del sistema interactúan entre ellos.

2.4.1 Clasificación del sistema según su arquitectura

De acuerdo a la clasificación que se presenta en [**39**] sobre herramientas para detección de plagio en documentos digitales, el sistema que se plantea en el proyecto pertenece a las siguientes clases:

- *Servidor.* Debido a que se posee una infraestructura de tipo Cliente-Servidor.
- *Local.* Debido a que el sistema no requiere de una estructura de red para su funcionamiento.
- *Basado en corpus.* Eso indica que se poseerá un repositorio en el que se almacenarán un conjunto de documentos.
- *Intracorpus.* Debido a que el sistema sólo permite la comparación de un documento sospechoso contra los que se encuentran almacenados en el repositorio.
- *Orientado a base de datos.* Debido a que el sistema poseerá una base de datos de tipo texto (repositorio).

2.4.2 Diseño de la Arquitectura

Tal y como se indicó en la sección anterior, el sistema empleará un arquitectura de tipo cliente – servidor. En la Figura 2.6 se presenta un diagrama de la arquitectura que se empleará para el sistema, la cual se basa en la del sistema MatchDetectReveal (MDR) [**30**]. Como se puede apreciar, la funcionalidad principal de evaluación será realizada por el módulo correspondiente que reside en el servidor dado que se requiere mayor capacidad de procesamiento de su parte. Por otro lado, el proceso de convertir el formato del archivo ingresado a uno que el sistema pueda procesar (Convertidor de formatos), el transformar el texto para que adquiera la estructura necesaria para poder ser almacenado en el repositorio (Parseador) y el módulo para visualizar los resultados de la comparación (Visualizador) radican en el Cliente.

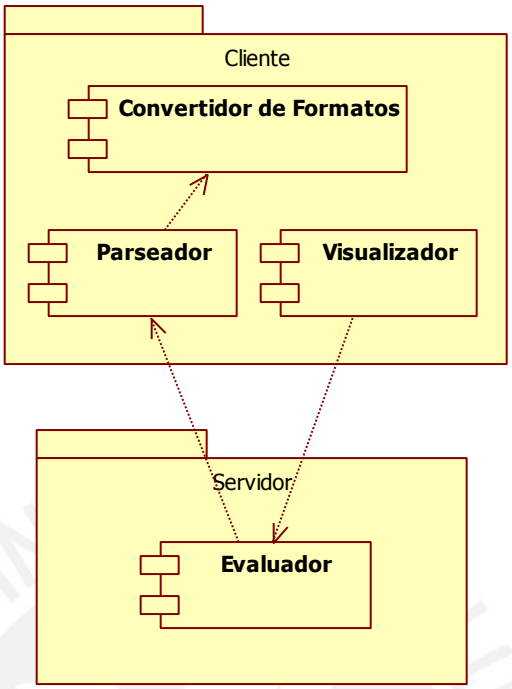

Figura 2.6 - Arquitectura de Software

2.5 Arquitectura de la Información

Esta sección está destinada a describir cómo es que se almacenarán los documentos dentro del repositorio.

El repositorio está conformado por dos secciones. La primera corresponde a la que almacena los documentos que el usuario ha registrado en el sistema, agrupándolos según algún criterio establecido por el usuario. Para ello, se empleará un esquema de *Categorías*, las cuales serán establecidas por el mismo usuario, con un límite de dos niveles de auto-referencia.

La segunda corresponde al almacenamiento de los documentos en el formato que requiere el algoritmo. Como se explicará más adelante, la estructura a almacenar es una tabla que contiene las particiones de cada documento, así como cierta información adicional y características de los mismos. El diseño de la base de datos a emplear, considerando las dos secciones, es el que se muestra en la Figura 2.7:

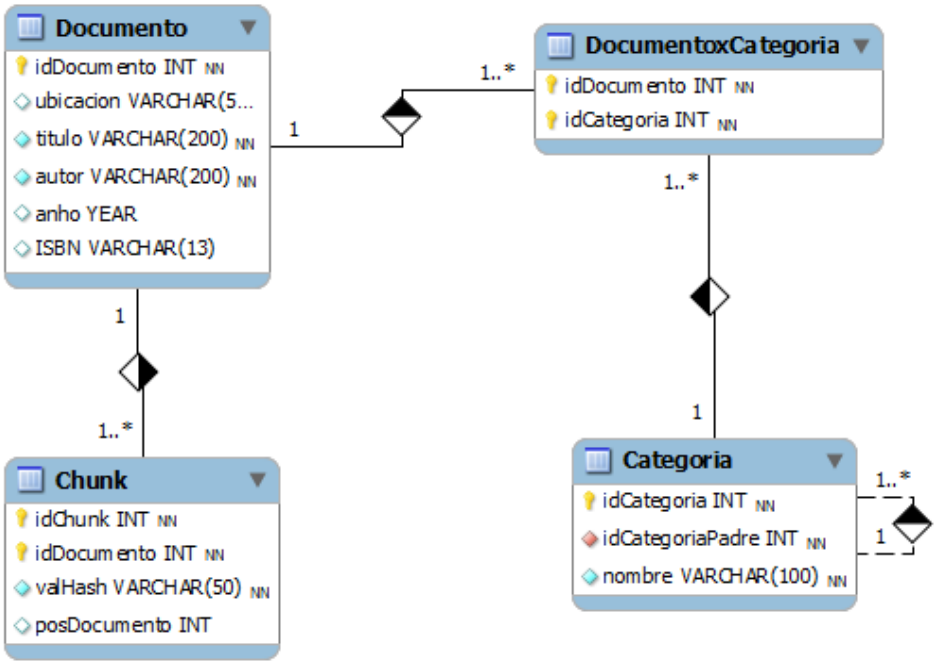

Figura 2.7 - Diseño de Base de Datos

Algunas características del diseño que vale la pena señalar son:

- En la tabla *Documento* se almacenarán los datos principales del documento que se requieren para poder realizar la comparación. El campo **ubicación** almacena la ruta en el disco en la que se ha almacenado el documento.

- La tabla *Chunk* contiene información importante sobre cada una de las particiones del documento. El campo **valHash** contiene el valor HASH de la partición en el *fingerprint* del documento. El campo **posDocumento** contiene un valor que indica la posición de la partición en el documento al que pertenece que se encuentra identificado por **idDocumento**.

- La tabla *Categoría* contiene información de la clasificación o el grupo al que pertenece un documento. Como se puede apreciar, una *Categoría* puede poseer *Sub-Categorías,* hasta un máximo de 2 niveles, lo cual será controlado por software. Por otro lado, el diseño soporta que un documento pueda estar clasificado en varias *Categorías*.

2.6 Diseño de la Interfaz de Usuario

Un aspecto fundamental en el desarrollo de aplicaciones es que la interfaz de usuario es, en realidad, el sistema para todo aquel que lo emplee. Una buena interfaz de usuario permite que quienes entienden el dominio del problema puedan trabajar con la aplicación de manera rápida y fácil. Para poder detectar el plagio, es necesario que el usuario evaluador tenga disponible todas las herramientas que le permitan determinar la originalidad del documento que está analizando, ya sean éstas reportes, visualizaciones a detalle de los documentos, entre otras.

En el diseño de la interfaz de usuario que se plantea en el presente documento, se tomaron en consideración las recomendaciones del análisis realizado en [**39**]. Este es un estudio en el que se analizan las diferentes interfaces de usuario que ofrecen la gran variedad de sistemas de detección de plagio que se encuentran disponibles en el mercado. Las características a tener en consideración y que se incluyen en el sistema a desarrollar son analizados en las secciones siguientes.

2.6.1 Interfaz Textual

Las características visuales y de distribución que no incluyen gráficos son:

- *Distribución en Fases.* Esta característica indica que el proceso que se debe realizar para poder emplear la herramienta se presenta en una distribución de pasos, lo cual hace más fácil que el usuario pueda dar un seguimiento al proceso, así como tener un mayor control sobre él.

- *Código de Colores.* Una de las principales características que presenta el sistema es la comparación detallada entre el documento sospechoso con los demás textos almacenados. Para poder visualizar las diferentes secciones de los documentos que presentan similitud se deben de emplear una serie de colores que permita identificar qué sección del documento sospechoso presenta solapamiento con qué documento almacenado en el repositorio.

2.6.2 Interfaz Gráfica

Para la representación de resultados se tendrá en consideración las siguientes características:

- *Análisis de un documento sospechoso contra varios almacenados en repositorio.* Se propone una adaptación de la gráfica empleada por el sistema PRAISE [**40**]. Consiste en un diagrama de barras agrupadas por pares, cada uno de los cuales posee la cantidad total de oraciones del documento sospechoso, contra la cantidad de oraciones que son similares con otro documento del repositorio, contra el cual ha sido comparado.

- *Análisis del documento en detalle.* Se plantea mostrar el contenido del documento sospechoso y una lista de los documentos del repositorio con el que comparta oraciones similares. Considerando la codificación de colores indicada en la sección anterior, se pretende mostrar más a detalle cuáles de las oraciones del documento sospechoso corresponden con qué documentos del repositorio.

"Lo importante no es hacer cosas nuevas sino hacerlas como si nunca nadie las hubiera hecho antes"

Goethe

3 Aporte

3.1 Propuesta de solución

Como se ha indicado anteriormente, el método de *Document Fingerprinting* es el más empleado en los sistemas de detección de plagio actuales, principalmente, porque posee la ventaja que no requiere almacenar todas las oraciones del documento que se desea evaluar, sino, únicamente, sus valores numéricos, luego de aplicar una función hash, logrando que la comparación sea más rápida y reduciendo el espacio de almacenamiento.

Para el proceso de obtención de la *fingerprint* del documento de empleará el algoritmo *Winnowing* que se describe en [**22**]. En la presente sección de describe el método que se va a emplear, así como la forma en que se plantea mejorarlo.

3.1.1 El algoritmo *Winnowing*

El algoritmo *Winnowing* es una instancia específica de un algoritmo para extracción de huellas digitales de un documento (*document fingerprinting*). En el capítulo 1, sección

1.4.2, se describió el proceso que sigue el método de *document fingerprintig*; por tanto, aquí solo se indicarán las particulares que aporta al mismo el algoritmo *Winnowing*.

- **Cálculo de particiones.** Se empieza por calcular un valor hash para la primera palabra en el documento. Si el módulo del valor hash con `k` es igual a cero (para un `k` seleccionado), la primera partición es simplemente la primera palabra; si no, considerar la segunda palabra. Si el módulo del valor hash con `k` es cero, las dos primeras palabras son consideradas la partición. En caso contrario, se debe continuar considerando las siguientes palabras hasta que se encuentre una palabra cuyo módulo de su valor hash con `k` sea cero, y la secuencia de palabras desde el último final de una partición hasta esta palabra constituye la nueva partición. Cada una de estas particiones recibe el nombre de `k-grama`.

- **Selección de particiones.** Se define una *ventana* de tamaño `w`, la cual es un conjunto de `w` valores hash consecutivos de `k-gramas` en un documento (`w` es un parámetro seleccionado). Se debe seleccionar un valor hash de cada ventana para que forme parte de la *fingerprint* del documento. En cada ventana, se selecciona el mínimo valor hash. En caso exista más de un hash con el valor mínimo, se selecciona el que se ubica más a la derecha.

El conjunto de *fingerprints* asociado a un documento es reducido de manera significativa al considerar al documento como un conjunto máximo de `n - w - 1` ventanas que se solapan de longitud `w`, y también si sólo se mantienen suficientes hashes para asegurar que cada ventana contiene al menos un hash. Si `w` es suficientemente grande comparado con `k`, el número de *fingerprints* asociado al documento de longitud `n` es significativamente más pequeño que `n - k + 1`. Se demuestra en [**22**] que incluso este conjunto reducido de *fingerprints* puede encontrar copias de pasajes de texto bastante significativas, dependiendo de las configuraciones de los parámetros `k` y `w`: cualquier coincidencia al menos tan grande como el umbral de garantía `t = w + k - 1` será detectada.

En la Figura 3.1 se presenta un ejemplo del algoritmo *Winnowing*.

A do run run run, a do run run run
(a)Un texto cualquiera

adorunrunrunadorunrunrun
(b)El mismo texto con características innecesarias removidas

adoru dorun orunr runru unrun nrunr runru unrun nruna runad unado nador adoru dorun
(c)La secuencia de 5-gramas derivadas del texto

77 4 42 17 98 50 17 98 8 88 67 39 77 74 42 17 98
(d)Una secuencia hipotética de valores hash para los 5-gramas.

(77, 74, 42, **17**) (74, 42, 17, 98) (42, 17, 98, 50) (17, 98, 50, **17**)
(98, 50, 17, 98) (50, 17, 98, **8**) (17, 98, 8, 88) (98, 8, 88, 67)
(8, 88, 67, 39) (88, 67, **39**, 77) (67, 39, 77, 74) (39, 77, 74, 42)
(77, 74, 42, **17**) (74, 42, 17, 98)
(e) Ventanas de valores hash con w=4.

17 17 8 39 17
(f) *Fingerprints* seleccionadas por el método.

Figura 3.1 - Ejemplo algoritmo *Winnowing* [22]

En la Figura 3.1(e) los valores hash en negrita son los seleccionados por el algoritmo. Se debe resaltar que los mínimos son sólo seleccionados en la primera ventana que aparecen, reduciendo de esta manera el número final de *fingerprints* considerado para el documento.

3.2 Estructura de la solución

En [**42**] se indican cuáles son las funciones y procedimientos principales con los que debe contar un sistema que desea emplear una técnica de particionamiento y tablas hash para el reconocimiento de plagio. Si bien indica las características de los procedimientos, no especifica cada uno de ellos, dejando esa tarea al implementador. En las siguientes secciones se describen los procedimientos y funciones a implementar

y que están basados en lo descrito en el documento que se menciona pero particularizados para el objetivo que se persigue en este documento.

3.2.1 Operación de Carga Inicial de Documentos

El objetivo de este procedimiento es el de realizar la carga inicial de documentos en el repositorio, que incluye la creación de la taba hash correspondiente (tabla de particiones).

Sean `R` un conjunto de documentos a registrar, `H` la tabla hash a crear y `r` un documento particular del conjunto `R`, el algoritmo que implementa este procedimiento se muestra en la Figura 3.2

```
CARGA_INICIAL (R, H)
      CREAR_TABLA(H)
      Para cada r en R INSERTAR (r, H)
```

Figura 3.2 - Seudocódigo del procedimiento `CARGA_INICIAL`

Luego de realizar este procedimiento, se tendrán los documentos de `R` registrados en tabla hash `H`, y por ende formarán parte del repositorio.

Este proceso de carga inicial de documentos se realizará sólo en el caso que el usuario así lo desee, de tal manera que se forme un repositorio inicial para la aplicación.

3.2.2 Operación de Pre-procesamiento

Antes de poder ingresar en el repositorio, el documento que se desea registrar debe pasar por un proceso de pre-procesamiento. El algoritmo para este proceso se presenta en la Figura 3.3:

```
PRE_PROCESAMIENTO(r)
      lstOraciones <- SEPARAR_ORACIONES(r)
      lstOraciones <- LIMPIAR_TEXTO(lstOraciones)
      lstOraciones <- ELIMNAR_SECCIONES (lstOraciones)
      retornar lstOraciones
```

Figura 3.3 - Seudocódigo del procedimiento `PRE_PROCESAMIENTO`

Para el pre-procesamiento de los documentos, se empleará lo que se indica en [**43**], cuyos pasos son:

- **Separación de Oraciones**

Lo primero que se requiere es dividir el texto en oraciones. Los puntos y cambios de línea son posibles marcadores de bordes entre oraciones; sin embargo se debe tener mucho cuidado con los mismos por lo que se establecen las siguientes reglas:

1. Un cambio de línea seguido de una letra mayúscula es considerado el final de una oración.
2. Un cambio de línea que no es seguido de una letra mayúscula es ignorado.
3. Un punto al final de abreviaciones comunes como “Prof.”, “Doc.”, “Fig.” es ignorado.
4. Un punto seguido de un carácter diferente de una letra (por ejemplo un espacio en blanco) es considerado el final de una oración si la regla previa no aplica.
5. Un punto seguido directamente por una letra es ignorado.

Existen otras consideraciones que se tienen que tener en cuenta para mejorar la calidad de las oraciones que se obtengan luego del pre-procesamiento indicado. Esto es, por ejemplo, un guión seguido de un cambio de línea se reconoce como la división de una palabra, entonces deben de unirse los dos fragmentos de la palabra en uno solo. Sin embargo, para la implementación del sistema que se plantea sólo se consideran las indicadas en la lista anterior.

- **Limpieza del texto**

Luego de obtener las oraciones, se procede a simplificar el texto en cada oración como sigue:

1. Secuencias de caracteres de espacios en blanco son reducidas a un solo espacio.
2. Todos los caracteres a excepción de letras y espacios son removidos.
3. Palabras que consisten de una letra son removidas.
4. Todas las letras con convertidas a minúscula o mayúscula.

- **Eliminación de secciones innecesarias**

El paso final del pre-procesamiento es el retiro de aquellas oraciones que no deban ser empleadas en la comparación, como son las correspondientes a secciones no relevantes para la detección de plagio. Esto es, se deben de retirar las oraciones correspondientes a la **Tabla de Contenidos** y **Bibliografía**. El posible solapamiento de textos en estas secciones del documento no es de interés para la detección de similitudes.

Para poder lograr este objetivo, se debe ubicar el inicio de estas secciones en la representación de texto de cada documento buscando palabras o frases como: "Bibliografía", "Referencias", "Tabla de Contenidos" e "Índice".

3.2.3 Operación de registro de nuevo documento

En este caso se considera el procedimiento para añadir documentos a H de manera particular, es decir registrar nuevos documentos. Este se muestra en la Figura 3.4.

```
INSERTAR (r, H)
     C <- OBTENER_CHUNKS (r)
     Para cada <t, l> en C
          h <- HASH (t)
          INSERTAR_CHUNK (<h, r, l>, H)
```

Figura 3.4 - Seudocódigo del procedimiento `INSERTAR`

Para poder insertar un documento en la tabla hash, el procedimiento `INSERTAR` emplea una función `OBTENER_CHUNKS (r)` para obtener las particiones de un documento `r`. Esta función retorna un conjunto de tuplas; cada tupla `<t, l>` representa una partición en `r`, donde `t` es el texto en la partición, y `l` es la ubicación de la partición. Una entrada es almacenada en la tabla hash para cada partición `<t, l>` en el documento.

Las funciones para obtener el valor hash y las particiones, serán descritas en secciones posteriores.

3.2.4 Operación de Selección de Particiones (*Culling*)

De un documento se pueden obtener una gran cantidad de particiones; sin embargo no todas son significativamente útiles para la comparación. Asimismo, no es conveniente,

por cuestiones de espacio de almacenamiento, el tener que guardar todas las particiones de un archivo. Por tal motivo, se empleará el método de selección de particiones descrito en [**44**].

Una partición corta no es muy representativa de un texto. El hecho de que dos documentos compartan una partición corta no indica necesariamente que se haya cometido delito de plagio. Por otro lado, particiones muy largas son bastante representativas pero, a menos que un plagiario sea lo suficientemente ingenuo, es bastante improbable que una copia contenga una sección extensa del texto original. Por ello, se descartan las particiones más largas y más cortas.

Sea n el número de particiones, m la media del tamaño de una partición (medido en unidades), s la desviación estándar del tamaño de una partición, b una constante y L la longitud de una partición cualquiera, se debe mantener aquellas particiones tales que: $|L - m| \leq bs$. Incrementar b, si es necesario, hasta que al menos $\sqrt{n}$ particiones sean seleccionadas. Se empieza con $b = 0.1$.

3.2.5 Operación de Compresión de Particiones

Como se indica en [**44**], en lugar de almacenar las particiones, lo más conveniente es reducirlas aplicando una herramienta de compresión. Eso se refleja en el empleo de la función `HASH`, que permite obtener el valor hash correspondiente para una partición dada. En este caso se propone el uso del algoritmo MD5, el cual convierte un flujo arbitrario de bytes a un número de 128 bits. Se calibra el mismo para establecer la cantidad de dígitos hexadecimales de la compresión MD5, por ejemplo a 10.

3.2.6 Operación de Evaluación

En este caso se considera el procedimiento que se encargará de evaluar si un documento `d` viola a algún documento de `H` de acuerdo a la prueba a la que se le someta, en nuestro caso la de plagio. El algoritmo correspondiente se muestra en la Figura 3.5.

```
EVALUAR (d, H)
      C <- OBTENER_CHUNKS (d)
      TAMANHO <- | C |
      COINCIDENCIAS <- { }
```

```
Para cada <t, ld> en C
      h <- HASH (t)
      SS <- BUSCAR (h, H)
      //Retorna todos los <r, lr> que coinciden con h
            Para cada <r, lr> en SS
            COINCIDENCIAS + = < |t|, ld, r, lr>
Retornar DECIDIR (COINCIDENCIAS, TAMANHO)
```

Figura 3.5 - Seudocódigo del procedimiento `EVALUAR`

El procedimiento `EVALUAR (d, H)` evalúa un documento `d` en busca de violaciones. El procedimiento emplea la función `OBTENER_CHUNKS` para partir `d`. Luego de realizar el particionamiento, el procedimiento `BUSCAR` busca las particiones en la tabla hash `H` produciendo un conjunto de tuplas `SS`. Cada una de estas tuplas es luego insertada en `COINCIDENCIAS` con el formato adecuado. Cada `<s, ld, r, lr>` en `COINCIDENCIAS` representa una coincidencia: una partición de tamaño `s`, en la ubicación `ld` en el documento `d` coincide (posee la misma llave hash) que una partición en la ubicación `lr` en un documento registrado `r`. El conjunto `COINCIDENCIAS` es luego empleado por la función `DECIDIR (COINCIDENCIAS, TAMANHO)` (donde `TAMANHO` es el número de particiones en `d`) que retorna el resultado del cálculo de la función de evaluación de plagio que se haya establecido.

Una función simple de decisión podría ser *coincidencias* (con un parámetro *C*) que simplemente pruebe si el número de coincidencias entre el documento prueba y el documento registrado sobrepasa un cierto valor *C*. De sobrepasar este valor, se puede determinar que el documento ha sido, efectivamente, plagiado.

3.2.7 Operación de obtención de *chunks*

La operación que falta describir es la que viene representada en los seudocódigos anteriores como `OBTENER_CHUNKS`. Para poder desarrollar este proceso, se empleará el algoritmo *Winnowing* descrito previamente. En la Figura 3.6 se presenta el algoritmo que corresponde a este procedimiento.

```
OBTENER_CHUNKS (d)
        //Se realiza el pre-procesamiento del texto
        //para obtener las oraciones
        lstOraciones <- PRE_PROCESAMIENTO(d)
        lstParticiones <- {}
        Para cada oración en lstOraciones hacer
            //Se obtienen las particiones (k-gramas)
            lstParticiones += OBTENER_PARTICIONES(oracion)
        //Se inicia el algoritmo Winnowing
        lstChunks <- {}
        limIzq <- 0, limDer <- limIzq + W - 1
        indMin <- 0, indParticion <- indLimIzq + 1
        nuevoMin <- TRUE
        Mientras indParticion < LENGTH(lstParticiones) hacer
        INICIO
            Si (indParticion > indLimDer) entonces
            INICIO
                Si (nuevoMin) entonces
                INICIO
                    lstChunks += lstParticiones [indMin]
                    nuevoMin <- FALSE
                FIN
                indLimIzq <- indLimIzq + 1
                indLimDer <- indLimIzq + W - 1
                indParticion <- indLimIzq + 1
            FIN
            Si (indMin < indLimIzq) entonces
                indMin <- indLimIzq
            Caso Contrario
            INICIO
                Si ((indParticion <> indMin ) AND
                    (lstParticiones[indParticion]<
                     lstParticiones[indMin])) entonces
                INICIO
```

```
                    indMin <- indParticion
                    nuevoMinimo <- TRUE
                FIN
                indParticion<-indParticion + 1
            FIN
        FIN
        Retornar lstChunks
```

Figura 3.6 - Seudocódigo del procedimiento `OBTENER_CHUNKS`

Como se puede observar, en `OBTENER_CHUNKS` se emplea una función `OBTENER_PARTICIONES`, cuyo objetivo es obtener todas las particiones de una oracione, empleando k-gramas. El seudocódigo de esta función de puede observar en la Figura 3.7.

```
OBTENER_PARTICIONES(o)
      lstPalabras <- OBTENER_PALABRAS (o)
      lstParticiones <- {}
      inicio <- 0, fin <- 0, numPalabra <- 1
       Mientras (inicio <= (LENGTH(lstPalabras - K) hacer
       INICIO
            cadParticion <- “”
            fin <- inicio + K – 1
            Para i <- inicio hasta fin hacer
                 cadParticion += lstPalabras[i]
            //Se obtiene el valor HASH con el algoritmo MD5
            particion.valHash <- OBTENER_HASH(cadParticion)
            particion.numPalabra <- numPalabra
            numPalabra <- numPalabra + 1
            lstParticiones += particion
            inicio <- inicio + 1
       FIN
       Retornar lstParticiones
```

Figura 3.7 - Seudocódigo del procedimiento `OBTENER_PARTICIONES`

3.3 Evaluación de la precisión de la solución

Para poder evaluar la precisión en el resultado que proporciona la solución, es necesario realizar una serie de experimentos empleando un conjunto adecuado de documentos prueba. El objetivo de estos experimentos es entender cuántas particiones similares se puede esperar que posean los documentos al ser comparados con los que se encuentran en el repositorio y, consecuentemente, determinar qué tan bien funciona el algoritmo de detección. El proceso que se emplea en los experimentos es similar al que se expone en [**42**].

Se emplea un grupo de 150 documentos. Aproximadamente, la mitad de estos se agrupa por temas (etiquetados como A, B,…, I). Los documentos dentro de cada grupo se encuentran estrechamente relacionados; mientras que los que se encuentran en diferentes grupos no poseen relación significativa. Los documentos que pertenecen a la mitad restante no pertenecen a un grupo particular y no se encuentran relacionados con los documentos en la colección.
Todos estos documentos fueron registrados en el sistema, y luego cada uno fue evaluado contra todos lo demás. El objetivo es determinar si el sistema puede determinar cuáles son los documentos que se encuentran estrechamente relacionados.

En la Figura 3.8, se muestran los resultados de la experimentación realizada. En lugar de mostrar el número exacto de particiones similares, se ha calculado el porcentaje de particiones compartidas en cada caso. Esto provee una mayor información respecto a la cercanía entre los documentos.

El primer resultado de la Figura 3.8, provee el porcentaje de similitud de cada documento contra sí mismo. Esto es, para cada documento en el grupo, se calcula:

```
100 * NUM_PART_SIMILAR(prueba, prueba)/ NUM_PART_TOTAL(prueba)
```

Se promedian los valores y se muestra en la fila para ese grupo. El que todos los valores en la primera columna sean 100%, permite confirmar que el sistema funciona correctamente, dado que es de esperarse que un documento sea totalmente similar consigo mismo.

GRUPO	SIMILITUD CONTRA SÍ MISMO	SIMILITUD DOCUMENTOS RELACIONADOS (AFINIDAD)	SIMILITUD DOCUMENTOS NO RELACIONADOS (RUIDO)
A	100%	61.9%	0.8%
B	100%	40.5%	1.1%
C	100%	39.8%	1.4%
D	100%	32.8%	0.7%
E	100%	28.9%	1.5%
F	100%	56.0%	1.8%
G	100%	49.0%	0.9%
H	100%	50.4%	1.9%
I	100%	61.7%	1.6%
Promedio Total	***100%***	***46.8%***	***1.3%***

Figura 3.8 - Resultados experimentación de calidad del algoritmo

Para la segunda columna, el proceso a seguir es el siguiente: para cada documento en el grupo se calcula:

```
100*NUM_PART_SIMILAR(prueba, grupo)/NUM_PART_TOTAL(prueba)
```

Luego de realizar este cálculo para todos los otros documentos en el grupo, se promedian los resultados. Estos valores son denominados **afinidad**, dado que representan cuan relacionados se encuentran los documentos.

Para la tercera columna, el proceso es similar al anterior, pero en lugar de comparar cada documento contra los de su mismo grupo, se realiza contra los otros grupos. Estos resultados se denominan **ruido**, dado que son detecciones no deseadas.

Idealmente, se desea obtener un valor alto de afinidad y uno bajo de ruido, dado que esto permite determinar cuán preciso es el algoritmo al momento de realizar la detección. De acuerdo con la experimentación presentada, los documentos relacionados poseen, en promedio, un 46.8% de particiones similares, mientras que los que no se encuentran relacionados poseen 1.3%. Esto quiere decir que si dos documentos poseen alrededor de 46.8% de oraciones similares, de acuerdo con la

solución que se plantea, entonces es bastante seguro afirmar que se encuentran relacionados y, por tanto, se ha incurrido en plagio. Por otro lado, si solamente poseen un 1.3% de particiones similares, no se puede asegurar que los documentos se encuentren relacionados.

Con estos resultados, se pueden establecer rangos de calidad de la solución planteada de acuerdo con la respuesta que se obtiene (porcentaje de oraciones similares). Estos se muestran en la Figura 3.9.

Rango	Significado
Porcentaje < 1.3 %	No se incurre en plagio
1.3% < Porcentaje < 46.8%	Posiblemente se incurre en plagio
Porcentaje > 46.8%	Seguramente se incurre en plagio

Figura 3.9 - Rangos de calidad de la respuesta de la solución

En el caso del segundo rango (“Posiblemente se incurre en plagio”), es necesario que intervenga el usuario para que verifique, directamente, las particiones que se señalan como comunes y así determine fehacientemente si se ha incurrido en plagio o no.

“No basta saber, se debe también aplicar. No es suficiente querer, se debe también hacer.”

Goethe

4 Implementación del sistema de información

En el presente capítulo se describen las diferentes consideraciones que se han tomado en cuenta para la implementación del algoritmo que fue planteado en la sección anterior. Asimismo, se indican las decisiones tomadas en cuanto a la implementación de las demás características del sistema, como es, principalmente, la interfaz con la que contará.

4.1 Composición del Sistema

En esta sección se presentan los módulos que componen el sistema de detección de plagio que se detalla en el documento. En la Figura 4.1, se muestra una visión general de los módulos del sistema y la forma cómo interactúan.

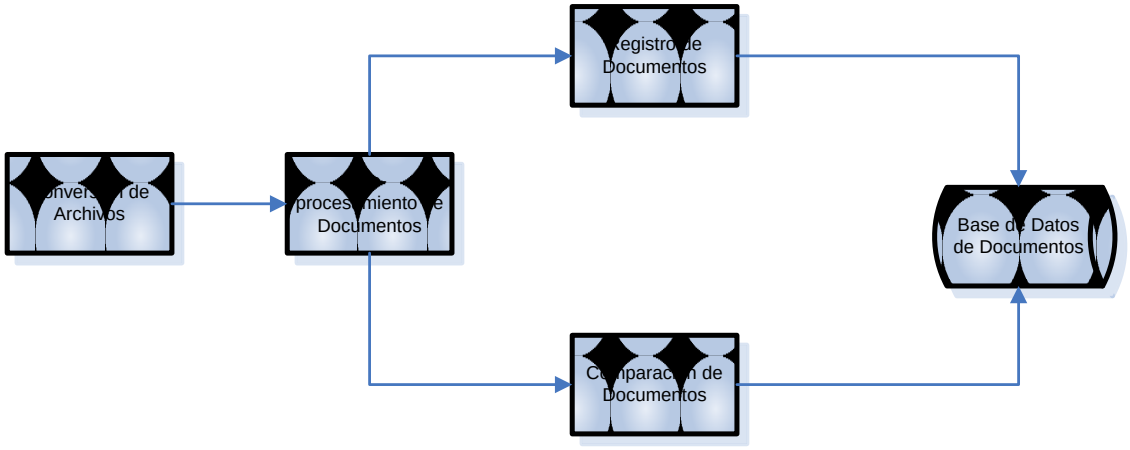

Figura 4.1 - Composición del Sistema

4.1.1 Módulo de conversión de archivos

Este módulo es el encargado de transformar los archivos de los diferentes formatos a archivos de texto planos. De forma particular, se encarga de leer los documentos en formato .PDF y obtener su contenido en un formato de caracteres ASCII. Para poder realizar la implementación de este módulo se emplea la siguiente librería:

PDFBox [**45**] **.** Es una librería JAVA PDF open source que permite trabajar con documentos en formato PDF. Esta librería permite la creación de nuevos documentos PDF, manipulación de documentos existentes y la habilidad de extraer el contenido de los mismos. Es esta última característica la que es de particular interés para la implementación que se plantea.

4.1.2 Módulo de pre-procesamiento de documentos

Este módulo es el encargado de preparar los documentos en texto plano para que puedan ser empleados por el algoritmo de particionamiento. Para ello realiza todas las operaciones descritas en la sección 3.2.2 del capítulo 3. Para poder realizar la extracción de oraciones del documento, una vez que ya ha sido transformado a texto plano, se emplea una librería especializada llamada LingPipe [**46**]. En realidad, LingPipe es un conjunto de librerías que proveen de diferentes funcionalidades para el análisis lingüístico del lenguaje humano. Dentro de las funcionalidades que ofrece tenemos: clasificación de documentos por temas, etiquetado morfológico, *clustering* de documentos, corrección ortográfica, **detección de oraciones**, entre otros.

4.1.3 Módulo de registro de documentos

Este módulo es el encargado de realizar el registro de los documentos nuevos que se deseen ingresar al repositorio de archivos. Un usuario puede ingresar un documento al sistema e indicar que es un documento original. En este caso, el sistema simplemente llamará al módulo de registro de documentos.

4.1.4 Módulo de comparación de documentos

El módulo de comparación de documentos es el encargado de ejecutar el algoritmo que permite determinar si un documento prueba ingresado por el usuario posee algún tipo de solapamiento con los documentos ya registrados en el repositorio y, por tanto, se incurre en plagio. Para poder registrar un documento, el usuario puede acceder directamente al módulo de registro de documentos, o si no puede realizar esta acción al momento de realizar comparación. En este último caso, el módulo de comparación, simplemente, invocará al módulo de registro antes de realizar sus propias actividades.

4.1.5 Base de datos de documentos (Repositorio)

En la base de datos de documentos es en donde se almacenan todos los documentos que han sido registrados a través del sistema. Está diseñada siguiendo los lineamientos de la arquitectura de información indicados en el capítulo 2.

4.2 Framework de implementación

Para poder realizar la implementación del sistema de información se decidió emplear Eclipse Rich Client Platform (Eclipse RCP) como framework para la aplicación. Eclipse RCP es una poderosa plataforma para construir y desplegar diferentes tipos de aplicaciones de escritorio. Su fundamento básico es el uso de plug-ins, es decir, componentes que pueden ser reutilizados en diferentes aplicaciones, de acuerdo con las necesidades que se posea. Toda aplicación que se desarrolla empleando esta tecnología es, en realidad, un plug-in que puede ser reutilizado, incluso, en el mismo Eclipse IDE. En la terminología de Eclipse RCP, los principales componentes de una UI son las vistas y los editores. Los editores están concebidos para realizar las tareas principales de la aplicación, mientras que las vistas proveen información adicional y significativa. Otros componentes que provee la plataforma son: diálogos y wizards. Toda esta terminología será empleada para describir la interfaz de usuario en la sección siguiente.

4.3 Interfaz de Usuario

En esta sección se presentan los “pantallazos” de las diferentes vistas, diálogos y wizards de interfaz gráfica con las que interactuará el usuario en el uso del sistema a implementar.

4.3.1 Pantalla Principal

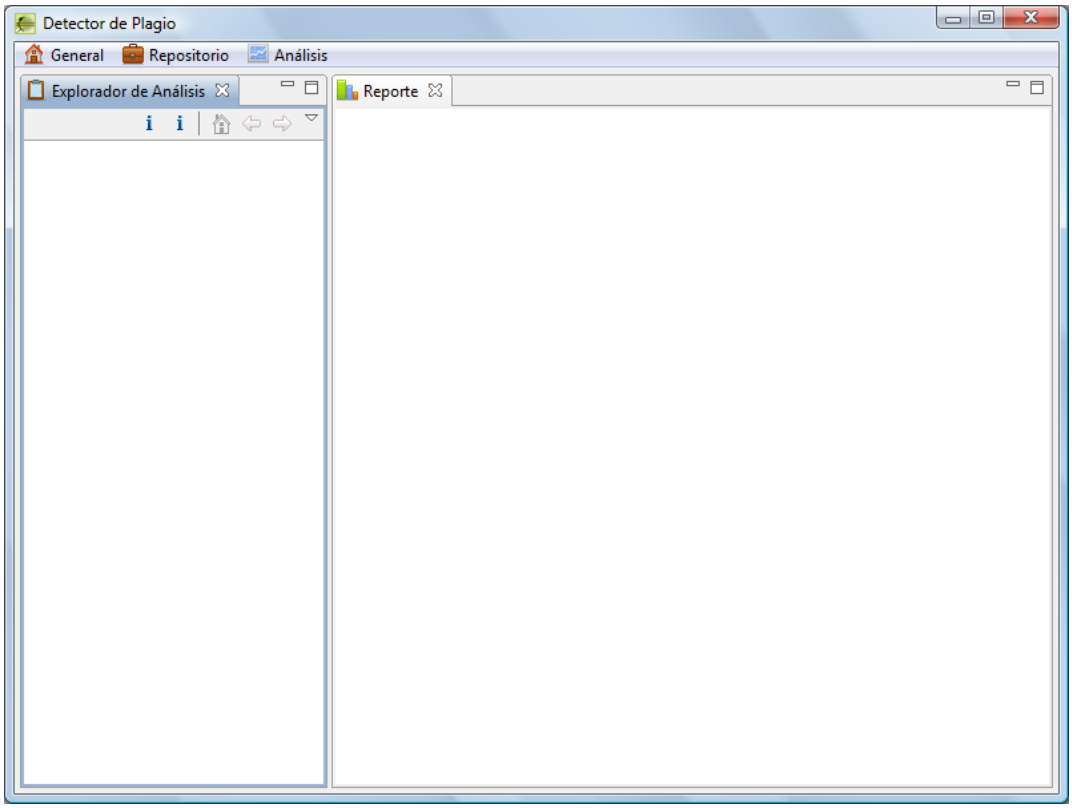

Figura 4.2 - Pantalla Principal del Sistema

La página principal del sistema (Figura 4.2) brinda los accesos necesarios para las principales funcionalidades de la aplicación y muestra la distribución de las vistas de la misma.

El menú General permite el acceso a opciones de uso común como el salir de la aplicación y las preferencias.

El menú Repositorio permite el acceso a las diferentes acciones que se pueden realizar con el mismo, como son: registro de categorías y registro de documentos.

El menú Análisis permite el acceso a las opciones relacionadas con el análisis o evaluación de documentos sospechosos.

En secciones posteriores se presenta una descripción detallada de las vistas que se pueden apreciar.

4.3.2 Pantalla de Preferencias

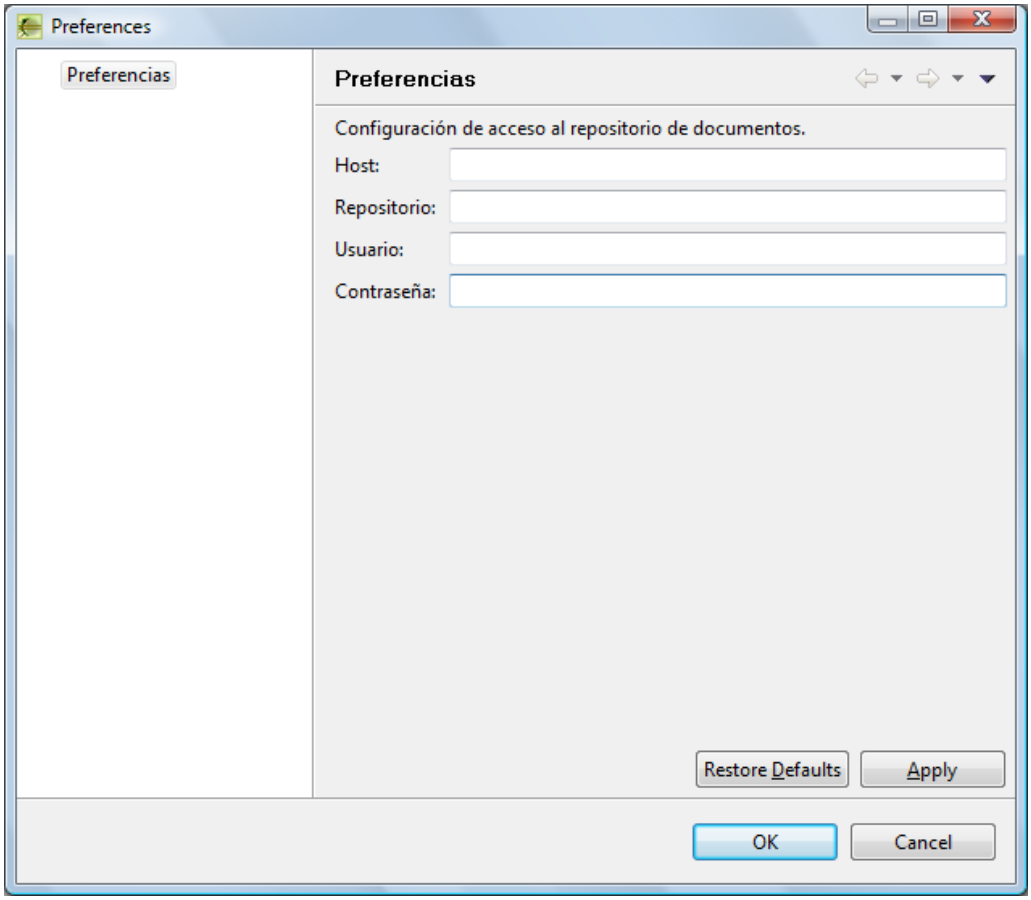

Figura 4.3 - Pantalla de Preferencias

La pantalla de preferencias permite establecer los valores para la conexión del sistema al repositorio de datos. Los campos disponibles son: host, repositorio (o base de datos), usuario y contraseña.

4.3.3 Pantalla de Registro de Categorías

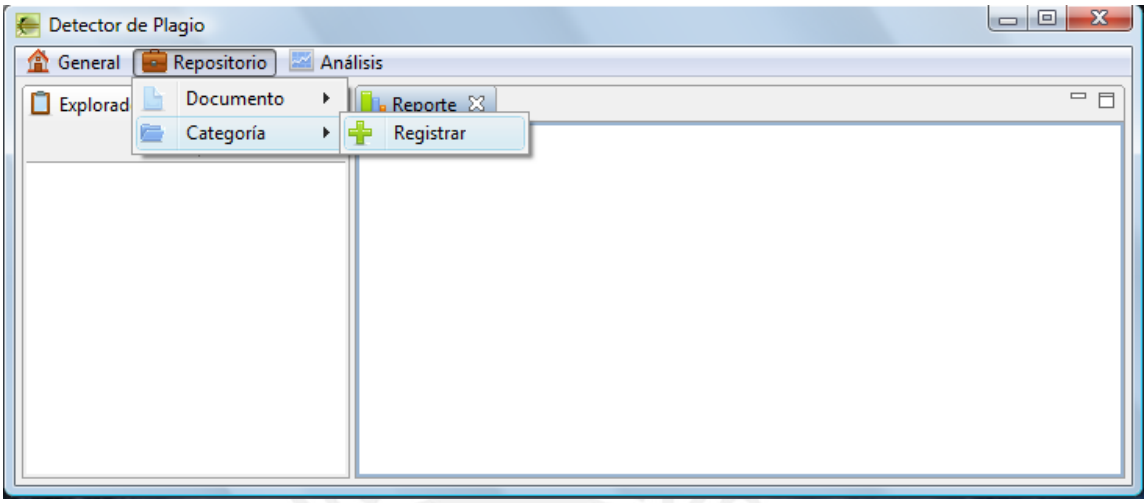

Figura 4.4 - Acceso a la opción de registro de categorías

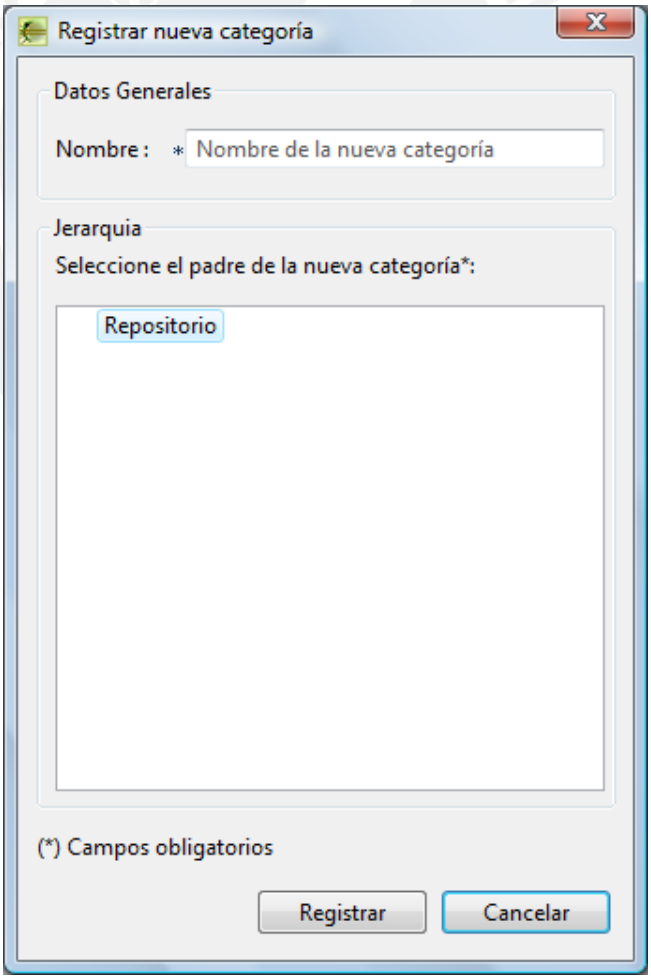

Figura 4.5 - Pantalla Registrar Categoría

La pantalla de Registro de Categoría (Figura 4.5) permite al usuario registrar una nueva categoría en el repositorio.

En el campo **Nombre** se ingresa el nombre que poseerá la categoría a registrar.

En el campo **Jerarquía** se muestra un árbol donde se listan todas las categorías establecidas por el usuario dentro del repositorio. Se deberá indicar qué categoría padre posee la nueva categoría a ingresar.

4.3.4 Pantalla de Registro de Documentos

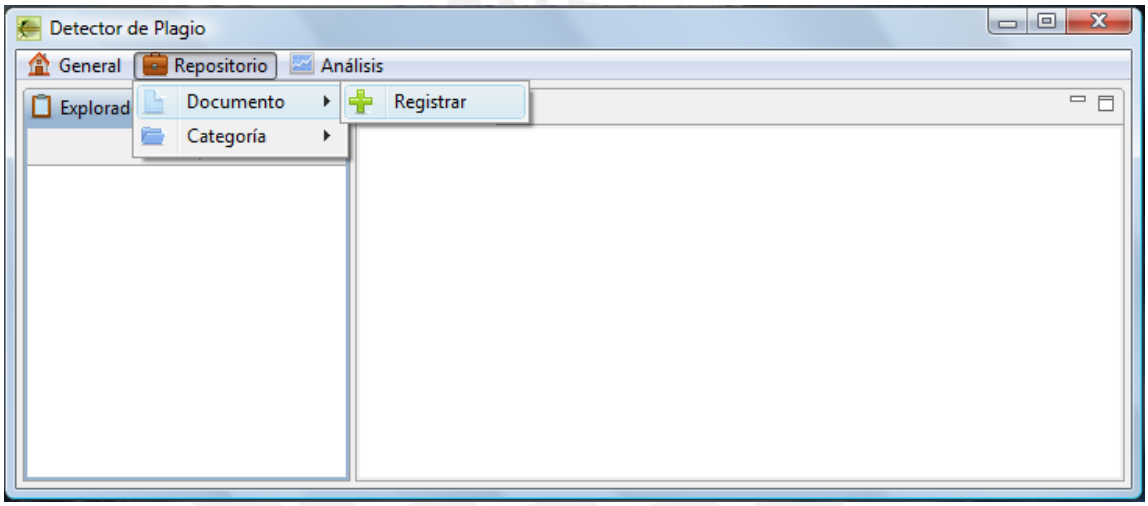

Figura 4.6 - Acceso a la opción de registro de documentos

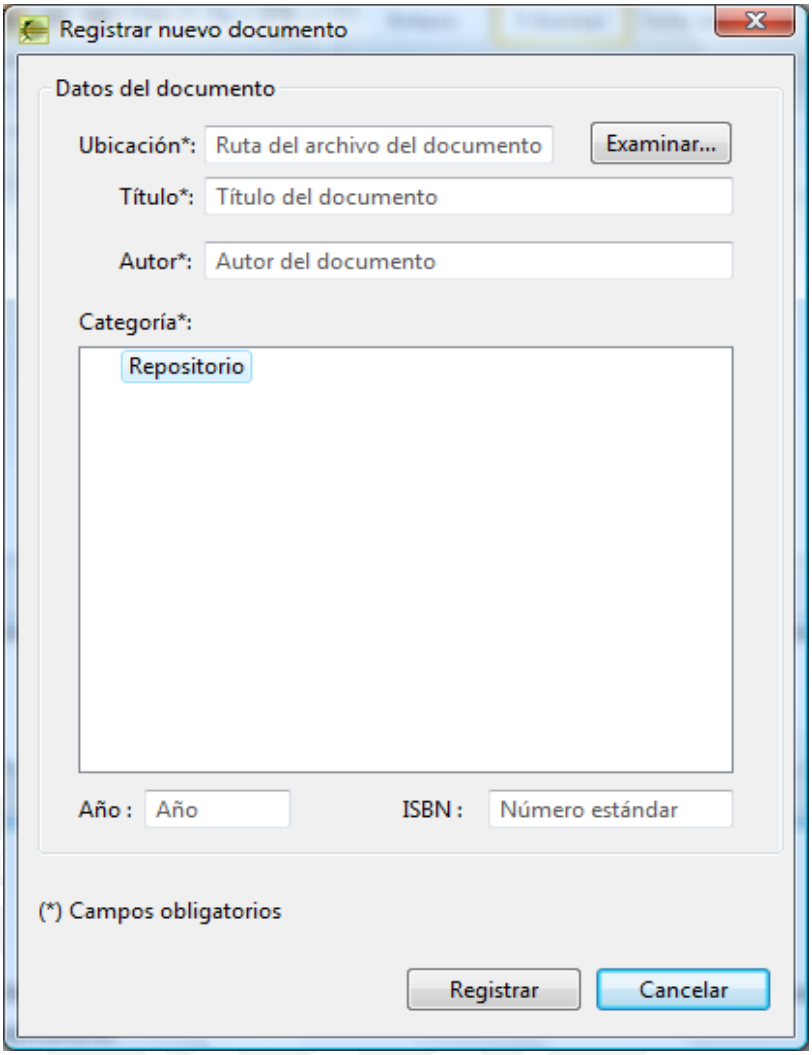

Figura 4.7 - Pantalla Registrar Documento

La pantalla de Registro de Documento (Figura 4.7) permite al usuario registrar un nuevo documento en el repositorio.

En el campo **Ubicación** se ingresa la ruta en la que se encuentra el documento dentro del disco. Para realizarlo, se puede emplear el botón **Examinar** que permite la exploración de carpetas del disco en una interfaz similar a la del Explorador de Windows. Otros datos a ingresar son: **Título, Autor, Año** e **ISBN.**

En el campo **Categoría** se muestra un árbol donde se listan todas las categorías establecidas por el usuario dentro del repositorio. Se deberá indicar a qué categoría pertenece el documento que se desea registrar.

4.3.5 Wizard de Análisis de Documentos

La wizard de Análisis de Documentos permite al usuario comparar un documento sospechoso con los que ya se encuentran registrados en el repositorio de documentos. En la Figura 4.8 se muestra cómo se puede acceder a este wizard.

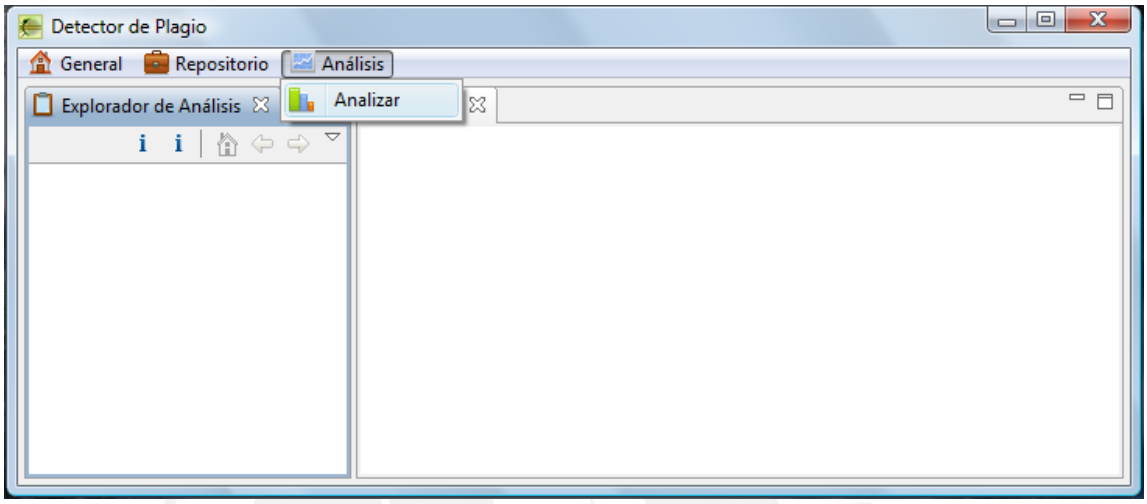

Figura 4.8 – Acceso a la opción de análisis de documento

Una vez que se ha accedido a la opción, se debe seleccionar el documento sospechoso a analizar. Este puede ser uno que ya fue registrado previamente (Figura 4.9) en el repositorio o, si así se desea, puede registrarse uno nuevo (Figura 4.10).

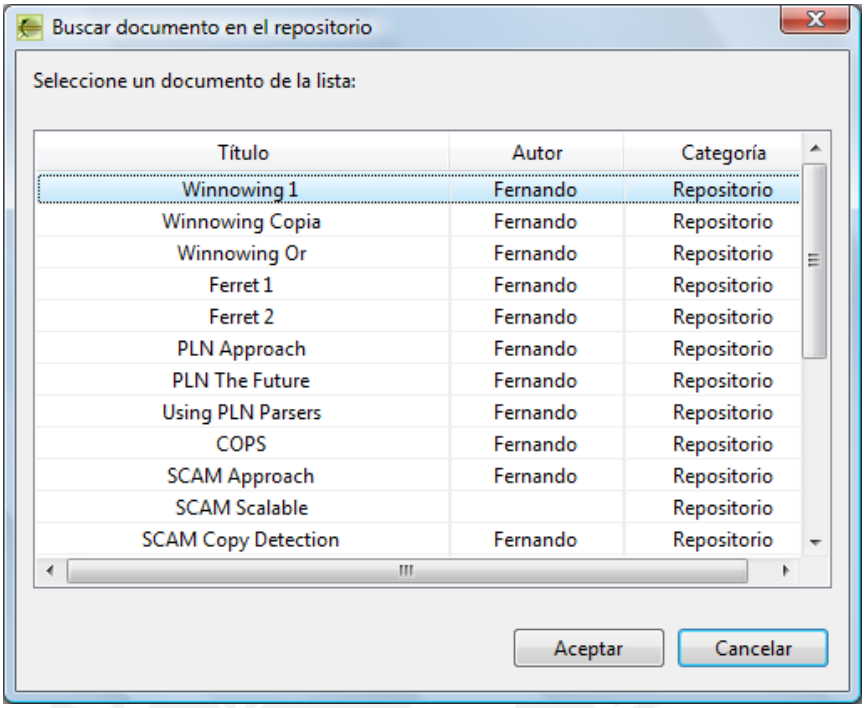

Figura 4.9 - Wizard Análisis de Documento: Selección de documento (documento ya registrado)

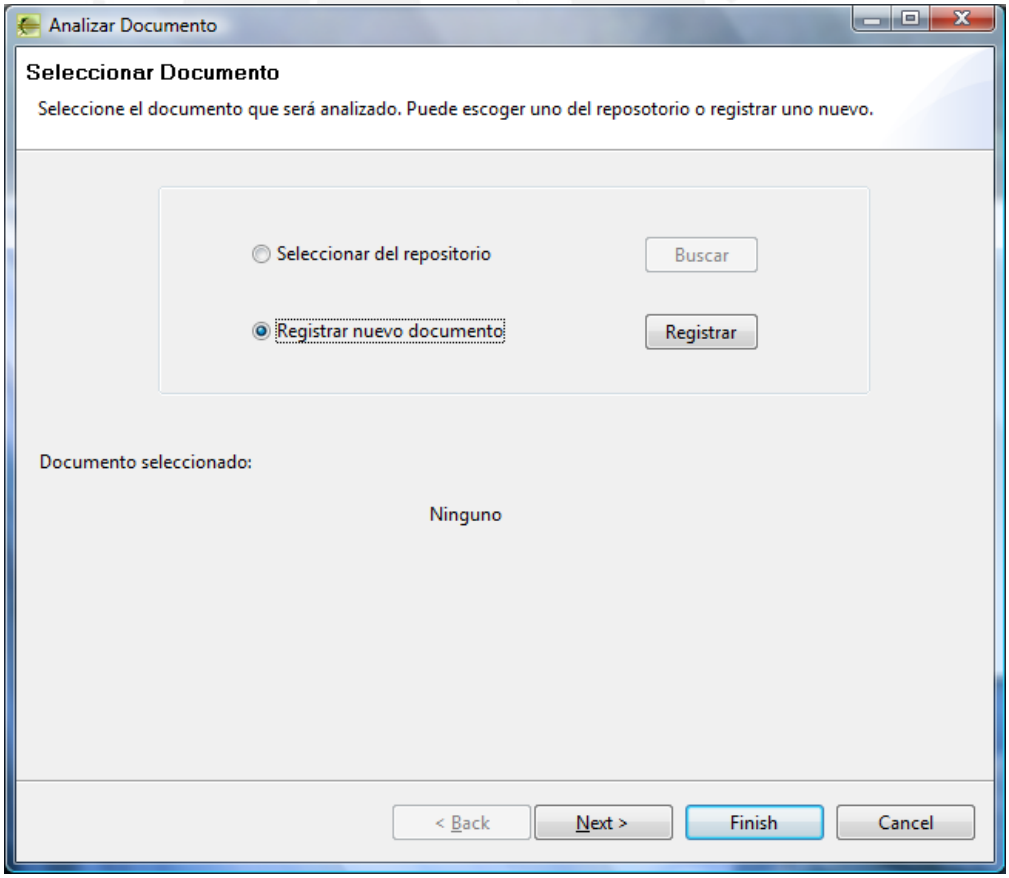

Figura 4.10 - Wizard Análisis de Documento: Selección de documento (nuevo registro)

Finalmente, se debe seleccionar la contraparte de comparación del documento sospechoso (Figura 4.11). Se puede indicar que se desea comprar el documento contra todos los que ya se encuentran registrados en el repositorio o, quizás, solamente contra aquellos que pertenezcan a alguna categoría en particular.

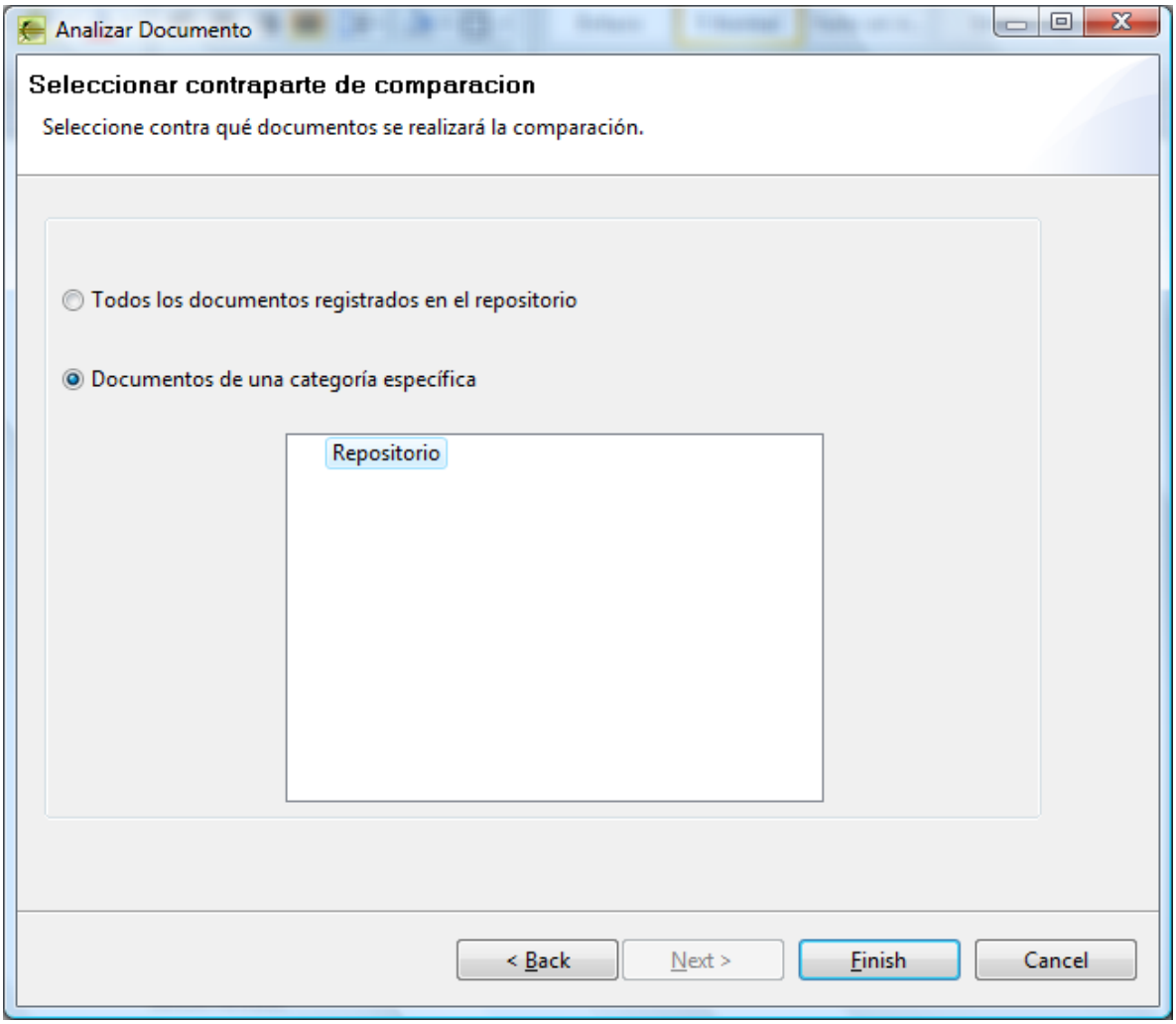

Figura 4.11 - Wizard Análisis de Documento: Selección de contraparte de comparación

4.3.6 Vista de Reporte de Resultados

En la vista Reporte se podrán visualizar diferentes reportes relacionados con el análisis realizado.

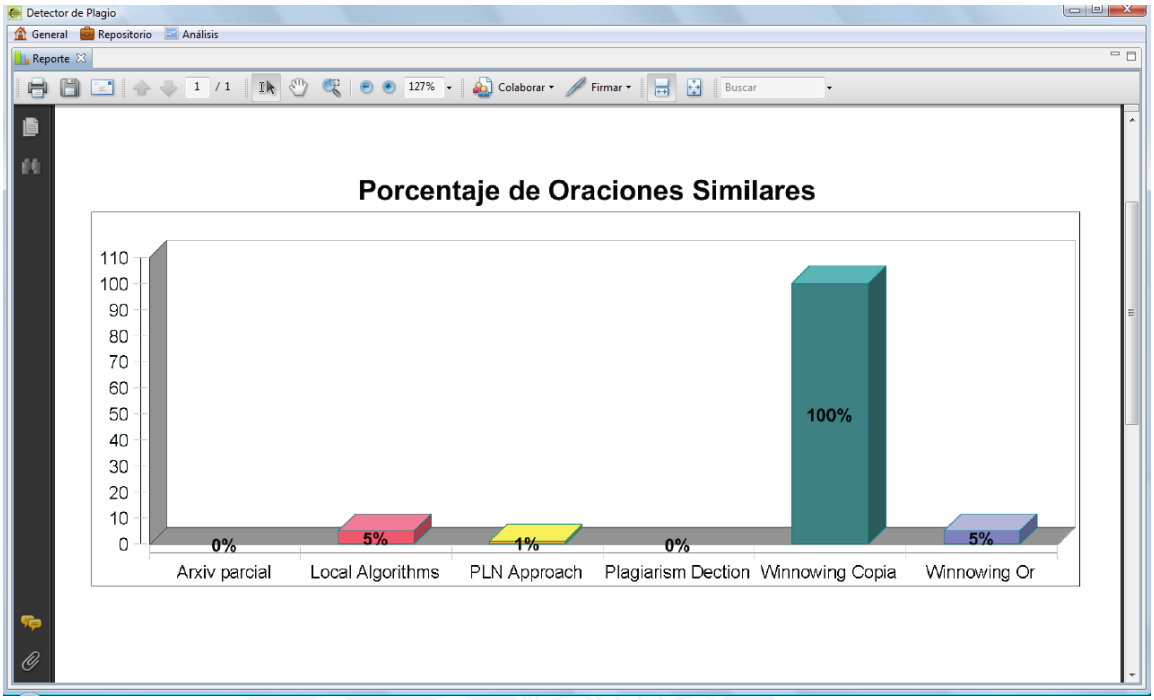

Figura 4.12 - Vista Reporte: Porcentaje de Oraciones Similares

En el reporte Porcentaje de Oraciones Similares, se muestra un gráfico de barras, para cada documento, el porcentaje de similitud que posee con el documento que se está evaluando.

En el caso del reporte mostrado en la Figura 4.12, el documento sospechoso tiene un 5% de similitud con el documento “Local Algorithms”, 1% con “PLN Approach”, 5% con “Winnowing Or” y 100% de similitud con “Winnowing Copia”.

Asimismo, se puede apreciar la barra de herramientas de Adobe Acrobat, lo cual permite que se puedan realizar funcionalidades propias de ese sistema, como son el guardar el reporte como un archivo PDF, imprimir, entre otros.

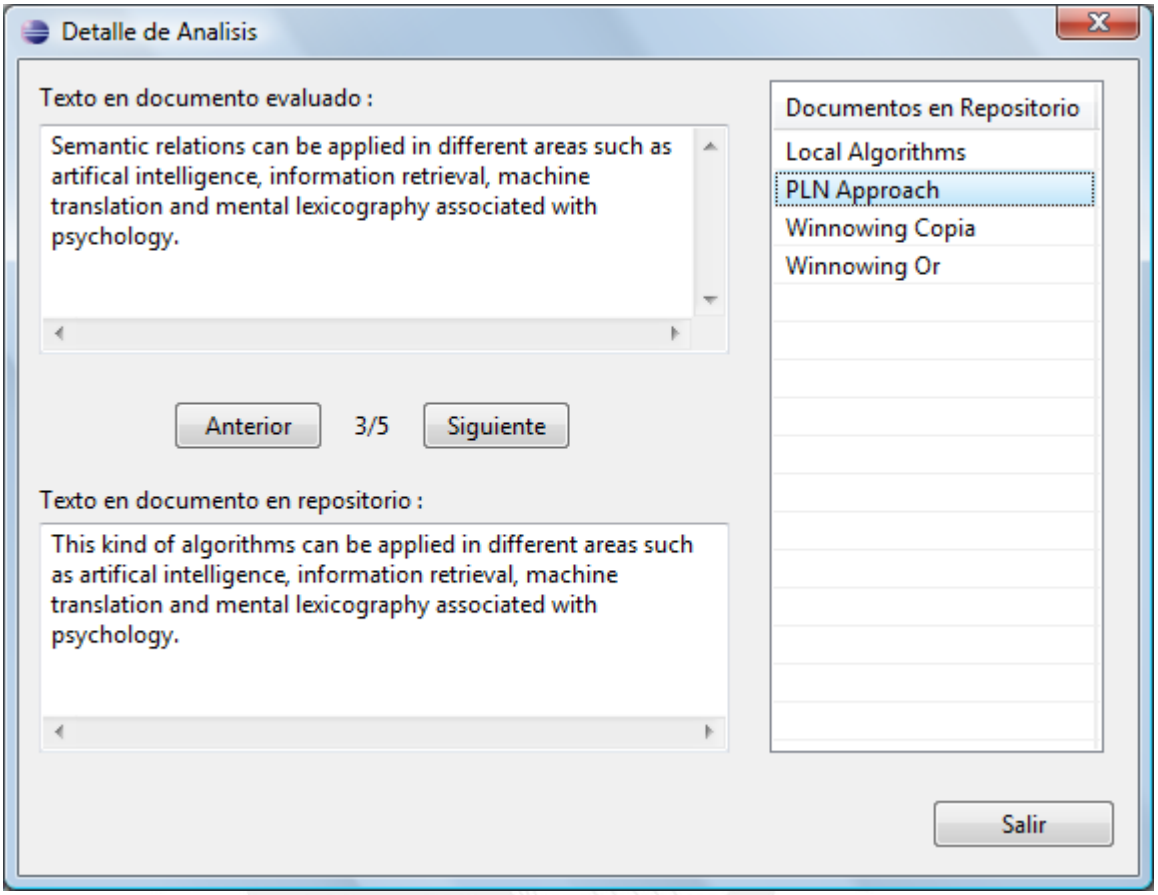

Figura 4.13 - Diálogo de detalle de análisis

Adicionalmente al gráfico mostrado anteriormente, se puede visualizar a detalle las oraciones que han sido detectadas como plagiadas, a través de una comparación entre la versión de la misma en el documento sospechoso y su contraparte en alguno de los documentos del repositorio con el cual se han hallado coincidencias. Como se puede apreciar en la Figura 4.13, en la parte derecha se listan los documentos del repositorio con el cual se encontraron similitudes. En la parte superior, aparece la oración sospechosa de plagio que aparece en el documento que se está evaluando. Al seleccionar un archivo de la lista, aparece en la parte inferior la versión de esta oración en el documento registrado en el repositorio. Empleando los botones de la parte central, se puede navegar entre las diferentes oraciones con solapamiento.

"La conclusión es que sabemos muy poco y, sin embargo, es asombroso lo mucho que conocemos. Y más asombroso todavía que un conocimiento tan pequeño pueda dar tanto poder."
Bertrand Arthur William Russell

5 Observaciones, conclusiones y recomendaciones

Lo que se expone en la presente sección del documento son las observaciones realizadas, las conclusiones a las que se ha llegado y las recomendaciones planteadas luego del desarrollo del presente documento.

5.1 Observaciones

- Existe una gran necesidad a nivel mundial de poder verificar la autenticidad de los documentos inéditos que son presentados en diferentes contextos académicos.

- Existen muchas herramientas en el mercado que se encargan de realizar la tarea de poder comparar documentos en busca de solapamientos entre ellos y así verificar la autenticidad de los mismos.

- Cada una de las herramientas que existen en el medio, tanto de carácter comercial como académico, emplea un método distinto para poder lograr su objetivo de verificación de plagio.

- Hasta la actualidad, el campo de la detección de plagio en documentos digitales sigue siendo explorado dado que surgen nuevos y mejores métodos que permiten incrementar la efectividad en el reconocimiento y disminuir el tiempo de respuesta.

5.2 Conclusiones

- Las técnicas de reconocimiento de patrones de texto pueden ser empleadas de manera exitosa para la detección de plagio en documentos digitales, puesto que permiten una comparación exitosa de cadenas de caracteres comunes entre diferentes archivos.

- Existe una gran variedad de algoritmos que permiten la detección de patrones comunes de texto entre diferentes documentos, lo cual provee la ventaja de poseer un amplio de campo de exploración para pruebas de eficiencia y eficacia de algoritmos y así seleccionar el más apropiado.

- El algoritmo para obtención de firmas *Winnowing* permite realizar el procesamiento inicial para la comparación de documentos en la búsqueda de secciones comunes. Dicho algoritmo no sólo provee un método eficaz que ya ha sido probado en sistemas existentes, sino que es de rápido entendimiento facilitando su implementación en el sistema que se plantea en el presente documento.

- El sistema presentado en este documento, el cual implementa la variante *Winnowing* del algoritmo *Document Fingerprinting*, permite la detección de plagio entre documentos, empleando una arquitectura de tipo repositorio central de documentos. Se realizaron las pruebas necesarias para asegurar que el sistema, verdaderamente, cumple el objetivo para el cual fue planteado.

5.3 Recomendaciones y trabajos futuros

- Si bien técnicas de reconocimiento de patrones de texto son bastante útiles para la detección de plagio documentario dado que presentan un bajo tiempo de respuesta, su precisión se puede ver afectada por la habilidad del plagiario al

modificar de tal manera el documento que dichos métodos no puedan detectarla. Esto se debe, principalmente, a que los métodos descritos basan su capacidad de reconocimiento en la estructura sintáctica de las oraciones y no así en el significado de las mismas. Es por tanto necesario poder desarrollar métodos que puedan seguir brindando un tiempo de respuesta bajo pero cuya vulnerabilidad al fallo sea menor.

- Recientes investigaciones apuntan al empleo de técnicas de procesamiento de lenguaje natural para el reconocimiento de plagio. Esto dado que los métodos PLN se basan en la semántica de las oraciones que conforman los documentos, logrando así mayor precisión en el reconocimiento y disminuyendo su vulnerabilidad a fallos. El punto débil de dichos métodos es que su tiempo de respuesta es bastante alto debido al procesamiento tan pesado que deben de realizar. Es por tanto necesario poder analizar dichos métodos e identificar aquellas secciones que pueden ser optimizadas de tal forma que su eficiencia sea mejorada.

Se recomienda emplear un método híbrido de técnicas tanto de reconocimiento de patrones de texto como de PLN para poder desarrollar un sistema que sea tanto eficiente como preciso al momento de detectar plagios. El sistema podría emplear técnicas de PLN para poder obtener una estructura estándar en todas las oraciones que componen el texto y luego emplear un método como el indicado en este documento para el almacenamiento y posterior comparación de las oraciones. Esto lograría aumentar la efectividad del sistema al reducir el riesgo de no detectar plagio por parafraseo.

Bibliografía

[1] Real Academia Española. Diccionario de la Lengua Española - Vigésima segunda edición. [Online]. http://www.rae.es

[2] Vicerrectorado Académico PUCP, "Por qué y cómo debemos combatir el plagio," Lima,.

[3] Jason Stephens, Michael Young, and Thomas Calabrese, "Does Moral Judgment Go Offline When Students Are Online? A Comparative Analysis of Undergraduates' Beliefs and Behaviors Related to Conventional and Digital Cheating," *Ethics & Behavior*, 2007.

[4] Santos Urbina Ramírez, "Ciberplagio: "construyendo" trabajos universitarios," in *EDUTEC 2004*, Barcelona, 2004.

[5] Rubén Comas and Jaime Sureda, "Ciber-Plagio Académico. Una aproximación al estado de los conocimientos," *Revista TEXTOS de la CiberSociedad*, no. 10, 2007.

[6] Alexa The Web Information Company. Top Sites in Spanish. [Online]. http://www.alexa.com/site/ds/top_sites?ts_mode=lang&lang=es

[7] Lucio Mendieta y Núñez, *Ensayo sobre el plagio y la investigación : el caso de Juan Comas*. México: Cultura, 1966.

[8] (2009, Noviembre) PsycCrawler Online User's Guide. [Online]. http://www.psychcrawler.com/plweb/info/help/dbfund.html

[9] Norma Edith Herrera, Base de Datos de Texto, Octubre 2006.

[10] J. (revisado y ampliado por GTI-IIE) Kittler, Notas del seminario de Reconocimiento de Patrones de Grupo de Tratamiento de Imágenes del Instituo de Ingeniería Eléctrica en la Univ. de Surrey, Setiembre 1, 2002.

[11] Lauro Soto. Arboles de Decisión. [Online]. http://www.mitecnologico.com/Main/ArbolesDeDecision

[12] Recuperación con árboles de decisión. [Online]. http://recuperacionorganizacion.googlepages.com/representacion.html

[13] Arboles Trie Patricia. [Online]. http://ingecomp.mforos.com/674005/3112616-arboles-trie-patricia/

[14] Eduardo Morales and L. Enrique Sucar. (1999, Junio) Redes Bayesianas. [Online]. http://ccc.inaoep.mx/~emorales/Cursos/RdeC/node153.html

[15] José L. Ruiz Reina, Introducción a las Redes Bayesianas, 2006.

[16] Luis Alonso Romero and Teodoro Calogne, Redes Neuronales y Reconocimiento de Patrones.

[17] Emiliano Aldabas-Rubira, "Introducción al reconocimiento de patrones mediante redes neuronales," in *IX Jornades de Conferències d'Enginyeria Electrònica del Campus de Terrassa*, 2002.

[18] Raúl Solórzano, Derecho para Ingenieros Informáticos Guía 1, 2008.

[19] Organización Mundial de la Propiedad Intelectual. Resumen del Convenio de Berna para la Protección de las Obras Literarias y Artísticas (1886). [Online]. http://www.wipo.int/treaties/es/ip/berne/summary_berne.html

[20] (2009, Junio) Mountain Goat Software. [Online]. http://www.mountaingoatsoftware.com/scrum

[21] Seo Jangwon and W. Bruce Croft, "Local text reuse detection," *Proceedings of SIGIR'2008*, pp. 571-578, 2008.

[22] Saul Schleimer, Daniel S. Wilkerson, and Alex Aiken, "Winnowing: local algorithms for document fingerprinting," *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pp. 76-85, 2003.

[23] S. Brin, J. Davis, and H. García-Molina, "Copy detection mechanisms for digital documents," *Proceedings of the 1995 ACM SIGMOD Int. Conf. on Management of data*, pp. 398-409, 1995.

[24] T. Lancaster and F. Culwin, Classifications of Plagiarism Detection Engines, 2005.

[25] Romans Lukashenko, Vita Graudina, and Janis Grundspenkis, "Computer-based plagiarism detection methods and tools: an overview," in *International Conference on Computer Systems and Technologies - CompSysTech'07*, 2007.

[26] Joanna Bull, Carol Collins, Elisabeth Coughlin, and Dale Sharp,

"Technical Review of Plagiarism Detection Software Report," Computer Assisted Assessment Centre, Luton, 2001.

[27] N. Shivakumar and García-Molina H., "The SCAM Approach to Copy Detection in Digital Libraries," in *D-Lib Magazine*, 1995.

[28] N. Shivakumar and H. García-Molina, "SCAM: A Copy Detection Mechanism for Digital Documents," in *Proceedings of the Second Annual Conference on the Theory and Practice of Digital Libraries*, 1995.

[29] A Si, H. V. Leong, and R. W. Lau, "CHECK: A Document Plagiarism Detection System," in *Proceedings of ACM Symposium for Applied Computing*, 1997, p. 70–77.

[30] K. Monostori, A. Zaslavsky, and Bia A., "Using the MatchDetectReveal System for Comparative Analysis of Texts," in *Proceedings of the Sixth Australasian Document Computing Symposium (ADCS 2001)*, Pacific Bay Resort, Coffs Harbour, 2001, pp. 51-58.

[31] K. Monostori, A. Zaslavsky, and H. Schmidt, "Efficiency of Data Structures for Detecting Overlaps in Digital Documents," in *Proceedings of the 24th Australasian Computer Science Conference*, Queensland, 2000, pp. 140-147.

[32] Stefan Kurtz, "Reducing the space requirement of suffix trees," *Software—Practice & Experience*, vol. 29, no. 13, pp. 1149-1171, Noviembre 1999.

[33] Carlos Enrique Mesones Barrón, *Comprensión y generación de lenguaje natural en un sistema de diálogo usando inteligencia artificial para servicios telefónicos de información de cines*. Lima, Perú, 2006.

[34] Zdenek Ceska, "The Future of Copy Detection Techniques," *Proceedings of the 1st Young Researchers Conference on Applied Sciences (YRCAS 2007)*, pp. 5-10, Noviembre 2007.

[35] Chi-Hong Leung and Yuen-Yan Chan, "A Natural Language Processing Approach to Automatic Plagiarism Detection," in *Proceedings of the 8th ACM SIGITE conference on Information technology education*, Destin, Florida, EE.UU., 2007, pp. 213-218.

[36] (2009, Junio) XProgramming.com. [Online]. http://www.xprogramming.com/xpmag/whatisxp.htm

[37] Henrik Kniberg, *Scrum and XP from the Trenches*, C4Media, Ed. EE.UU.: InfoQ.com, 2007.

[38] (2009, Junio) Manifesto for Agile Software Development. [Online]. http://www.agilemanifesto.org/

[39] D. Campo, P. Latorre, and C. Higgins, "Interfaz de Usuario en Sistemas de Detección de Plagio," in *Congreso Español de Informática*, 2005.

[40] Thomas Lancaster and Fintan Culwin, "Using freely available tools to produce a partially automated plagiarism detection process," in *Beyond the comfort zone: Proceedings of the 21st ASCILITE Conference*, Perth, 2004, pp. 520-529.

[41] Ministerio de Administraciones Públicas. MÉTRICA VERSION 3 - Estudio de Viabilidad del Sistema. [Online]. http://www.csae.map.es/csi/metrica3/evs.pdf

[42] S. Brin, J. Davis, and H. Garcia-Molina, "Copy detection mechanisms for digital documents," *Proceedings of the 1995 ACM SIGMOD Int. Conf. on Management of data*, pp. 398-409, 1995.

[43] D. Sorokina, J. Gehrke, S. Warner, and P. Ginsparg, "Plagiarism Detection in arXiv," in *Technical Report TR2006-2046*, Computing and Information Science, Cornell University, 2006.

[44] Raphael Finkel, Arkady Zaslavsky, Krisztián Monostori, and Heinz Schmidt, "Signature extraction for overlap detection in documents," in *Proceedings of the twenty-fifth Australasian conference on Computer science*, Melbourne, Victoria, Australia , 2002, pp. 59 - 64.

[45] (2009, Noviembre) Apache PDFBox. [Online]. http://pdfbox.apache.org/

[46] (2009, Noviembre) LingPipe. [Online]. http://alias-i.com/lingpipe/index.html