Facultad 10

Asignación automatizada de categorías temáticas al contenido textual de documentos HTML

Trabajo de Diploma para optar por el Título de
Ingeniero en Ciencias Informáticas

Autores: Yandry Pérez Clemente
Dovier Antonio Ripoll Méndez

Tutor: Lic. Isachi Abreu Gil

Ciudad de la Habana, Cuba, Junio 2008

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Facultad 10 de la Universidad de las Ciencias Informáticas, así como a dicho centro, para que lo usen según estimen pertinente.

Para que así conste, firmamos la presente a los ___ días del mes de _____ del año _____ .

Yandry Pérez Clemente

_____

Firma del Autor

Dovier Antonio Ripoll Méndez

_____

Firma del Autor

Lic. Isachi Abreu Gil

_____

Firma del Tutor

DATOS DE CONTACTO

Isachi Abreu Gil, Licenciada en Ciencias de la Computación. Actualmente trabaja en el Departamento de Programación de la Facultad 10 en la Universidad de las Ciencias Informáticas (UCI). Pertenece, desde el año 2007, al Grupo de Análisis y Procesamiento de Información (API) del Polo Productivo Centro de Estudios de Internet (CENTERNET).
Correo electrónico: isachi@uci.cu

AGRADECIMIENTOS

Agradezco:

A mis abuelos Paco y Olimpia; mis tías Amalita, Doris y Lachy; mis tíos Ale, Juanki, Oscar y Paquito; y mis primitas (las más lindas del mundo) Daniela, Isabel, Lisandra, Samantha, Greta y Beverley; por darme todo su apoyo y confiar en mí.
A toda mi familia por ser la mejor del mundo.
A mi noviecita linda Diana 'la nena', mi princesita, por estar a mi lado estos 5 años de universidad y los 67 que nos quedan, in joy and sorrow.
A J.J e Irene por acogerme como un hijo más.
A mis amigos de la vieja escuela Ernesto, Ernesto Miguel, Marvin y Maurys y de la nueva escuela Dovier, Israel y Juan Carlos; gracias a ustedes puedo decir que tengo hermanos.
A Danielito, el mejor socio del barrio.
A Héctor Rodríguez Figueredo, de las buenas ideas que se nos han ocurrido, usted nos ha apoyado en el 110% de ellas.
A Alain Ramos Medina, Erick Bacallao Pedraza y Rubén de León Becerra por enseñarme a leer y escribir... en Pascal.
A todos los miembros del Grupo Sherpa, Distro Nova y CENTERNET por haberme enseñado tanto y por hacerme reír mientras trabajaba.
A nuestra tutora Isachi, por toda su ayuda.
A Jonathan Davis, Brian Warner y Kurt Cobain, por darme buena música para escuchar.

Yandry Pérez Clemente

AGRADECIMIENTOS

Agradezco:

A mis padres Lázaro Antonio Ripoll y Martha Méndez, por traerme a la vida...
A mis abuelos Ramón Muiño y Haydée Marrero, por darme la maravillosa vida que tengo...
A mi novia Yurisleidy Hernández, por todo su amor y estar siempre a mi lado...
A mis amigos Vismar Hernández y Roberto Menéndez, por mostrarme la vida de otro modo...
A mis profesores de la secundaria Irán Reina, Erick Lima y Miguel Antonio Pelegrín, por creer tanto en mi futuro...
A mi decano de facultad Hector Rodríguez, por su incondicional apoyo y confianza...
A mis compañeros de proyectos, por toda la ciencia que hemos hecho...
A mi compañero de competencias y tesis Yandry Pérez, por todo lo que hemos logrado...
A mi tutora Isachi Abreu, por su incondicional ayuda...
A nuestro compañero Fidel Castro, por ser artífice de la Revolución y la Universidad que tanto me han dado...
A todas las personas que, de una forma u otra, han hecho posible que hoy escriba (con la satisfacción del deber cumplido) estas líneas de agradecimientos...

Dovier Antonio Ripoll Méndez

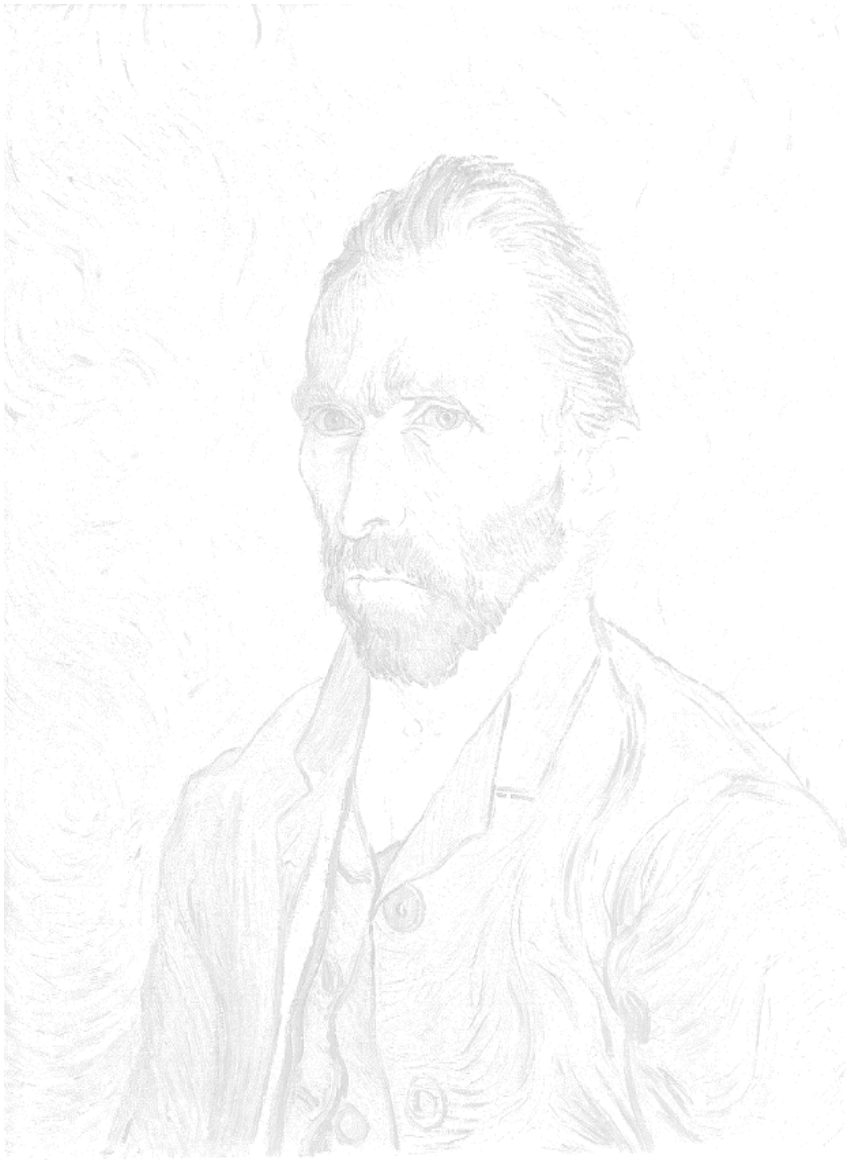
DEDICATORIA

Dedico este trabajo a mi mamá, Haydée Pérez Clemente, por su cariño y por estar siempre pensando en mí. Entre todas las mujeres del mundo tú eres la que más quiero. Mima: este logro también es tuyo, porque tú eres mi fuente de inspiración.

Yandry Pérez Clemente

Dedico este Trabajo de Diploma a mis abuelos Ramón Muiño y Haydée Marrero, por lo mucho que me quieren, cuidan, guían e incentivan...

Dovier Antonio Ripoll Méndez



¿Qué sería de la vida si no tuviéramos el valor de intentar algo nuevo?

Vincent Willem Van Gogh
(pintor holandés, 1853 - †1890)

OBSERVACIONES INTRODUCTORIAS

- **Marcas comerciales y marcas de servicio**.
  Todas las marcas comerciales, marcas de servicio, logotipos y nombres de compañías mencionadas en este trabajo son propiedad de sus respectivos dueños.
- **Glosario de términos**.
  Se recomienda consultar la sección Glosario de Términos antes de comenzar a leer el texto principal de este trabajo. Los términos aparecen ordenados alfabéticamente.
- **Notas al pie de página**.
  Aparecen numeradas al pie de página y proporcionan información adicional sin interrumpir la secuencia lógica del texto principal.
- **Referencias bibliográficas**.
  Desde el texto principal, se indican entre paréntisis con el primer apellido del(de los) autor(es) y el año de publicación de la obra, por ejemplo (PÉREZ y RIPOLL, 2008). Posteriormente, en la sección Referencias Bibliográficas aparecen ordenadas alfabéticamente y con los datos completos de cada obra:

  PÉREZ, Y. y RIPOLL, D. A. *Asignación automatizada de categorías temáticas al contenido textual de documentos HTML*. Tesis (Diploma). Ciudad de la Habana, Cuba: Universidad de las Ciencias Informáticas, Facultad 10, 2008. 94 p. Disponible en: http://biblioteca.uci.cu

- **Formatos para lectura**.
  Este trabajo se ha optimizado para uso digital e impreso con formato de papel A4. No obstante, se recomienda usar la versión digital cuando así proceda; puesto que es un archivo PDF, generado a partir de un documento ODT creado en OpenOffice Writter, con hipervínculos que facilitan la navegación.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

LISTA DE ACRÓNIMOS

- **ACC** - Combinación Analítica de Criterios, (*Analytical Combination of Criteria*).
- **API** - Análisis y Procesamiento de Información.
- **ASADOHTML** - ASignador Automatizado de categorías temáticas al contenido textual de DOcumentos HTML.
- **ASP** - *Active Server Pages*.
- **BDUC** - Base de Datos de URLs Categorizadas.
- **CPAN** - *Comprehensive Perl Archive Network*.
- **DTD** - Definición del Tipo de Documento, (*Document Type Definition*).
- **Filpacon** - Filtrado de paquetes por contenidos.
- **HTML** - Lenguaje para el Formato de Documentos de Hipertexto, (*HyperText Markup Language*).
- **JSP** - *Java Server Pages*.
- **MCADHTML** - Motor de Categorización Automatizada de Documentos HTML.
- **PHP** - *Hypertext Preprocessor*.
- **POD** - Documentación Simple, (*Plain Old Documentation*).
- **RUP** - Proceso Unificado de Rational, (*Rational Unified Process*).
- **SGML** - *Standard Generalized Markup Language*.
- **SVM** - Máquinas de Soporte Vectorial, (*Support Vector Machines*).
- **TCP/IP** - Protocolo de Control de Transmisión/Protocolo de Internet, (*Transmission Control Protocol/Internet Protocol*).
- **UCI** - Universidad de las Ciencias Informáticas.
- **UML** - Lenguaje Unificado de Modelado, (*Unified Modeling Language*).
- **URL** - Localizador Uniforme de Recurso, (*Uniform Resource Locator*).
- **XML** - Lenguaje de Marcado eXtensible, (*eXtensible Markup Language*).

RESUMEN

Desde el año 2005, en la Universidad de las Ciencias Informáticas (UCI) se desarrolla un producto informático denominado Filpacon, destinado a regular (permitir o denegar) el acceso de usuarios a determinados contenidos de Internet que les resulten inadecuados. En la actualidad dicho *software* comercial basa su filtrado en una Base de Datos de URLs Categorizadas (BDUC), creada y actualizada a partir de listas de URLs categorizadas por DMOZ, URLBlackList.com y Shalla. La construcción, el mantenimiento y las condiciones de uso de tales listas generan inconvenientes que motivaron el desarrollo de un Motor de Categorización Automatizada de Documentos HTML (MCADHTML) que, con técnicas de Inteligencia Artificial, sea capaz de eliminar la mencionada dependencia. Por lo anterior, el propósito del presente trabajo consistió en el diseño de un sistema informático (ASADOHTML) de asignación automatizada de categorías temáticas al contenido textual de documentos HTML, para contribuir al proceso de crear y actualizar la BDUC de Filpacon. Para ello, inicialmente, fue necesaria la revisión de la teoría relacionada con la Categorización Automatizada de Documentos HTML luego, empleando el Proceso Unificado de Rational, se definieron las características de ASADOHTML y se efectuó su Análisis y Diseño. Entre sus características distintivas, ASADOHTML estará orientado al Filtrado de Contenidos, representará los documentos de manera autocontenida, usará la información aportada por etiquetas HTML y empleará el Algoritmo de Aprendizaje Automático Máquinas de Soporte Vectorial. Al concluir el presente trabajo se obtuvo, principalmente, la documentación necesaria para la futura implementación de ASADOHTML y una base teórica para el desarrollo de sistemas afines.

PALABRAS CLAVES

categorización, automatizada, documento, HTML, Filpacon, BDUC, MCADHTML, ASADOHTML.

INTRODUCCIÓN

La Categorización Automatizada de Documentos HTML emerge como una de las soluciones de mayor potencial para tratar, en el contexto del Filtrado de Contenidos, el exponencial ritmo de cambio y crecimiento que experimenta la Web. El objetivo de un sistema de categorización es encontrar el conjunto de categorías más cercano para un determinado documento, a partir de una jerarquía de categorías previamente creada y compuesta por una colección de documentos de referencia. Por su parte, el Filtrado de Contenidos es utilizado principalmente por empresas, escuelas y grupos familiares para regular (permitir o denegar) el acceso a contenidos de Internet que consideran inadecuados para sus empleados, estudiantes e hijos respectivamente.

Existen dos métodos principales para filtrar contenidos de la Web, mediante Base de Datos de URLs Categorizadas (en lo adelante BDUC) y el Análisis Dinámico de Contenidos. En la Universidad de las Ciencias Informáticas (en lo adelante UCI[1]) se desarrolla desde el año 2005 una herramienta de filtrado, nombrada Filpacon, que basa su funcionamiento en el primer método mencionado. Comparando la URL pedida por un usuario contra las existentes en su BDUC, Filpacon permite o deniega su acceso según el tipo (ilícito, nocivo o adecuado) de la categoría asociada a dicha URL y la Política de Uso de Internet previamente definida. Para crear y actualizar la BDUC de tal filtro informático, se han creado programas que compendian listas de URLs categorizadas por DMOZ[2], URLBlackList.com[3] y Shalla[4].

La naturaleza dinámica de la Web y el hecho de ser Filpacon un producto con fines comerciales implican que la dependencia anterior no sea factible por presentar varios inconvenientes, principalmente relacionados con la construcción, el mantenimiento y las condiciones de uso (licencias) de dichas listas. A continuación se mencionan algunos problemas inherentes a lo anterior y relacionados con la Categorización Automatizada de Documentos HTML:

1 http://www.uci.cu

2 http://www.dmoz.org

3 http://www.urlblacklist.com

4 http://www.shallalist.de

- generalmente, la clasificación es realizada por personas que manualmente revisan y categorizan cada página web, lo cual constituye un proceso consumidor de tiempo y propenso a errores humanos.
- tales personas pueden usar criterios de categorización altamente dependientes de sus costumbres, conocimientos y compromisos con la labor que desempeñan.
- en ocasiones emplean técnicas muy generales o insuficientes para la categorización, tales como la identificación de palabras claves y la simple ocurrencia de estas en el texto.
- para analizar una página web sólo utilizan el texto visible para los usuarios, ignorando la valiosa información que es proporcionada por las notaciones propias del lenguaje HTML.
- las categorías asociadas a las URLs deben ser constantemente rectificadas, implicando que dichas personas deben verificar nuevamente tales URLs para detectar cambios en los contenidos que representan.

Por lo anteriormente explicitado, al Grupo de Análisis y Procesamiento de Información (en lo adelante API[5]) le asignaron el desarrollo de un Motor de Categorización Automatizada de Documentos HTML (en lo adelante MCADHTML) que, con técnicas de Inteligencia Artificial, sea capaz de crear y actualizar la BDUC de Filpacon. Por tal motivo surge el siguiente problema a resolver: ¿Cómo automatizar la asignación de categorías temáticas al contenido textual de documentos HTML para contribuir al proceso de crear y actualizar la BDUC de Filpacon?

La Categorización Automatizada de Documentos HTML es un caso peculiar de la Categorización Automatizada de Textos, la cual tiene una rica historia que se remonta a la década de los '60. Hasta fines de los '80, el enfoque predominante involucraba la construcción de categorizadores a partir de Ingeniería del Conocimiento. En los '90 un nuevo paradigma basado en el Aprendizaje Automático sustituye el enfoque anterior. Paralelamente, el surgimiento de la Web propició un vertiginoso crecimiento de documentos en formato digital, requiriendo la aplicación de técnicas automatizadas de categorización de documentos HTML para los más diversos fines. A partir de lo anterior, se define como objeto

5 API: grupo de investigación y desarrollo (I+D) que pertenece al Polo Productivo Centro de Estudios de Internet.

de estudio la Categorización Automatizada de Textos y como campo de acción la Categorización Automatizada de Documentos HTML.

La idea a defender, que se establece como base de la investigación efectuada, se puede formular en los siguientes términos: La asignación de categorías temáticas al contenido textual de documentos HTML puede ser automatizada, para contribuir al proceso de crear y actualizar la BDUC de Filpacon.

El objetivo general de este trabajo, derivado de la idea a defender, es el siguiente: Diseñar un sistema informático de asignación automatizada de categorías temáticas al contenido textual de documentos HTML, para contribuir al proceso de crear y actualizar la BDUC de Filpacon. El objetivo anterior puede articularse en los siguientes objetivos específicos:

- Revisar la teoría relacionada con la Categorización Automatizada de Documentos HTML para averiguar qué se ha dicho o investigado sobre el tema y reducir el ámbito de los hechos por tratar.
- Definir las características del Asignador Automatizado de Categorías Temáticas al Contenido Textual de Documentos HTML (en lo adelante ASADOHTML).
- Realizar el Análisis y Diseño de ASADOHTML.

Para cumplir los objetivos propuestos se requieren las siguientes tareas de investigación:

- Realizar investigación documental y bibliográfica sobre las siguientes temáticas:
  - La Web.
  - Lenguajes de marcado, especialmente el HTML y su vocabulario.
  - Filtrado de Contenidos y sus métodos principales.
  - Categorización Automatizada de Textos y Documentos HTML.
  - Representación de textos, especialmente los modelos vectoriales, las funciones de ponderación y las técnicas para la selección del vocabulario.
  - Algoritmos de Aprendizaje Automático.
  - Herramientas para construir categorizadores.
  - Proceso Unificado de Rational (en lo adelante RUP) para el desarrollo de sistemas orientados a objetos.
  - Investigaciones similares.

- Para definir las características de ASADOHTML, investigar sobre:
  - Estrategias para confeccionar el Modelo del Dominio.
  - Métodos para definir y especificar los requisitos y casos de uso.
- Para realizar el Análisis y Diseño de ASADOHTML, investigar sobre:
  - Pasos para construir diagramas de clases del Análisis y Diseño.

Un método de investigación provee estrategias elementales para ahorrar esfuerzo y tiempo en una investigación científica. Para guiar las tareas de investigación se utilizan principalmente los métodos de investigación Histórico-Lógico y Analítico-Sintético. El primero permite estudiar la trayectoria histórica (antecedentes) y la lógica de los temas a tratar; mientras que el segundo permite extraer las partes de un todo para estudiarlas por separado y reunir, racionalmente, varios elementos dispersos en una nueva (sintetizada) totalidad.

A continuación se describe brevemente la estructura del presente documento (organización de la memoria), resumiéndose el contenido de cada capítulo:

- En el Capítulo 1 (Fundamentación Teórica) se hará una breve revisión de los lenguajes de marcado, especialmente el HTML y su vocabulario. Del Filtrado de Contenidos se tratará: surgimiento, introducción en la UCI (Filpacon), ámbito tecnológico y social, métodos de filtrado y *software* de filtrado. Se introducirá la Categorización Automatizada de Textos y sus procesos fundamentales, para posteriormente presentar la Categorización Automatizada de Documentos HTML y principales enfoques de representación. Se mencionarán algunas herramientas para construir categorizadores y aspectos de RUP para el desarrollo de sistemas orientados a objetos. Finalmente, se presentarán investigaciones similares a la desarrollada en este trabajo y las conclusiones que reflejarán la posición de los autores respecto a la teoría tratada.
- En el Capítulo 2 (Características del Sistema) se realizará el modelado del dominio con el objetivo de comprender el contexto del sistema; se hará su propuesta, describiendo cómo debe funcionar y destacando sus características distintivas; se especificarán sus Requisitos Funcionales, No Funcionales y Adicionales y se elaborará el Modelo de Casos de Uso, definiendo sus actores, casos de uso y las relaciones entre ellos.

- En el Capítulo 3 (Análisis y Diseño del Sistema), de ASADOHTML, se presentarán: sus Diagramas de Clases del Análisis, mostrando las clases participantes y relaciones entre ellas; sus Diagramas de Secuencia, evidenciando el orden de las acciones en los casos de uso cuando son invocados y, de sus Clases del Diseño, el diagrama y una descripción detallada. Además, se mostrarán los patrones utilizados para su Diseño y las técnicas que facilitarán documentarlo para su uso y tratar posibles errores en tiempo de ejecución.

Finalmente se presentan las Conclusiones, Recomendaciones, Referencias Bibliográficas y el Glosario de Términos.

1. CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

La Web es un sistema de documentos digitales enlazados y accesibles a través de Internet (WIKIPEDIA, 2008). Tales documentos se conocen como páginas web y pueden:

- contener textos, imágenes, vídeos u otros contenidos multimedia.
- ser estáticas o dinámicas.
- agruparse en sitios web.
- ser cargadas desde un ordenador llamado servidor web (Apache e Internet Information Services, entre otros).

Generalmente, la visualización de una página web comienza cuando un usuario teclea su URL en un navegador web (Internet Explorer y Mozilla Firefox, entre otros) o sigue un hipervínculo a dicha página o recurso. En ese momento el navegador comienza una serie de comunicaciones, transparentes al usuario, para obtener los datos de la página y visualizarla (WIKIPEDIA, 2008).

Debido a la naturaleza dinámica de la Web, medir su tamaño se plantea complicado. No obstante, es posible realizar estimaciones respecto al tamaño de su porción visible (Web Visible) o conjunto de páginas accesibles desde los diferentes buscadores (Google[6] y Yahoo[7], entre otros). La otra porción es conocida como Web Invisible y está formada principalmente por información:

- almacenada en bases de datos y accesible por medio de lenguajes interpretados (PHP, ASP y JSP, entre otros) que se incorporan al código HTML de las páginas web (BERNERS-LEE *et al.*, 1992).

6 http://www.google.com

7 http://www.yahoo.com

- ajena a los recorridos efectuados por *robots* web[8] de buscadores, generalmente debido a que es mínimamente enlazada desde otros recursos, está protegida por contraseñas e incorpora protocolos que excluyen a tales agentes autónomos.
- excluida de los buscadores, entre otros motivos, para evitar saturar sus bases de datos con materiales no deseados (BARKER y KUPERSMITH, 2000).

En (BHARAT y BRODER, 1998) los autores estimaron el tamaño de la Web Visible para Hotbot[9], Altavista[10], Excite[11] e Infoseek[12] (los principales buscadores del momento) en 200 millones de páginas. Un año después, se hablaba de 800 millones (LAWRENCE y GILES, 1999). En enero del 2004, estudios del Internet Software Consortium Inc.[13] mencionaban que la cantidad de documentos accesibles en la Web se situaba alrededor de 550.000 millones (CÁCERES, 2004). Otro estudio, basado en datos del Search Engine Watch[14], mostraba que la cantidad de páginas web indexadas por los buscadores era 11,5 billones (GULLI y SIGNORINI, 2005). En mayo de 2008, un estudio de Netcraft[15] planteaba que habían recibido respuestas desde 168.408.112 sitios web. Además, en (O'NEILL, LAVOIE y BENNETT, 2003) puede encontrarse un estudio sobre la evolución de los contenidos públicos de Internet desde 1998 hasta el 2002. Por otra parte, y en términos de volumen de información, se estimaba que la Web Invisible (con 4.500 *terabytes*[16]) era de 400 a 550 veces mayor que la Web Visible (con 19 *terabytes*) (BERGMAN, 2001).

Además de su tamaño, otra característica propia de la Web es que permite una fácil y libre publicación de contenidos; prácticamente cualquier persona (sin distinción de tipo y condición), que posea conocimientos básicos sobre el uso de un editor de textos estándar (gedit y notepad, entre otros) y codificaciones en lenguaje HTML, puede crear y publicar una página web. Esto ha implicado que la información aparezca mínimamente estructurada y

8 http://www.robotstxt.org

9 http://www.hotbot.com

10 http://www.altavista.com

11 http://www.excite.com

12 http://www.infoseek.com

13 http://www.isc.org

14 http://searchenginewatch.com

15 http://news.netcraft.com/archives/2008/05/06/may_2008_web_server_survey.html

16 terabyte: unidad de medida informática que es equivalente a 1.099.511.627.776 bytes o letras.

abarque todos los temas imaginables, incluyendo contenidos que resultan ilícitos[17] (pornografía infantil y violencia, entre otros), nocivos[18] (en el caso de menores de edad: pornografía) y adecuados[19] (educación y salud, entre otros) para determinados sectores de la sociedad (ALONSO-ALLENDE, 2001).

Por lo anterior, la Web se ha convertido en un escenario idóneo para la convergencia de investigadores en varios campos, tales como el Filtrado de Contenidos, la Categorización Automatizada de Textos y Documentos HTML y la Recuperación de Información, entre otros. Sin embargo, su propia naturaleza provoca que modelos aplicados con éxito en otros escenarios puedan ser poco efectivos en el tratamiento de la información contenida en la Web.

En este capítulo se hace una breve revisión de los lenguajes de marcado, especialmente el HTML y su vocabulario. Del Filtrado de Contenidos se trata: surgimiento, introducción en la UCI (Filpacon), ámbito tecnológico y social, métodos de filtrado y *software* de filtrado. Se introduce la Categorización Automatizada de Textos y sus procesos fundamentales, para posteriormente presentar la Categorización Automatizada de Documentos HTML y principales enfoques de representación. Se mencionan algunas herramientas para construir categorizadores y aspectos de RUP para el desarrollo de sistemas orientados a objetos. Finalmente, se presentan investigaciones similares a la desarrollada en este trabajo y las conclusiones que reflejan la posición de los autores respecto a la teoría tratada.

1.2. Lenguajes de Marcado

Con el desarrollo de editores y procesadores de textos surgen los primeros lenguajes informáticos especializados en tareas de descripción y estructuración de información: los lenguajes de marcado (FERNÁNDEZ-VALMAYOR *et al.*, 2006). De manera concreta, un lenguaje de marcado o etiquetas se define como un conjunto de reglas que permiten estructurar y dar formato a un documento (FRESNO, 2006).

17 contenidos ilícitos: merecen una respuesta penal por quebrantar un bien jurídico.

18 contenidos nocivos: ofensivos, pero no lo suficiente como para ser merecedores de una respuesta penal.

19 contenidos adecuados: aquellos que no son ni ilícitos ni nocivos.

Dentro de los lenguajes de marcado se distinguen dos tipos de estructura:

- estructura lógica, formada por las diferentes partes que componen un documento (por ejemplo: introducción, capítulos y secciones) y sus relaciones.
- estructura física, relativa a la forma en que se presenta un documento, incluyendo la tipografía y la ordenación espacial de sus componentes, entre otros.

Utilizando lenguajes de marcado genéricos la información relativa a dichas estructuras puede, en documentos electrónicos, almacenarse de manera independiente. Un ejemplo de ellos es SGML que, considerado un metalenguaje, permite definir lenguajes de marcado para la representación y transmisión de documentos en un formato adecuado.

Los lenguajes de marcado HTML (fundamento de la Web) y XML, basados en la especificación SGML, han devenido en estándares para la presentación e intercambio de información en Internet (FRESNO, 2006).

1.2.1. Lenguaje HTML

Publicar y distribuir información globalmente requiere el uso de un lenguaje universalmente entendido, algo así como una lengua franca de publicación que todos los ordenadores puedan comprender. El lenguaje de publicación usado por la Web es HTML (W3C, 1999).

El lenguaje HTML fue diseñado para estructurar documentos y presentarlos en forma de hipertexto, estableciendo relaciones unidireccionales entre ellos (hipervínculos). Inicialmente fue concebido para visualizar e interconectar el contenido de documentos electrónicos, considerando por ello un pequeño conjunto de etiquetas. Posteriormente, la ventaja que representaba la simplicidad de HTML se convirtió en un inconveniente, ya que su marcado no siempre cubría todos los aspectos de presentación que los usuarios requerían. La solución adoptada fue el desarrollo de extensiones privadas del lenguaje, lo que complicó su estandarización. Las luchas comerciales entre las principales empresas de desarrollo de navegadores web condujeron a un lenguaje HTML que, aunque universalmente utilizado,

carece de estandarización real. Desde finales de 1993 existen comités de estandarización, impulsados por el World Wide Web Consortium[20], que tienden a un modelo único de HTML (FRESNO, 2006).

La versión 4 de HTML desarrolla el lenguaje con mecanismos para hojas de estilo, marcos, objetos incluidos, soporte mejorado para texto de derecha a izquierda y mejoras en formularios, entre otros. El HTML 4.01, revisión de la versión anterior, corrige errores e introduce algunos cambios (W3C, 1999).

1.2.1.1. Vocabulario HTML

En el contexto de este trabajo, el vocabulario HTML puede definirse como el conjunto de etiquetas que un documento, codificado en tal lenguaje, puede contener. Dicho vocabulario se manifiesta en la estructura global de un documento HTML, determinada por tres partes o secciones principales (W3C, 1999):

- sección 1, contiene información sobre la versión del HTML utilizado.
- sección 2, cabecera declarativa.
- sección 3, cuerpo que posee el contenido principal del documento.

Seguidamente se muestra, como ejemplo sencillo, un documento HTML codificado según la estructura anterior:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<HTML>
 <HEAD>
   <TITLE>Ejemplo sencillo de un documento HTML</TITLE>
 </HEAD>
 <BODY>
   <H1>Conocer</H1>
   <H2>el</H2>
```

20 http://www.w3c.org

```
   <H3>Vocabulario HTML</H3>
   <H4>es vital para<H4>
   <H5>tareas de</H5>
   <H6>Categorizaci&oacute;n Automatizada de Documentos HTML</H6>
 </BODY>
</HTML>
```

En la sección 1 se especifica la Definición del Tipo de Documento (en lo adelante DTD). A continuación se mencionan los DTD que HTML 4.01 soporta (W3C, 1999):

- DTD HTML 4.01 Estricto (*Strict* DTD)[21], incluye todos los elementos y atributos que no han sido desaprobados o que no aparecen en documentos con marcos.
- DTD HTML 4.01 Transicional (*Transitional* DTD)[22], abarca todo lo incluido por el DTD Estricto más los elementos y atributos desaprobados (la mayoría de los cuales están relacionados con la presentación visual).
- DTD HTML 4.01 para Documentos con Marcos (*Frameset* DTD)[23], abarca todo lo incluido por el DTD Transicional más los marcos que permiten dividir una página en varias zonas.

En la sección 2 se incluyen definiciones globales a todo el documento, se puede agregar un fondo de pantalla y definir los colores del texto, entre otros. El texto contenido en su etiqueta (HEAD) no resultará visible en un navegador web. Estas definiciones pueden estar relacionadas con el formato global del documento, para lo que se emplea la etiqueta STYLE, o tratarse de características más cercanas a la visualización que su autor desea dar a cada etiqueta y que podrían diferir de las que establezca por defecto el navegador (FRESNO, 2006). En esta sección también pueden incluirse, mediante la etiqueta SCRIPT, códigos escritos en diferentes lenguajes interpretados (JavaScript, PHP y ASP, entre otros).

La sección 3 está formada por etiquetas relativas a la estructura y a cómo debe visualizarse la información contenida en el documento HTML. Dentro de su etiqueta (BODY) se incluye el

21 http://www.w3.org/TR/1999/REC-html401-19991224/sgml/dtd.html

22 http://www.w3.org/TR/1999/REC-html401-19991224/sgml/loosedtd.html

23 http://www.w3.org/TR/1999/REC-html401-19991224/sgml/framesetdtd.html

texto que se desea visualizar en la página web. Además, en esta sección pueden utilizarse diferentes encabezados (etiquetas H1, H2, H3, H4, H5 y H6) que permiten hacer una ordenación jerárquica de los apartados en los que se quiere estructurar un documento (FRESNO, 2006). Finalmente, notar que esta sección y la 2 deben estar delimitadas o anidadas por la etiqueta HTML.

Además de las etiquetas mencionadas, y entre otras, aparecen las siguientes:

- TITLE, delimita el título del documento.
- I, muestra el texto con estilo itálica.
- B, muestra el texto con estilo negrita.
- BIG, muestra el texto con una fuente grande.
- EM, indica énfasis.
- STRONG, indica un énfasis más fuerte.
- CITE, contiene una cita o una referencia a otras fuentes.
- DFN, definición.
- BLOCKQUOTE, cita larga.
- U, muestra el texto subrayado.

En este trabajo tales etiquetas, junto a las de encabezados, serán consideradas para representar los documentos HTML, puesto que generalmente los autores de páginas web las utilizan para denotar mayor importancia o significado en los términos que delimitan.

1.3. Filtrado de Contenidos

El primer intento de un gobierno por controlar los contenidos nocivos para los menores tuvo lugar en Estados Unidos, con la famosa Ley de Decencia de las Comunicaciones. Después de un intenso debate social, en junio de 1997, la Corte Suprema anuló dicha ley debido a que: no estaba redactada con precisión y existían alternativas menos restrictivas e igualmente eficaces (el uso de *software* de filtrado). Tras el fracaso de dicha ley, la Unión Europea tomó la misma orientación del Gobierno de Clinton: promover la llamada autorregulación, cuya

base sería el *software* de filtrado (VILLATE, 2001). A partir de ese momento se sucedieron iniciativas y planes para incentivar el Filtrado de Contenidos.

Inicialmente el ámbito del Filtrado de Contenidos era la Web, con el tiempo se ha expandido a otros servicios de Internet (Correo Electrónico y Mensajería Instantánea, entre otros) que también necesitaron reducir la exposición de sus usuarios a contenidos inadecuados.

Dentro de las herramientas tecnológicas de protección, los filtros de contenidos son muy comunes (MIRANDA, 2005). Entre las funcionalidades de estos se mencionan:

- establecen distintos perfiles de usuario y políticas de uso de Internet.
- limitan el tiempo de navegación.
- registran los intentos de acceso a contenidos inadecuados.
- generan reportes estadísticos sobre el uso de Internet por los usuarios.
- permiten, a usuarios afectados, reportar posibles regulaciones incorrectas.

En la actualidad existen varios filtros comerciales, tales como Net Nanny[24], CYBERsitter[25] y CyberPatrol[26], entre otros (TOPTENREVIEWS, 2008). Además, navegadores web como Internet Explorer y Netscape permiten el Filtrado de Contenidos de manera gratuita, aunque ofrecen menos prestaciones que las herramientas comerciales. El despliegue de filtros de contenidos está en ascenso. Incluso en la fuerte economía de hoy, los negocios reconocen el potencial del filtrado para reducir la pérdida de productividad de sus empleados. Por otra parte, instituciones educacionales desean cumplir la responsabilidad de proporcionar un acceso seguro a Internet para sus estudiantes, y los padres quieren estar seguros de que sus hijos no están expuestos a contenidos inadecuados en el hogar (PURESIGHT, 2004).

El Filtrado de Contenidos se introdujo en la UCI a finales del año 2004, cuando fue solicitado a dicho centro por la Oficina de Seguridad para las Redes Informáticas (OSRI). A partir de ese momento se inició el desarrollo del proyecto, denominado entonces, Filtrado de paquetes por contenidos. Dicho proyecto tiene como objetivo principal: desarrollar una

24 http://www.netnanny.com

25 http://www.cybersitter.com

26 http://www.cyberpatrol.com

herramienta informática (nombrada Filpacon) que permita regular (permitir o denegar) el acceso de usuarios a contenidos inadecuados de Internet. Actualmente la versión 1.0.0 de Filpacon está orientada a servidores, o sea, brinda su servicio a grandes redes de usuarios con acceso a Internet.

1.3.1. Métodos de Filtrado

Existen dos métodos principales para filtrar contenidos de la Web, uno es mediante BDUC y el otro a través del Análisis Dinámico de Contenidos (PURESIGHT, 2004). Comúnmente, el primer método se materializa del siguiente modo:

- el usuario solicita una página web a través del filtro.
- el filtro busca en su BDUC la URL solicitada.
- si la URL no existe en la BDUC, dependiendo de las opciones de configuración del filtro, puede ser permitida o denegada.
- si la URL existe en la BDUC, dependiendo de la naturaleza de su categoría asociada, puede ser permitida o denegada.

En la actualidad este es el enfoque utilizado por Filpacon, aunque depende de DMOZ, URLBlackList.com y Shalla para crear y actualizar su BDUC. Generalmente las listas de URLs categorizadas, que están disponibles en Internet, son creadas a partir del esfuerzo colaborativo de varias personas que difícilmente pueden tratar con el dinamismo de la Web. La labor de estos seres humanos es: manualmente revisar y asignar cada URL a una categoría específica de contenido; lo cual, evidentemente, es un proceso consumidor de tiempo, propenso a errores humanos y altamente dependiente de sus costumbres, conocimientos y compromisos con la labor de categorización. Por otra parte, la mayoría de estas listas no pueden ser utilizadas para fines comerciales, a partir de las condiciones de uso que imponen (licencias). Claramente no existe un enfoque exacto para recolectar y categorizar documentos de la Web, provocando que algunos proveedores de filtros desarrollen mejores metodologías (generalmente privativas) que otros (PURESIGHT, 2004).

Por su parte, el Análisis Dinámico de Contenidos se efectúa en el instante que el usuario solicita una URL determinada. Diferentes técnicas para el Análisis Dinámico de Contenidos incluyen: análisis de textos según sus contextos, motores de inferencia y redes neuronales, entre otros. Cuando una página web es solicitada a través de un filtro que use este método, entonces será analizada y categorizada según la naturaleza de su contenido; seguidamente, a partir de la categoría identificada para la URL, el filtro determina si denegarla o simplemente registrar la actividad (PURESIGHT, 2004).

Aunque el Análisis Dinámico de Contenidos se plantea interesante y tentativo (provee mayor cobertura de la Web para los filtros), a corto plazo, no se incluye en los planes de desarrollo para Filpacon. De cualquier manera, ambos métodos requieren de la asignación automatizada de categorías temáticas al contenido textual de documentos HTML, proceso que este trabajo se propone automatizar.

1.4. Categorización Automatizada de Textos

La Categorización Automatizada de Textos ha tenido un interesante auge en los últimos 10 años, debido a la creciente disponibilidad de documentos en formato digital y la consiguiente necesidad de organizarlos. El enfoque dominante para este problema se basa en técnicas de Aprendizaje Automático: un proceso general inductivo que construye automáticamente un categorizador mediante el aprendizaje de características de categorías, a partir de un conjunto de documentos precategorizados. Las ventajas de este método sobre el enfoque de Ingeniería del Conocimiento (la definición manual de un categorizador a partir de expertos de dominios) son: muy buena efectividad (comparable con la conseguida por expertos humanos), el ahorro considerable en términos de la exigente labor del experto y la sencilla portabilidad a dominios diferentes (SEBASTIANI, 2002).

La categorización de textos (ver la Figura 1) es la tarea de asignar valores *booleanos* (0 o 1) a cada par $\langle d_j, c_i \rangle$ que pertenece a $D \times C$, donde D es un dominio de documentos y $C = \{c_1, c_2, .., c_{|c|}\}$ es un conjunto de categorías predefinidas. El valor 1 asignado a $\langle d_j, c_i \rangle$ indica la decisión de archivar el documento d_j bajo la categoría c_i, mientras que el valor 0 indica la decisión de no archivar el documento d_j bajo la categoría c_i. Más formalmente, la tarea es

aproximar la *función objetivo* `Ω : D x C -> {1, 0}` (que describe cómo los documentos deben ser categorizados) a través de una función `ð : D x C -> {1, 0}` llamada el *categorizador*, tal que `Ω` y `ð` coincidan tanto como sea posible (SEBASTIANI, 2002).

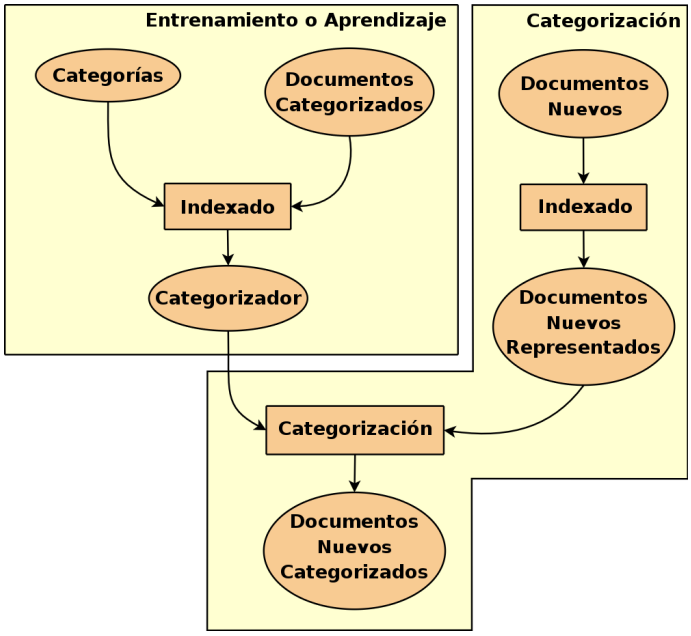


Figura 1: Esquema General de la Categorización de Textos.

Según (VENEGAS, 2007), la Categorización Automatizada de Textos puede concebirse como un proceso de aprendizaje, donde un algoritmo capta las características que distinguen cada categoría de las demás, es decir, aquellas que deben poseer los documentos nuevos para pertenecer a esta. Tales características no indican de forma absoluta (determinísticamente) la pertenencia a una categoría, sino que lo hacen en función de una escala o graduación (coeficiente). Este coeficiente lo que expresa en realidad es el grado de confianza o certeza de que el documento en cuestión pertenezca a la categoría asociada. Incluso, cuando dos expertos humanos deciden categorizar un documento pueden estar en desacuerdo, algo que ocurre con mucha frecuencia.

En términos más sencillos, materializar la Categorización Automatizada de Textos consiste en descubrir variables que sean útiles para discriminar textos que pertenecen a categorías existentes. Por tanto, el objetivo de un categorizador (programa que ejecuta algoritmos de categorización) es indicar la categoría que debe asociarse a cada texto, partiendo de un esquema o modelo de categorización previamente creado (FIGUEROLA *et al.*, 2000).

Con respecto al número de categorías que pueden ser asignadas a cada documento, se tienen dos casos: uno donde exactamente una categoría puede ser asignada a cada documento (mono-etiqueta, en inglés *single-label*) y otro donde puede asignarse cualquier cantidad de categorías (multi-etiqueta, en inglés *multilabel*) (SEBASTIANI, 2002).

La Categorización Automatizada de Textos posee varias aplicaciones, incluyendo: filtrado de documentos, creación y mantenimiento de directorios web temáticos a partir de la Categorización Automatizada de Documentos HTML (caso peculiar del primero), desambiguación del sentido de las palabras y, en general, cualquier tarea que requiera la organización y selección de grandes cantidades de documentos.

1.4.1. Representación de Textos

Representar adecuadamente los documentos es fundamental para determinados procesamientos automatizados, especialmente la categorización de textos. En líneas generales, la representación deberá ser fiel al contenido del documento, incluyendo la información necesaria para extraer el conocimiento útil que se espera obtener y, a la vez, deberá ser adecuada a las especificaciones de los algoritmos que se empleen a continuación (FRESNO, 2006).

Evidentemente los documentos no pueden ser interpretados directamente por un categorizador o por el algoritmo que lo construye. Por tanto, un proceso de representación (indexación) necesitará ser aplicado uniformemente, tanto a los documentos que serán utilizados para enseñar o entrenar al categorizador como a los que serán categorizados (SEBASTIANI, 2002). La representación de textos se materializa a partir de un modelo de

representación de documentos que puede definirse, según (FRESNO, 2006), a partir de los siguientes términos:

- Un *conjunto de objetos*, que constituye el vocabulario empleado en la representación.
- Un *álgebra*, que generalice las relaciones entre los objetos.
- Una *función de medida*, que establece la distancia (o similitud) entre objetos dentro del espacio medible.
- Una *función de proyección* o *ponderación*, aplicable sobre el *conjunto de objetos* y que indique la relevancia de los objetos en la representación.

Independientemente del modelo que se emplee, casi todos consideran la palabra como elemento fundamental. Por tanto, una representación será un conjunto de palabras que, de una forma u otra, represente el contenido del documento.

Uno de los métodos empleados para tratar de mejorar los modelos de representación de documentos es el empleo de procesos de análisis sintácticos para identificar sintagmas y usarlos como objetos dentro del modelo de representación. Las ventajas que ofrece el uso de sintagmas es que suelen resultar menos ambiguos que las palabras aisladas, sin embargo sus propiedades estadísticas son menores (FRESNO, 2006). En este trabajo se considerarán palabras aisladas para representar los documentos.

1.4.1.1. Modelos Vectoriales

Los modelos vectoriales se incluyen en el conjunto de técnicas que han sido empleadas para representar documentos, y descansan sobre la premisa de que el significado de un documento puede derivarse del conjunto de rasgos presentes en el mismo; dichos rasgos serán, de un modo u otro, los vectores generadores de un espacio vectorial. Los documentos se modelan como conjuntos de rasgos que pueden ser individualmente tratados y pesados. De este modo, en tareas de categorización de textos, los documentos pasan a ser representados como vectores dentro de un espacio euclídeo, de forma que midiendo la distancia entre dos vectores se trata de estimar su similitud como indicador de cercanía semántica (FRESNO, 2006).

En (LUHN, 1957) se propone el uso de un *espacio de conceptos* para atenuar los problemas de categorización derivados de suponer uniformidad en el vocabulario. Cada documento podría representarse como una lista de elementos correspondientes a ideas independientes del lenguaje. De este modo, dos documentos que compartieran conceptos similares tendrían, presumiblemente, una similitud semántica y estarían representados en posiciones cercanas dentro del *espacio de conceptos*. Por ejemplo, documentos que contienen las palabras *estudiantes*, *software*, *universidad*, *ciencias* e *informáticas* con mayor probabilidad podrían pertenecer a la categoría *UCI*. Dentro de los modelos vectoriales se mencionan: el Modelo de Espacio Vectorial (SALTON, WONG y YANG, 1975) y el Índice de Latencia Semántica (LANDAUER, FOLTZ y LAHAM, 1998).

El Modelo de Espacio Vectorial es un modelo matemático que ha sido empleado para la representación de textos en varios sistemas de Recuperación de Información y Categorización Automatizada de Textos. Asume el principio de independencia, considerando por ello que las cadenas aparecidas en un mismo texto no tienen relación entre sí, permitiendo que sean cuantificadas individualmente. De este modo, la semántica de un documento queda reducida a la suma de los significados de los rasgos que contiene. La representación de un documento, en este modelo, se genera a partir de un conjunto de rasgos previamente definidos, que forman un vocabulario y que constituyen los vectores base del espacio vectorial. Con la excepción de algunos sistemas de Recuperación de Información, las representaciones suelen ser vectores con componentes *booleanas* o numéricas (LEWIS, 1990). Finalmente, todo documento dentro del Modelo de Espacio Vectorial quedará representado como una combinación lineal de vectores base, donde cada coeficiente de la combinación representará la relevancia de cada rasgo en el contenido del documento, calculada con una Función de Ponderación (FRESNO, 2006).

En este trabajo se utilizará el Modelo de Espacio Vectorial, ya que reduce drásticamente la complejidad computacional del problema (representación de documentos) y proporciona satisfactorios resultados en tareas de categorización (SEBASTIANI, 2002).

1.4.1.2. Funciones de Ponderación

Las funciones de ponderación se utilizan para calcular la importancia o relevancia de un rasgo (palabra) en el contenido de un texto. Tratan de cuantificar, estadísticamente, cuanto aporta cada rasgo al significado del documento. En dichas funciones pueden distinguirse las de carácter local[27] y global[28] (FRESNO, 2006).

Las funciones de ponderación local más conocidas son:

- Binaria.
- Frecuencia del Término.
- Frecuencia Normalizada.
- Frecuencia Aumentada y Normalizada.
- Pesado Logarítmico.

Por su parte, las funciones de ponderación global más conocidas son:

- Frecuencia Inversa del Documento.
- Frecuencia del Término x Frecuencia Inversa del Documento.
- Frecuencia Inversa Ponderada.
- Frecuencia Inversa Probabilística.
- Normal.
- Frecuencia Global x Frecuencia Inversa del Documento.
- Entropía.

Cuando una función local se aplica a un documento HTML para representarlo se obtiene, necesariamente, una representación autocontenida; enfoque que será utilizado en este trabajo para representar tales documentos.

27 funciones de ponderación local: toman únicamente información del propio documento para obtener una representación.
28 funciones de ponderación global: utilizan información de la colección para generar una representación.

1.4.1.3. Selección del Vocabulario

La Selección del Vocabulario incluye aquellas fases que transforman un texto en el conjunto de rasgos que lo podrán representar. Cuando la representación se basa en palabras individuales, las fases más comunes son las siguientes:

- **Análisis Léxico**.
  Fase donde se analiza un texto para distinguir las cadenas que formarán parte de la representación. Para encontrar tales cadenas se tienen en cuenta, por ejemplo, delimitadores como espacios en blanco, signos de puntuación, guiones y otros signos especiales.
- **Lematización**.
  El proceso de lematización es aquel donde a cada forma flexiva se le asigna su lema. De este modo, formas diferentes como *tendrán* o *tuvieran*, compartirían el lema *tener* y serían considerados como un mismo rasgo dentro del vocabulario. La idea de este proceso es: diferentes palabras construidas sobre un mismo lexema suelen representar un mismo concepto. Existen algoritmos de lematización para diferentes idiomas: inglés (PORTER, 1980), castellano (FIGUEROLA *et al.*, 2002) y holandés (KRAAIJ, 2002), entre otros.
- **Eliminación de Palabras Vacías**.
  En este proceso se eliminan palabras que se emplean para articular el discurso, tales como artículos, preposiciones y conjunciones, entre otros; es decir, aquellas que no tienen por sí solas una semántica relevante en el contenido de un texto. La comunidad científica dispone de listas de palabras vacías para numerosos idiomas, entre las que se incluyen también algunos verbos, adverbios o adjetivos de uso frecuente.

Las fases mencionadas anteriormente serán aplicadas, en ASADOHTML, para seleccionar el vocabulario durante la representación de los documentos de la Colección de Entrenamiento y los nuevos que serán categorizados.

1.4.2. Algoritmos de Aprendizaje Automático

El Aprendizaje Automático puede definirse como la disciplina que permite desarrollar sistemas capaces de realizar tareas de modo automático y de forma que el desempeño de dichas tareas resulte mejor con experiencia que sin ella. Posee dos enfoques principales: (a) aprender para poder generar un nuevo conocimiento y (b) aprender para tratar de mejorar el comportamiento de un sistema (ALPAYDIN, 2004). Para (a) se suelen utilizar técnicas basadas en razonamiento inductivo, mientras que (b) suele estar relacionado con la utilización de técnicas analíticas. Además, dentro del Aprendizaje Automático se pueden distinguir el Supervisado y el No Supervisado.

El Aprendizaje Supervisado se construye sobre un conocimiento previo. Por tal motivo, en la Categorización Automatizada de Textos se debe disponer de documentos de referencia (Colección de Entrenamiento) para cada una de las categorías en las que se quiere categorizar. Después de una etapa de entrenamiento el sistema queda ajustado, de modo que ante nuevos documentos el algoritmo es capaz de categorizarlos en alguna de las categorías existentes. Cuanto mayor sea el conjunto de documentos precategorizados, mayor será la información potencial disponible y, previsiblemente, mejor resultará el aprendizaje. Por su parte, los sistemas de Aprendizaje No Supervisado son aquellos que no disponen de conocimiento previo, es decir, de una Colección de Entrenamiento.

El aprendizaje es la etapa en la que se obtiene la información para cada una de las categorías en las que será posible categorizar un documento determinado. La obtención de las características de categorías es el resultado de la etapa de entrenamiento. Este entrenamiento se realiza sobre una jerarquía temática que puede estar estructurada en uno o varios niveles de generalidad. A partir de un número de documentos que resulte representativo de cada categoría, es posible encontrar las características de esta (YANG, 1999). Una vez que se hayan encontrado las características de cada categoría, es factible abordar la etapa de categorización.

Un Algoritmo de Aprendizaje Automático se define como aquel proceso capaz de elegir una función única a partir de una Colección de Entrenamiento, dando respuesta al problema

planteado de aprendizaje. La mayor parte de estos algoritmos no son específicos a la categorización de documentos, sino que sirven para categorizar cualquier tipo de objeto. Entre los más utilizados se mencionan:

- **Algoritmo Naïve Bayes**.
  Se basan en la teoría de probabilidad de Bayes. El teorema de Bayes permite estimar la probabilidad de un suceso a partir de la probabilidad de que ocurra otro, del cual depende el primero. Este algoritmo es el más conocido dentro de los probabilísticos (FRESNO, 2006).
- **Algoritmo de Rocchio**.
  Este algoritmo proporciona un mecanismo para construir vectores representativos (patrones) de cada una de las categorías de documentos que se consideren. De este modo, en la fase de entrenamiento se parte de una Colección de Entrenamiento precategorizada manualmente y se construyen patrones para cada una de las categorías, considerando como ejemplos positivos los documentos de entrenamiento de esa categoría y como ejemplos negativos los del resto. Para categorizar un nuevo documento, bastará con encontrar la distancia entre la representación de los documentos y cada uno de los patrones. Aquel patrón que presente mayor similitud indicará la categoría a la que se debe asignar el documento. En (FIGUEROLA *et al.*, 2000) se puede consultar un trabajo donde se trata y aplica este algoritmo.
- **Algoritmo de los Vecinos más Cercanos y variantes**.
  Este algoritmo se basa en la aplicación de una métrica que establezca la similitud entre un documento que se quiere categorizar y cada uno de los documentos de la Colección de Entrenamiento. La categoría que se asigna al documento será la del documento más cercano según la métrica establecida. Una de las variantes más conocidas de este algoritmo es la de los *k*-vecinos más cercanos, que consiste en tomar los *k* documentos más parecidos en lugar de sólo el primero. Como los documentos más cercanos pueden pertenecer a categorías diferentes, se asignará aquella que más veces haya aparecido (BERMEJO, 2000).
- **Algoritmos basados en redes neuronales**.
  Las redes neuronales han sido aplicadas a problemas de categorización de documentos en numerosas ocasiones. Es posible entrenar una red neuronal para que dada una entrada determinada (un vector de representación) produzca una salida

deseada (la categoría a la que corresponde el documento). Existen muchos tipos de redes neuronales, con topologías y características bien diferenciadas (FRESNO, 2006).

- **Algoritmo Máquinas de Soporte Vectorial** (en lo adelante SVM).
  Se trata de un método de aprendizaje automático presentado en (CORTES y VAPNIK, 1995) y (VAPNIK, 2000). Este algoritmo pretende encontrar una hipersuperficie de separación entre categorías dentro del espacio de representación (ver la Figura 2), de modo que una banda, o margen, lo más gruesa posible esté vacía alrededor de una región de separación entre categorías (CRISTIANINI y SHAWE-TAYLOR, 2000). Las SVM resultan ser sistemas de aprendizaje universal que, en su forma más simple, son capaces de encontrar una función lineal discriminante que separe el espacio de representación en regiones correspondientes a cada una de las categorías consideradas. Como en todo Aprendizaje Supervisado, la entrada al sistema es un conjunto de ejemplos de entrenamiento. Si el conjunto de entrenamiento es linealmente separable, el algoritmo SVM encuentra un hiperplano que maximiza la distancia euclídea a los ejemplos de entrenamiento más cercanos. En caso de que el conjunto no pueda ser linealmente separable, se mide el error en el entrenamiento. Así, el cálculo de este hiperplano se reduce a un problema de optimización de errores en el que se aplican restricciones (VAPNIK, 2000). Las restricciones fuerzan a que todos los ejemplos de entrenamiento sean precategorizados correctamente por encima de un error mínimo. Una característica fundamental de estos categorizadores es que son independientes de la dimensión del espacio de características. Así, la media de complejidad no depende del número de rasgos, sino de cómo puedan separarse los datos. Esto significa que siempre que los datos sean separables, podría generalizarse la presencia de muchos rasgos en una determinada categoría.

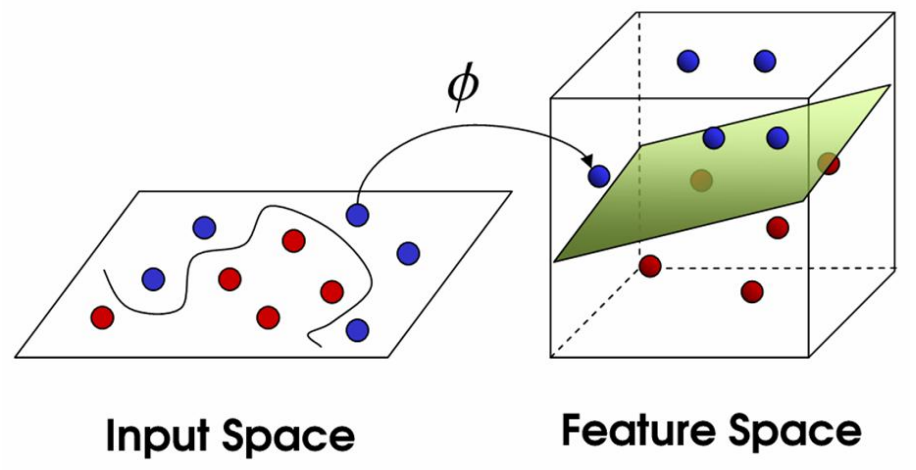


Figura 2: Principio de las SVM.

En (JOACHIMS, 1997) se proporciona evidencia, tanto teórica como empírica, que indica lo adecuado de usar SVM para la categorización de textos. Por tal evidencia, será el Algoritmo de Aprendizaje Automático a utilizar en ASADOHTML.

1.5. Categorización Automatizada de Documentos HTML

Muchas técnicas de la Categorización Automatizada de Textos han sido aplicadas a la Categorización Automatizada de Documentos HTML, implicando que esta última sea un caso peculiar de la primera. Por tanto, las teorías y definiciones presentadas en las secciones anteriores se importan a este apartado.

Desde la creación del lenguaje HTML se han propuesto diferentes modelos para la representación de documentos codificados en tal lenguaje. Estos modelos se centraron inicialmente en el análisis del contenido textual del propio documento HTML para, posteriormente, pasar a analizar en profundidad la estructura de hipervínculos que forma el conjunto de hipertextos contenidos en la Web. Además, también han sido representados en función del uso que se hace de ellos (FRESNO, 2006). Partiendo de estas ideas, los modelos de representación de documentos HTML se agrupan principalmente bajo tres enfoques,

correspondientes a cada una de las partes en las que puede dividirse la Minería Web (KOSALA y BLOCKEEL, 2000):

- Minería de contenido o **representaciones por contenido**.
  Su objetivo es la recogida de información e identificación de patrones relativos a los contenidos de las páginas web y a las búsquedas que se realizan sobre las mismas. Su carácter es esencialmente local.
- Minería de estructura o **representaciones por contexto**.
  En este caso se analiza la estructura de un sitio web o de una colección de páginas web a través de los datos relativos a su conectividad. Se analiza, por tanto, la topología del grafo de hipertexto. El análisis es de carácter global y considera tanto los *in-links*[29] como los *out-links*[30] presentes en un documento HTML.
- Minería de uso o **representaciones por uso**.
  En esta categoría se analizan principalmente los registros de acceso almacenados en los servidores web o las *cookies*[31] aceptadas por un usuario. El ámbito de este tipo de representaciones puede ser local o global, y su finalidad es encontrar patrones de uso asociados a un determinado sitio web, analizando las páginas más visitadas y los recorridos más habituales, entre otros.

En particular, las representaciones de carácter local tienen en cuenta únicamente la información presente en un documento HTML para representarlo, por tanto podrían considerarse como representaciones autocontenidas. Este tipo de representaciones resultará completamente independiente del tamaño actual y futuro de la Web, así como de su estructura, ya que no se tendrá que analizar otro documento para representar uno dado. Este trabajo se centrará en este tipo específico de representación de documentos HTML.

La Categorización Automatizada de Documentos HTML puede resultar muy útil para tratar, en el contexto del Filtrado de Contenidos, el exponencial ritmo de cambio y crecimiento que

29 in-links: conjunto de hipervínculos que apuntan hacia una página web.

30 out-links: conjunto de hipervínculos que salen de una página web.

31 cookies: información que se guarda en el disco duro del cliente, a petición del servidor de la página que se esté visitando, para luego poder ser recuperada por el propio servidor en posteriores visitas.

experimenta la Web. Además, el desarrollo de las taxonomías propias de los directorios web puede ser asistido por esta técnica, tanto en su creación como en su mantenimiento.

1.6. Herramientas para Construir Categorizadores

Los categorizadores automatizados de documentos HTML se construyen utilizando, entre otras, las siguientes herramientas informáticas:

- **Sistema Operativo Debian GNU/Linux**[32].
  Generalmente, todo *software* moderno requiere de un Sistema Operativo como plataforma para sustentar su desarrollo. Debian es libre y viene con más de 18.773 paquetes, distribuidos de forma gratuita. Su última versión estable es la 4.0, publicada el 17 de febrero de 2008. Es una de las distribuciones de GNU/Linux más utilizadas en el mundo.
- **Entorno de Desarrollo Integrado Anjuta**[33].
  La codificación o implementación de un *software* puede efectuarse en un Entorno de Desarrollo Integrado, que se ejecuta sobre un Sistema Operativo como, por ejemplo, Debian. Anjuta es un proyecto libre para C y C++, aunque también soporta otros lenguajes de programación como Perl y Python; ofrece muchas características avanzadas de programación e incluye un administrador de proyectos, asistentes, plantillas, depurador interactivo y un poderoso editor que verifica y resalta la sintaxis escrita. Su última versión estable es la 2.4.1, publicada el 10 de marzo de 2008.
- **Lenguaje de Programación Perl**[34].
  Perl es un lenguaje de propósito general, originalmente desarrollado para el procesamiento de textos y ahora utilizado para varias tareas, incluyendo administración de sistemas, desarrollo web y programación en red, entre otras. Es *software* libre y está disponible para la mayoría de los sistemas operativos como, por ejemplo, Debian. Se previó que fuera práctico (facilidad de uso, eficiente y completo) en lugar de hermoso (pequeño, elegante y mínimo). El 26 de octubre de 1995 se creó

32 http://www.debian.org

33 http://www.anjuta.org

34 http://www.perl.org/about.html

la CPAN[35]: colección de sitios web que almacenan y distribuyen fuentes en Perl, binarios, documentación, *scripts* y módulos. Características importantes y algunas construcciones esenciales han sido añadidas a Perl, incluyendo un soporte importante para la Programación Orientada a Objetos. Su última versión estable es la 5.10.0, publicada el 18 de diciembre de 2007.

- **Módulos Perl**.
  Perl posee mecanismos para usar bibliotecas de código externas, permitiendo que un simple fichero contenga rutinas o funciones comunes a varios programas; tales ficheros se conocen como módulos. Actualmente existen más de 13.561 módulos disponibles en la CPAN, aportados por más de 6.618 autores, para una amplia variedad de tareas, incluyendo matemáticas avanzadas, conectividad de bases de datos y conexión de redes, entre otras. Además, todo lo publicado en la CPAN está disponible de forma libre, estandarizada y documentada. Muchos de estos módulos son utilizados para la Inteligencia Artificial y el Procesamiento de Lenguaje Natural, tales como HTML::TreeBuilder[36], Lingua::Stem::Snowball[37] y Algorithm::SVM[38], entre otros.

En principio, para construir categorizadores automatizados de documentos HTML podrá utilizarse, prácticamente, cualquier herramienta destinada al desarrollo de *software*. No obstante, para estar a tono con el desarrollo y evolución actual de la Informática, ha de optarse por aquellas que faciliten su correcto desarrollo. Por tal motivo se utilizarán, entre otras, las herramientas antes mencionadas para concebir el sistema ASADOHTML.

1.7. Proceso Unificado de Rational

El desarrollo de RUP, como producto, ha sido influenciado de muchas fuentes y sigue un camino desde el Proceso Objectory (primera publicación en 1987) pasando por el Proceso Objectory de Rational (publicado en 1997) hasta el propio RUP (publicado en 1998). Más que un simple proceso de desarrollo de *software*, es un marco de trabajo genérico que puede

35 http://www.cpan.org

36 http://search.cpan.org/~petek/HTML-Tree-3.23/lib/HTML/TreeBuilder.pm

37 http://search.cpan.org/~creamyg/Lingua-Stem-Snowball-0.941/lib/Lingua/Stem/Snowball.pm

38 http://search.cpan.org/~lairdm/Algorithm-SVM-0.13/lib/Algorithm/SVM.pm

especializarse para diferentes áreas de aplicación, tipos de organizaciones y tamaños de proyectos. Además, utiliza el Lenguaje Unificado de Modelado (en lo adelante UML) para visualizar, especificar, construir y documentar un *software*. Los verdaderos aspectos definitorios de RUP se resumen en tres frases claves que lo hacen único: iterativo e incremental, dirigido por casos de uso y centrado en la arquitectura (JACOBSON, BOOCH y RUMBAUGH, 2000).

RUP se repite a lo largo de una serie de ciclos que constituyen la vida de un sistema. Cada ciclo concluye con una versión del producto para los clientes y consta de cuatro fases: Inicio, Elaboración, Construcción y Transición. A su vez, cada fase se subdivide en iteraciones que pasan por los principales flujos de trabajo: Modelado del Negocio, Requisitos, Análisis y Diseño, Implementación, Pruebas, Despliegue, Configuración y Control de Cambios, Gestión del Proyecto y Entorno (ver la Figura 3).

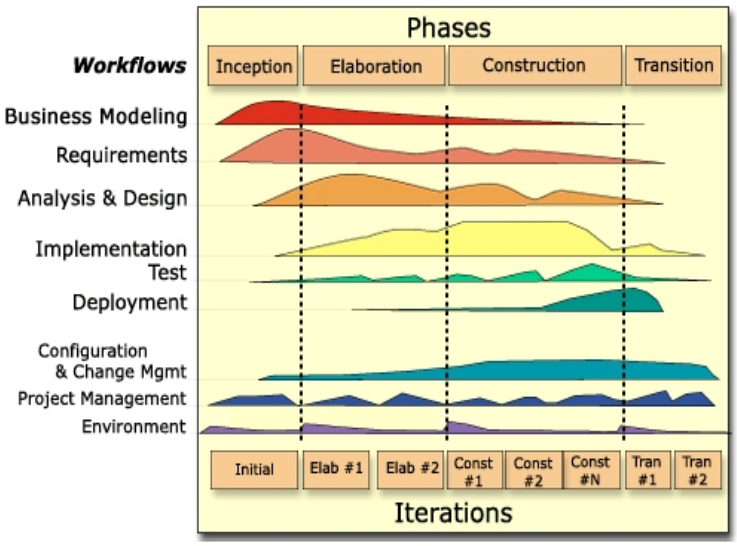


Figura 3: Fases, Iteraciones y Flujos de Trabajo de RUP.

El Modelo del Dominio pertenece al Flujo de Trabajo Modelado del Negocio y describe los conceptos más importantes del contexto como objetos del dominio, enlazándolos unos con otros. La identificación y asignación de un nombre para tales objetos ayuda a desarrollar un

Diccionario de Clases que mejora la comunicación entre los que trabajan en el sistema. Posteriormente, los objetos del dominio ayudarán a identificar algunas clases a medida que se analiza y diseña el sistema. La descripción del Modelo del Dominio se efectúa mediante diagramas de UML, específicamente usando diagramas de clases (JACOBSON, BOOCH y RUMBAUGH, 2000).

El propósito fundamental del Flujo de Trabajo Requisitos es: guiar el desarrollo hacia el sistema correcto mediante una descripción de sus requisitos. Partiendo de que cada proyecto de *software* es diferente, existen varios puntos de partida para la captura de requisitos. En ocasiones, se comienza haciendo un Modelo del Negocio. En otros casos, el *software* es un sistema empotrado que no da soporte directamente al negocio; entonces se puede tener como entrada un sencillo Modelo de Objetos, tal como un Modelo del Dominio. Aún cuando existen diferentes puntos de partida, generalmente, los siguientes pasos son factibles: enumerar los requisitos candidatos, comprender el contexto del sistema y capturar los Requisitos Funcionales y No Funcionales. Los trabajadores que participan en este Flujo de Trabajo son: el analista de sistemas, el especificador de casos de uso, el diseñador de interfaz de usuario y el arquitecto. Además, los artefactos que se generan son: el Modelo de Casos de Uso (que incluye los casos de uso y los actores), la Descripción de la Arquitectura, el Glosario y el Prototipo de Interfaz de Usuario. Tales artefactos constituyen un buen punto de partida para los flujos de trabajo Análisis y Diseño, Implementación y Pruebas (JACOBSON, BOOCH y RUMBAUGH, 2000).

Durante el Flujo de Trabajo Análisis se analizan los requisitos que se describieron en la captura de requisitos, refinándolos y estructurándolos. Los trabajadores de este Flujo de Trabajo son: el arquitecto, el ingeniero de casos de uso y el ingeniero de componentes. Además, los artefactos que se generan son: el Modelo del Análisis, las Clases del Análisis, la Realización de Caso de Uso-Análisis, el Paquete del Análisis y la Descripción de la Arquitectura (Vista del Modelo del Análisis). El Modelo del Análisis, principal resultado de este flujo de trabajo, se considera la entrada fundamental para las actividades de diseño subsiguientes (JACOBSON, BOOCH y RUMBAUGH, 2000).

Por su parte, durante el Flujo de Trabajo Diseño se modela el sistema y se encuentra su forma para que soporte todos los requisitos (incluyendo los Requisitos No Funcionales y

otras restricciones) que se le atribuyen. Los trabajadores de este flujo de trabajo son: el arquitecto, el ingeniero de casos de uso y el ingeniero de componentes. Los artefactos que se generan son: el Modelo del Diseño, las Clases del Diseño, la Realización de Caso de Uso-Diseño, el Subsistema de Diseño, la Interfaz, el Modelo de Despliegue y la Descripción de la Arquitectura (Vista del Modelo del Diseño y Vista del Modelo de Despliegue). El Modelo del Diseño, principal resultado de este flujo de trabajo, se esfuerza en conservar la estructura del sistema impuesta por el Modelo del Análisis y sirve como esquema para la implementación (JACOBSON, BOOCH y RUMBAUGH, 2000).

Teniendo en cuenta que RUP es la metodología estándar más usada, y adoptada por la UCI para el desarrollo de sistemas orientados a objetos será, por tanto, la utilizada en este trabajo. Para ello deberá configurarse tal metodología, de manera que se adecue a los requerimientos necesarios para efectuar satisfactoriamente el Análisis y Diseño de ASADOHTML.

1.8. Investigaciones Similares

Se consideran investigaciones similares aquellas que fueron materializadas para resolver la esencia del problema que origina este trabajo; es decir, las que se plantearon la necesidad de automatizar la asignación de categorías temáticas al contenido textual de documentos HTML, para contribuir al proceso de crear y actualizar la BDUC de un Filtro de Contenidos. En el ámbito internacional destacan, entre otros, los siguientes desarrollos:

- **PureSight**.
  Filtro de Contenidos que se adapta a la naturaleza dinámica de Internet. Su tecnología de filtrado dinámico es una tecnología sofisticada y privativa de Reconocimiento Artificial de Contenidos, que puede identificar el contenido de un sitio web y entonces decidir si permitirlo o no. Cada sitio web solicitado es inspeccionado por su algoritmo inteligente para asegurar su cumplimiento de las políticas de uso corporativas, institucionales o de grupos familiares. Provee una cobertura completa y confiable de la Web con una exactitud de reconocimiento incomparable. El Reconocimiento Artificial de Contenidos contiene un poderoso conjunto de algoritmos de Inteligencia

Artificial que analizan y categorizan datos en tiempo real. No obstante, por ser privativo, imposibilita el acceso a detalles de su implementación (PURESIGHT, 2008).

- **Optenet**[39].
  Filtro de Contenidos que, basado en técnicas de Inteligencia Artificial, reduce la dependencia de listas de URLs categorizadas y se adapta a la naturaleza dinámica de Internet. Su Analizador Inteligente de Contenidos proporciona un rápido análisis, con un tiempo de procesamiento de 0,1 milisegundos por página web solicitada. No obstante, por ser privativo, imposibilita el acceso a detalles de su implementación.
- **POESIA**[40].
  Filtro de Contenidos que se adapta a la naturaleza dinámica de Internet. Desarrolla un enfoque inteligente y efectivo para el filtrado de páginas web. Aún cuando es *software* libre y posibilita, por tanto, el acceso a detalles de su implementación, sus componentes desarrollados para la Categorización Automatizada de Textos no serán reutilizados en ASADOHTML por los siguientes motivos: su arquitectura (STARYNKEVITCH *et al.*, 2002) no es compatible con la prevista para MCADHTML y para categorizar una página web sólo utiliza el texto plano de la misma, ignorando la información proporcionada por las etiquetas HTML. No obstante podrán utilizarse, en ASADOHTML, muchas de las técnicas que implementa para el Procesamiento de Lenguaje Natural.

En el ámbito nacional (incluyendo a la UCI) no existen investigaciones similares o son desconocidas debido a que, en la actualidad, Filpacon es el único *software* de su tipo que se desarrolla en Cuba. No obstante, en el transcurso del tiempo y en diferentes ámbitos geográficos, se han desarrollado programas que categorizan documentos de manera automatizada, aunque fueron descartados por ser, entre otros motivos, privativos y/o específicos a tareas distintas del Filtrado de Contenidos.

39 http://www.optenet.com/es/index.asp

40 http://www.poesia-filter.org/poesiafilter.htm

1.9. Conclusiones

Después de analizar los inconvenientes de las principales investigaciones similares efectuadas en los ámbitos internacional, nacional y de la UCI, se evidencia la necesidad de concebir, analizar y diseñar un *software* de categorización propio. Para ello, en primera instancia, se debe tener en cuenta el dinamismo de la Web, su desmesurado tamaño actual, ritmo exponencial de crecimiento y la naturaleza de sus contenidos. Por otra parte, conociendo que HTML es el lenguaje de publicación usado por la Web, los documentos codificados en dicho lenguaje serán los procesados por ASADOHTML, específicamente el contenido textual y aprovechando, para sus representaciones, la valiosa información proporcionada por las etiquetas HTML. Una vez examinada la teoría inherente a la Categorización Automatizada de Textos y Documentos HTML se decide utilizar en ASADOHTML: palabras aisladas y el Modelo de Espacio Vectorial para representar los documentos; el Análisis Léxico, la Lematización y la Eliminación de Palabras Vacías como fases para seleccionar el vocabulario; funciones de ponderación local para lograr representaciones autocontenidas (que no dependen del tamaño actual y futuro de la Web) y las SVM como Algoritmo de Aprendizaje Automático. Tras abundar en aspectos de RUP, como la metodología más usada para el desarrollo de sistemas orientados a objetos, se considera apropiada para analizar y diseñar ASADOHTML.

Con la realización del presente capítulo: se definió la posición de los autores respecto a lo que se ha dicho o investigado sobre la Categorización Automatizada de Documentos HTML y se materializó la Fundamentación Teórica que servirá de sustento para tomar decisiones en los capítulos siguientes.

2. CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1. Introducción

Para guiar el desarrollo de ASADOHTML hacia el sistema correcto es necesario comprender su contexto e identificar las condiciones o capacidades que debe cumplir. En este capítulo, para definir las características del sistema: se realiza el modelado del dominio con el objetivo de comprender su contexto; se hace su propuesta, describiendo cómo debe funcionar y destacando sus características distintivas; se especifican sus Requisitos Funcionales, No Funcionales y Adicionales y se elabora el Modelo de Casos de Uso, definiendo sus actores, casos de uso y las relaciones entre ellos.

2.2. Descripción del Problema

Actualmente, para crear y actualizar la BDUC de Filpacon, se han creado programas que compendian listas de URLs categorizadas por DMOZ, URLBlackList.com y Shalla. Para evitar esta inconveniente dependencia, al Grupo API le asignaron el desarrollo de MCADHTML.

MCADHTML incluirá, entre otros, los módulos ASADOHTML y Administrador; este último se encargará de gestionar los procesos que en MCADHTML ocurran. La comunicación entre ambos módulos será bidireccional. Administrador le enviará a ASADOHTML el documento a categorizar, y este le devolverá el conjunto de categorías que considera más cercano, semánticamente, al documento HTML en cuestión.

En ASADOHTML (sistema que se necesita diseñar) un proceso general inductivo construirá un categorizador a partir de una Colección de Entrenamiento que estará formada por documentos HTML precategorizados. Posteriormente, para categorizar un documento HTML sólo será necesario utilizar el categorizador previamente construido.

Lo anterior es una descripción muy preliminar de necesidades, que sirve como punto de partida para identificar las clases del dominio y sus asociaciones.

2.3. Modelo del Dominio

ASADOHTML es un sistema que estará empotrado o incrustado dentro de otro (MCADHTML), implicando que no exista un negocio bien definido y, mucho menos, claridad en sus procesos. Lo anterior, unido a que MCADHTML está igualmente en desarrollo, ha propiciado que se haga un modelado del dominio en lugar del negocio. El objetivo del modelado del dominio es capturar, comprender y describir las clases más importantes dentro del contexto del sistema. El Modelo del Dominio, junto al Diccionario de Clases del Dominio, ayuda a los usuarios, clientes, desarrolladores y otros interesados a utilizar un vocabulario común. En la Figura 4 se muestra el Diagrama de Clases del Dominio.

Por su parte, el Diccionario de Clases del Dominio describe textualmente las clases identificadas durante el modelado del dominio del problema. Este diccionario sirve como un glosario de términos y se muestra a continuación:

- Categorizador: aprende de una Colección de Entrenamiento y se utiliza para categorizar documentos HTML.
- Colección de Entrenamiento: está formada por documentos HTML precategorizados y la utiliza un Categorizador para su aprendizaje.
- Documento HTML: archivo de texto, codificado en HTML, para crear una página web. Puede estar precategorizado en una Colección de Entrenamiento o asignársele, por medio de un Categorizador, un conjunto de categorías. Se representa como un Vector de Términos.
- Categoría: etiqueta simbólica que no aporta información adicional a los documentos HTML que agrupa, o al Vector de Términos que la representa.
- Vector de Términos: elemento de un espacio vectorial, formado por un conjunto de términos. Representa documentos HTML y categorías.

Una vez realizado el modelado del dominio se comprende el contexto de ASADOHTML, lo cual posibilita estar en condiciones de hacer su propuesta y ayuda a la determinación de los requisitos que se desprenden de tal contexto.

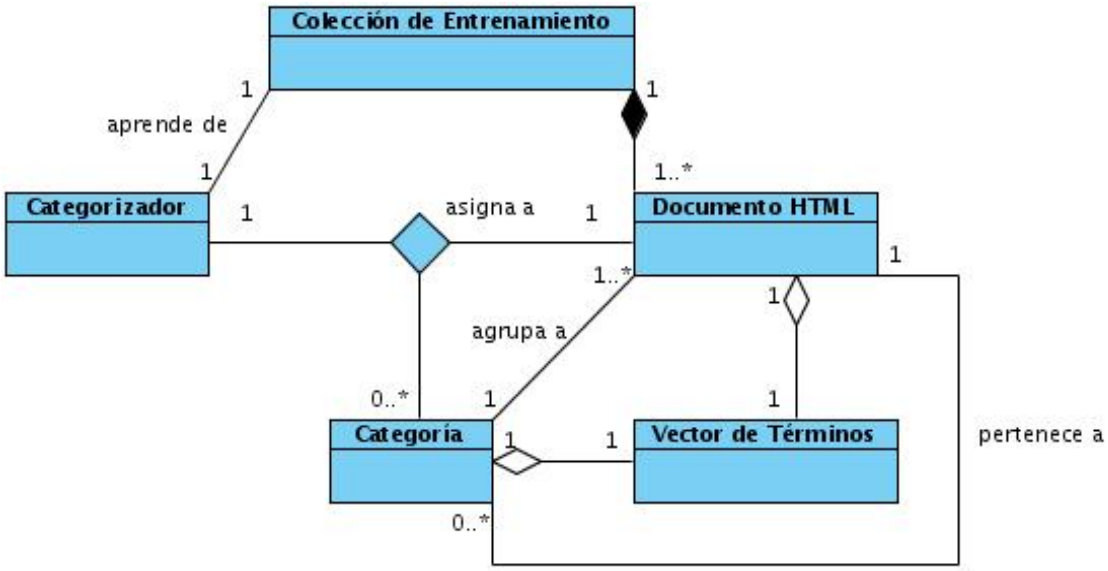


Figura 4: Diagrama de Clases del Dominio.

2.4. Propuesta

Se propone que ASADOHTML trabaje en dos modos de funcionamiento: conectado y no conectado. En el modo conectado estará interactuando con el módulo Administrador dentro de MCADHTML; mientras que en el modo no conectado una persona podrá ajustarlo, efectuando entrenamientos y categorizaciones de prueba, hasta lograr la configuración deseada. Además se sugiere que sea una aplicación de consola, pues en el modo conectado la interacción será ordenador-ordenador y en su complemento (modo no conectado) la misma no necesitará de una Interfaz Gráfica de Usuario por ser mínima y simple.

La Representación de Documentos HTML ha de efectuarse tanto en el entrenamiento como en la categorización. Dicha representación se compone de las siguientes tareas:

- **Selección del Vocabulario**.
  Incluye las fases de Análisis Léxico, Lematización y Eliminación de Palabras Vacías.
- **Captura de Información de Criterios**.

Encargada de la captura de los criterios heurísticos a combinar (FRESNO, 2006). Los criterios a tener en cuenta son: *frecuencia* de aparición de un rasgo en el texto, frecuencia de aparición de un rasgo en el *título* del documento, *posición* en la que aparece un rasgo dentro del texto y frecuencia de un rasgo en las partes *enfatizadas* del documento.

- **Ponderación de Términos**.
  Calcula la relevancia de un rasgo usando la Función de Ponderación ACC (FRESNO, 2006), que se expresa como:

$$ACC_{0,3/0,15/0,25/0,3}(t_i,d_j) = 0,3f_{frec}(t_i,d_j) + 0,15f_{tit}(t_i,d_j) + 0,25f_{enf}(t_i,d_j) + 0,3f_{pos}(t_i,d_j)$$

Además, el sistema debe guardar el modelo de categorización creado durante el entrenamiento y cargarlo antes de iniciar la categorización. En sentido general, ASADOHTML debe proporcionar un adecuado índice de aciertos y fallos (90% de efectividad como mínimo) en su labor de categorización. Labor que ha de consumarse en tiempos relativamente rápidos, considerando que tratará con grandes cantidades de documentos HTML.

El sistema propuesto, a diferencia de sistemas afines, estará orientado al Filtrado de Contenidos (tarea exigente y demandada), representará los documentos de manera autocontenida (independencia del tamaño actual y futuro de la Web), usará la información aportada por etiquetas HTML (acercamiento a la semántica real de un documento) y empleará las SVM (eliminan la necesidad de realizar la selección de características, haciendo la categorización mucho más fácil).

2.5. Requisitos Funcionales

Los Requisitos Funcionales son capacidades o condiciones que el sistema debe cumplir (JACOBSON, BOOCH y RUMBAUGH, 2000). A continuación se relacionan los Requisitos Funcionales de ASADOHTML:

- RF 1. Permitir representar un documento HTML.
  - RF 1.1. Seleccionar el vocabulario de un documento HTML.

  - RF 1.2. Capturar la información de criterios heurísticos a combinar.
  - RF 1.3. Ponderar los términos de un documento HTML.
- RF 2 Permitir efectuar el entrenamiento del categorizador.
  - RF 2.1. Ver RF 1.
  - RF 2.2. Crear el modelo de categorización.
  - RF 2.3. Guardar el modelo de categorización.
- RF 3 Permitir categorizar un documento HTML.
  - RF 3.1. Ver RF 1.
  - RF 3.2. Cargar el modelo de categorización.
  - RF 3.3. Asignar un conjunto de categorías temáticas a un documento HTML.

2.6. Requisitos No Funcionales

Los Requisitos No Funcionales especifican propiedades del sistema y, generalmente, se asocian a ciertos casos de uso (JACOBSON, BOOCH y RUMBAUGH, 2000). A continuación se relacionan los Requisitos No Funcionales de ASADOHTML:

- **RNF 1. Rendimiento**.
  Cuando se solicita la categorización de un documento HTML, el sistema debería efectuarla en menos de 0,5 segundos en el 90% de los casos (como mínimo).
- **RNF 2. Efectividad**.
  Cuando se categoriza un documento HTML, el sistema debería asignarle correctamente el conjunto de categorías en el 90% de los casos (como mínimo).
- **RNF 3. Interfaz**.
  Durante el modo conectado, la interacción deberá ser a través de *sockets* (implementando una arquitectura cliente/servidor) y condicionada por el formato que regirá la comunicación (por definir) entre módulos de MCADHTML. Por su parte en el modo no conectado, la interacción deberá estar condicionada por la sintaxis POSIX[41] para opciones de la línea de comandos.

41 http://search.cpan.org/author/JV/Getopt-Long-2.37/lib/Getopt/Long.pm

2.7. Requisitos Adicionales

Los Requisitos Adicionales son fundamentalmente Requisitos No Funcionales más genéricos y no pueden relacionarse con un caso de uso en concreto (JACOBSON, BOOCH y RUMBAUGH, 2000). A continuación se especifican los Requisitos Adicionales de ASADOHTML:

- **RA 1. Usabilidad**.
  El sistema debe ser fácil de usar, proporcionando una ayuda (desde la línea de comandos) que permita efectuar de manera rápida y satisfactoria todas sus funcionalidades.
- **RA 2. Soporte**.
  Se debe proporcionar mantenimiento al sistema, suministrando *parches* o nuevas versiones del mismo. Además, cuando se notifiquen problemas relacionados con su efectividad o necesidades específicas de categorización, se deben recepcionar y atender.
- **RA 3. Seguridad**.
  La información que manejará el sistema es sensible (pornografía infantil y terrorismo, entre otros) y a la postre tendrá valor comercial en Filpacon, por tanto debe ser protegida de accesos, divulgaciones o modificaciones no autorizadas. Además, el sistema debe estar disponible en todo momento para MCADHTML.
- **RA 4. Legal**.
  Para concebir el sistema, se deben usar únicamente herramientas con licencias libres.
- **RA 5. Confiabilidad**.
  El sistema debe seguir funcionando en la presencia de errores, tratándolos adecuadamente y registrando detalles que faciliten su corrección.
- **RA 6. Implementación**.
  Se debe usar la Guía de Estilo de Perl (perlstyle[42]) para que el código del sistema sea más fácil de leer, entender y mantener.
- **RA 7. *Hardware***.

42 http://perldoc.perl.org/perlstyle.html

El sistema debe funcionar en un ordenador para él dedicado. Además debe comunicarse con el módulo Administrador a través de la red, usando el protocolo TCP/IP.

2.8. Definición de los Actores

Un actor es una persona o sistema externo que juega un rol en una o más interacciones con el sistema. En la Tabla 1 se muestran los actores de ASADOHTML y una breve descripción de los mismos.

Actor	**Breve Descripción**
Comunicador	Sistema externo que actúa como intermediario entre el módulo Administrador y ASADOHTML. Solicita la categorización de un documento HTML o el entrenamiento del categorizador.
Ajustador	Persona que solicita la categorización de un documento HTML o el entrenamiento del categorizador.
Usuario	Actor que generaliza a Comunicador y Ajustador.

Tabla 1: Actores del Sistema.

2.9. Listado de Casos de Uso

Cada forma en que los actores usan el sistema se representa como un caso de uso; estos son fragmentos de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores (JACOBSON, BOOCH y RUMBAUGH, 2000). A continuación se listan los casos de uso de ASADOHTML:

CU-1	CU_Entrenar Categorizador
Actores	Usuario
Descripción	El CU se inicia cuando el Usuario solicita el entrenamiento del categorizador. Posteriormente se representan los documentos HTML de la Colección de Entrenamiento, se crea el modelo de categorización y se guarda para uso futuro.
Referencias	RF 2, RNF 3, CU-3

Tabla 2: Caso de Uso Entrenar Categorizador.

CU-2	CU_Categorizar Documento HTML
Actores	Usuario
Descripción	El CU se inicia cuando el Usuario solicita la categorización de un documento HTML. Se representa tal documento, se carga el modelo de categorización generado en el entrenamiento y se le asigna un conjunto de categorías temáticas.
Referencias	RF 3, RNF 1, RNF 2, RNF 3, CU-3

Tabla 3: Caso de Uso Categorizar Documento HTML.

CU-3 (incluido)	CU_Representar Documento HTML
Actores	Usuario
Descripción	El CU se inicia cuando el Usuario solicita el entrenamiento o la categorización de un documento HTML. Representa tal documento mediante la selección de su vocabulario, la captura de información de criterios y la ponderación de sus términos.
Referencias	RF 1, CU-1 (base), CU-2 (base), CU-4, CU-5, CU-6

Tabla 4: Caso de Uso Representar Documento HTML.

CU-4 (incluido)	CU_Seleccionar Vocabulario
Actores	Usuario
Descripción	El CU se inicia cuando se representa un documento HTML. Transforma el texto de tal documento en el conjunto de rasgos que lo podrán representar mediante su análisis léxico, la lematización de sus rasgos y la eliminación de palabras vacías.
Referencias	RF 1.1, CU-3 (base)

Tabla 5: Caso de Uso Seleccionar Vocabulario.

CU-5 (incluido)	CU_Capturar Información de Criterios
Actores	Usuario
Descripción	El CU se inicia cuando se representa un documento HTML. Captura la información de los criterios heurísticos a combinar: la frecuencia de aparición de un rasgo en su texto, la frecuencia de aparición de un rasgo en su título, la posición en la que aparece un rasgo dentro de su texto y la frecuencia de un rasgo en sus partes enfatizadas.
Referencias	RF 1.2, CU-3 (base)

Tabla 6: Caso de Uso Capturar Información de Criterios.

CU-6 (incluido)	CU_Ponderar Términos
Actores	Usuario
Descripción	El CU se inicia cuando se representa un documento HTML. Calcula la relevancia de sus rasgos usando la Función de Ponderación Local ACC.
Referencias	RF 1.3, CU-3 (base)

Tabla 7: Caso de Uso Ponderar Términos.

2.10. Diagrama de Casos de Uso

Un Diagrama de Casos de Uso es una representación gráfica de los actores y casos de uso del sistema, incluyendo sus interacciones. En la Figura 5 se muestra el Diagrama de Casos de Uso de ASADOHTML.

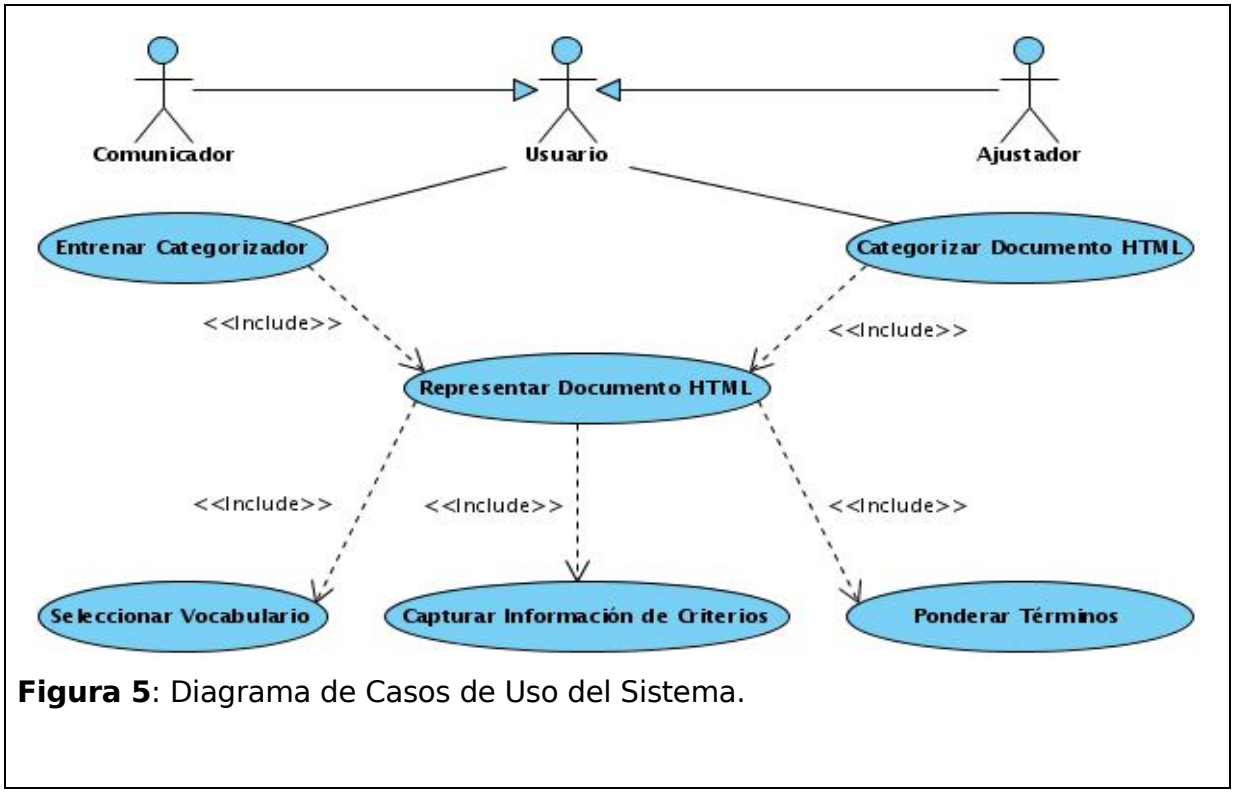


Figura 5: Diagrama de Casos de Uso del Sistema.

2.11. Casos de Uso Expandidos

A continuación se listan los Casos de Uso Expandidos de ASADOHTML:

CU-1	CU_Entrenar Categorizador
Propósito	Este CU tiene como objetivo entrenar el categorizador.
Actores	Usuario
Resumen	
El CU se inicia cuando el Usuario solicita el entrenamiento del categorizador. Posteriormente se representan los documentos HTML de la Colección de Entrenamiento, se crea el modelo de categorización y se guarda para uso futuro.	
Referencias	RF 2, RNF 3, CU-3
Acción del Actor	**Respuesta del Sistema**
1. El Usuario solicita entrenar el categorizador, proporcionando la localización de la Colección de Entrenamiento.	2. El sistema solicita la representación de los documentos HTML de la Colección de Entrenamiento.
	3. El sistema crea el modelo de categorización.
	4. El sistema guarda el modelo de categorización.
	5. El sistema envía al Usuario los resultados del entrenamiento.
Puntos de Extensión	
Línea 2: El sistema representa los documentos HTML. Ver CU-3.	
Flujos Alternativos	
Acción del Actor	**Respuesta del Sistema**
Acción 1: El Usuario proporciona una localización de la Colección de Entrenamiento incorrecta.	1.1. El sistema envía un mensaje al Usuario, informando que la localización de la Colección de Entrenamiento es incorrecta.
	1.2. Concluye el CU.

Tabla 8: Caso de Uso Expandido Entrenar Categorizador.

CU-2	CU_Categorizar Documento HTML
Propósito	Este CU tiene como objetivo categorizar un documento HTML.
Actores	Usuario
Resumen	
El CU se inicia cuando el Usuario solicita la categorización de un documento HTML. Se representa tal documento, se carga el modelo de categorización generado en el entrenamiento y se le asigna un conjunto de categorías temáticas.	
Referencias	RF 3, RNF 1, RNF 2, RNF 3, CU-3
Acción del Actor	**Respuesta del Sistema**
1. El Usuario solicita categorizar un documento HTML, proporcionando su localización o contenido textual.	2. El sistema solicita la representación del documento HTML.
	3. El sistema carga el modelo de categorización.
	4. El sistema asigna un conjunto de categorías temáticas al documento HTML.
	5. El sistema envía al Usuario los resultados de la categorización.
Puntos de Extensión	
Línea 2: El sistema representa el documento HTML. Ver CU-3.	
Flujos Alternativos	
Acción del Actor	**Respuesta del Sistema**
Acción 1: El Usuario proporciona una localización incorrecta del documento HTML o no provee su contenido textual.	1.1. El sistema envía un mensaje al Usuario, informando que la localización del documento HTML es incorrecta o no recibió su contenido textual.
	1.2. Concluye el CU.

Tabla 9: Caso de Uso Expandido Categorizar Documento HTML.

CU-3 (incluido)	CU_Representar Documento HTML
Propósito	Este CU tiene como objetivo representar un documento HTML.
Actores	Usuario
Resumen	
El CU se inicia cuando el Usuario solicita el entrenamiento o la categorización de un documento HTML. Representa tal documento mediante la selección de su vocabulario, la captura de información de criterios y la ponderación de sus términos.	
Referencias	RF 1, CU-1 (base), CU-2 (base), CU-4, CU-5, CU-6
Acción del Actor	**Respuesta del Sistema**
	1. El sistema solicita seleccionar el vocabulario del documento HTML.
	2. El sistema solicita capturar la información de criterios.
	3. El sistema solicita ponderar los términos del documento HTML.
Puntos de Extensión	
Línea 1: El sistema selecciona el vocabulario del documento HTML. Ver CU-4.	
Línea 2: El sistema captura la información de criterios. Ver CU-5.	
Línea 3: El sistema pondera los términos del documento HTML. Ver CU-6.	
Flujos Alternativos	
Acción del Actor	**Respuesta del Sistema**

Tabla 10: Caso de Uso Expandido Representar Documento HTML.

CU-4 (incluido)	CU_Seleccionar Vocabulario
Propósito	Este CU tiene como objetivo seleccionar el vocabulario de un documento HTML.
Actores	Usuario
Resumen	
El CU se inicia cuando se representa un documento HTML. Transforma el texto de tal documento en el conjunto de rasgos que lo podrán representar mediante su análisis léxico, la lematización de sus rasgos y la eliminación de palabras vacías.	
Referencias	RF 1.1, CU-3 (base)
Acción del Actor	**Respuesta del Sistema**
	1. El sistema analiza léxicamente el documento HTML.
	2. El sistema lematiza los rasgos del documento HTML.
	3. El sistema elimina las palabras vacías del documento HTML.
Puntos de Extensión	
Flujos Alternativos	
Acción del Actor	**Respuesta del Sistema**

Tabla 11: Caso de Uso Expandido Seleccionar Vocabulario.

CU-5 (incluido)	CU_Capturar Información de Criterios
Propósito	Este CU tiene como objetivo capturar la información de los criterios heurísticos a combinar.
Actores	Usuario
Resumen	
El CU se inicia cuando se representa un documento HTML. Captura la información de los criterios heurísticos a combinar: la frecuencia de aparición de un rasgo en su texto, la frecuencia de aparición de un rasgo en su título, la posición en la que aparece un rasgo dentro de su texto y la frecuencia de un rasgo en sus partes enfatizadas.	
Referencias	RF 1.2, CU-3 (base)
Acción del Actor	**Respuesta del Sistema**
	1. El sistema captura la frecuencia de aparición de los rasgos en el texto del documento HTML.
	2. El sistema captura la frecuencia de aparición de los rasgos en el título del documento HTML.
	3. El sistema captura la posición de los rasgos en el texto del documento HTML.
	4. El sistema captura la frecuencia de los rasgos en las partes enfatizadas del documento HTML.
Puntos de Extensión	
Flujos Alternativos	
Acción del Actor	**Respuesta del Sistema**

Tabla 12: Caso de Uso Expandido Capturar Información de Criterios.

CU-6 (incluido)	CU_Ponderar Términos
Propósito	Este CU tiene como objetivo ponderar los términos de un documento HTML.
Actores	Usuario
Resumen	
El CU se inicia cuando se representa un documento HTML. Calcula la relevancia de sus rasgos usando la Función de Ponderación Local ACC.	
Referencias	
Acción del Actor	**Respuesta del Sistema**
	1. El sistema calcula la relevancia de los términos de acuerdo a la frecuencia de aparición en el texto.
	2. El sistema calcula la relevancia de los términos de acuerdo a la frecuencia de aparición en el título.
	3. El sistema calcula la relevancia de los términos de acuerdo a la posición en el texto.
	4. El sistema calcula la relevancia de los términos de acuerdo a la frecuencia en las partes enfatizadas.
	5. El sistema pondera los términos con ACC, según sus relevancias previamente calculadas.
Puntos de Extensión	
Flujos Alternativos	
Acción del Actor	**Respuesta del Sistema**

Tabla 13: Caso de Uso Expandido Ponderar Términos.

2.12. Conclusiones

Después de definir las características de ASADOHTML se está en condiciones de efectuar su Análisis y Diseño a partir de las mismas. Además, dadas sus características distintivas, se concluye que el sistema propuesto podrá cumplir satisfactoriamente su labor dentro de MCADHTML.

3. CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

3.1. Introducción

El objetivo principal del Análisis y Diseño es transformar los requisitos a una especificación que describa cómo implementar el sistema. En este capítulo, de ASADOHTML, se presentan: sus Diagramas de Clases del Análisis, mostrando las clases participantes y relaciones entre ellas; sus Diagramas de Secuencia, evidenciando el orden de las acciones en los casos de uso cuando son invocados y, de sus Clases del Diseño, el diagrama y una descripción detallada. Además, se muestran los patrones utilizados para su Diseño y las técnicas que facilitarán documentarlo para su uso y tratar posibles errores en tiempo de ejecución.

3.2. Análisis

Durante el Análisis, los requisitos capturados en el capítulo anterior se especifican de manera más precisa y se estructuran de un modo que facilita su mantenimiento. Además se utiliza el lenguaje de los desarrolladores para describir los resultados, permitiendo razonar más sobre aspectos internos del sistema (JACOBSON, BOOCH y RUMBAUGH, 2000).

3.2.1. Diagramas de Clases del Análisis

En las figuras siguientes se presentan los Diagramas de Clases del Análisis de ASADOHTML:

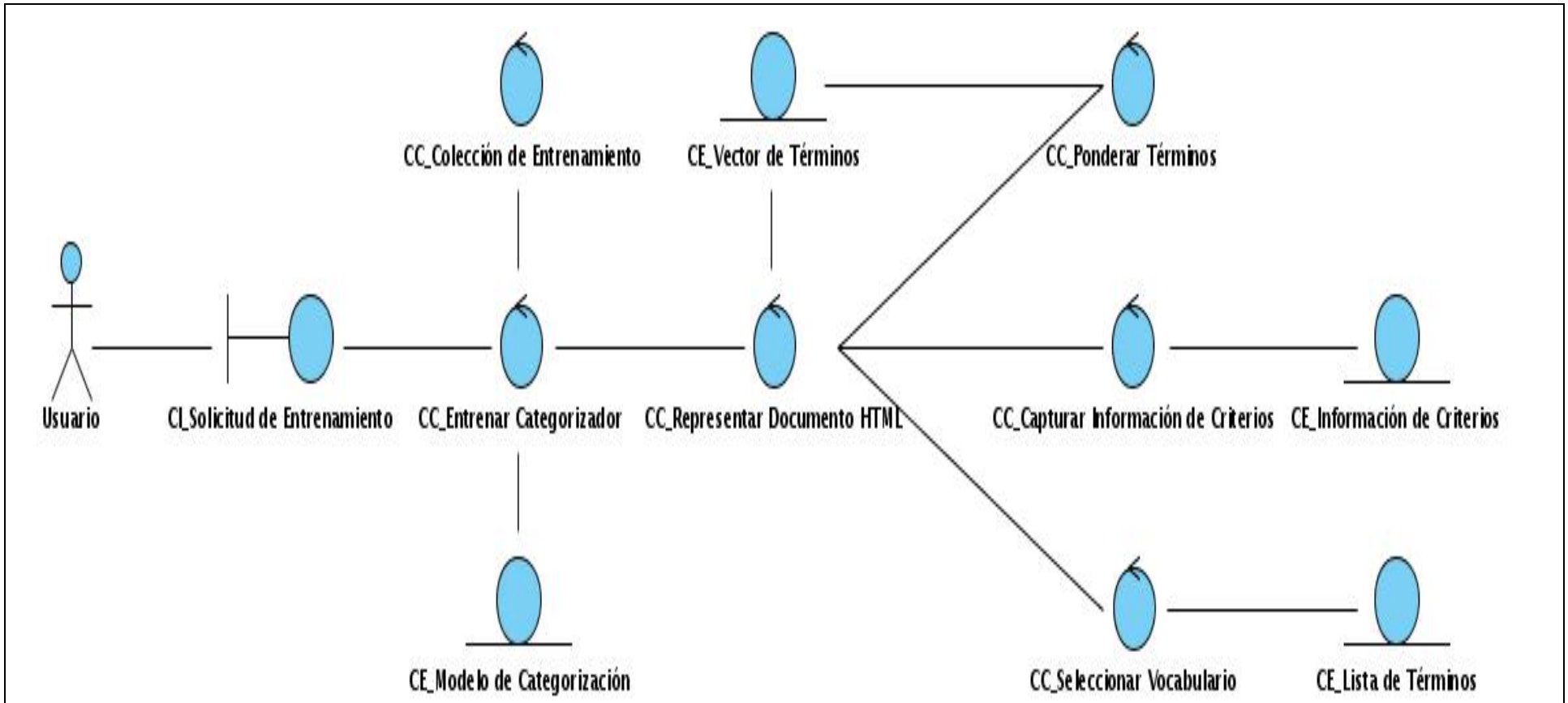


Figura 6: Diagrama de Clases del Análisis - CU_Entrenar Categorizador.

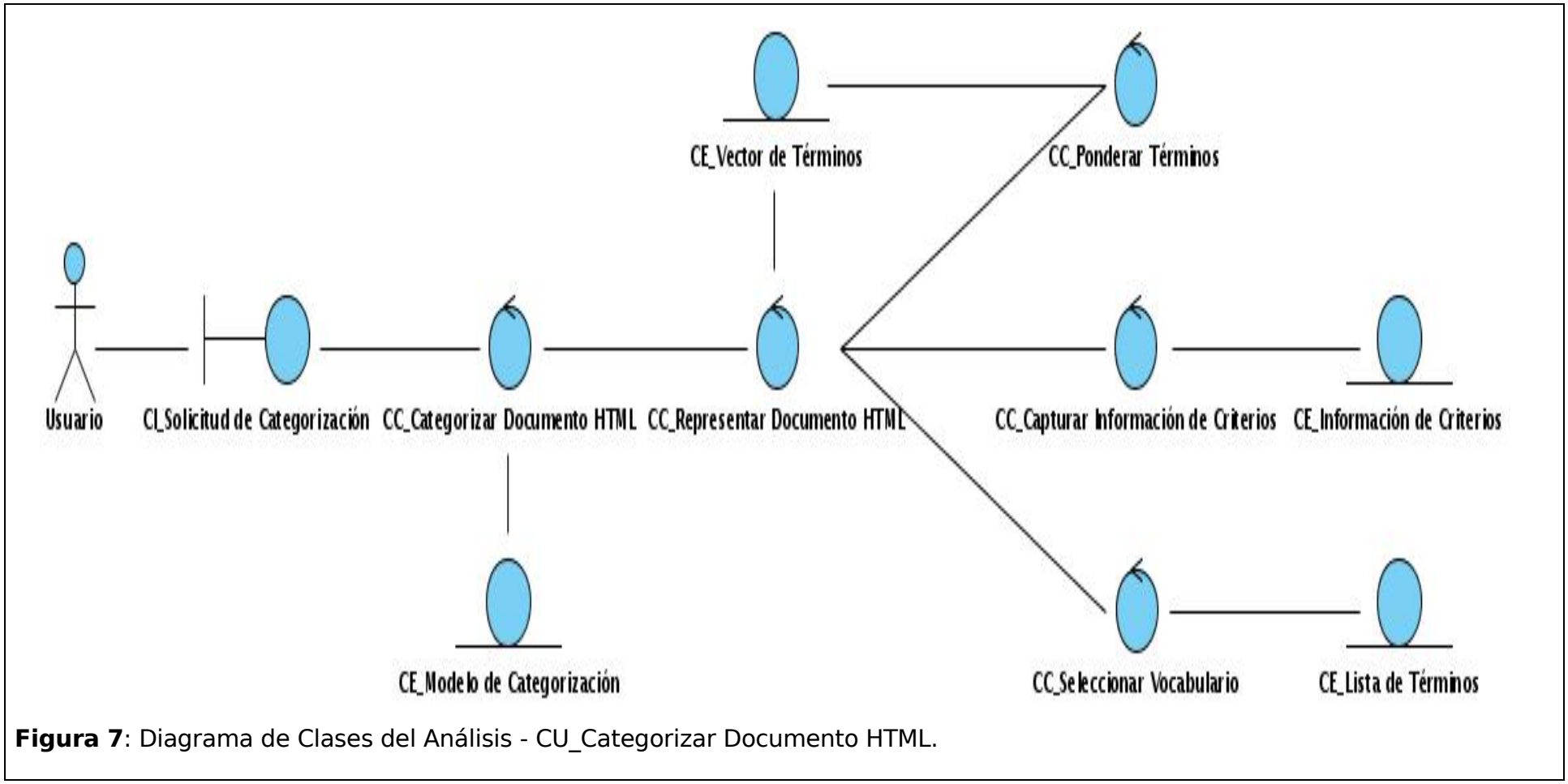


Figura 7: Diagrama de Clases del Análisis - CU_Categorizar Documento HTML.

3.3. Diseño

El Diseño es un refinamiento del Análisis que tiene en cuenta los Requisitos No Funcionales, es decir, indica cómo cumple el sistema sus objetivos. Debe ser suficiente para que el sistema sea implementado sin ambigüedades.

3.3.1. Patrones

Un patrón es una pareja de problema/solución representada por un nombre y es aplicable a varios contextos; además, viene acompañado de una sugerencia sobre la manera de usarlo en situaciones nuevas. Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos (LARMAN, 1999). A continuación se mencionan los patrones GRASP que serán utilizados para diseñar eficazmente el sistema ASADOHTML:

- **Experto**.
  Problema: ¿Cuál es el principio fundamental en virtud del cual se asignan las responsabilidades en el diseño orientado a objetos?
  Solución: Asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.
- **Creador**.
  Problema: ¿Quién sería responsable de crear una nueva instancia de alguna clase?
  Solución: Asignarle a la clase B la responsabilidad de crear una instancia de clase A en uno de los siguientes casos:
  - B *agrega* los objetos de A.
  - B *contiene* los objetos de A.
  - B *registra* las instancias de los objetos A.
  - B *utiliza* específicamente los objetos A.
  - B *tiene los datos de inicialización* que serán transmitidos a A cuando este objeto sea creado (así que B es un Experto respecto a la creación de A).
- **Bajo Acoplamiento**.
  Problema: ¿Cómo dar soporte a una dependencia escasa y a un aumento de la reutilización?

Solución: Asignar una responsabilidad para mantener bajo acoplamiento.

- **Alta Cohesión**.
  Problema: ¿Cómo mantener la complejidad dentro de los límites manejables?
  Solución: Asignar una responsabilidad de modo que la cohesión siga siendo alta.
- **Controlador**.
  Problema: ¿Quién debería encargarse de atender un evento del sistema?
  Solución: Asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente una de las siguientes opciones:
  - el sistema global (*controlador de fachada*).
  - la empresa u organización global (*controlador de fachada*).
  - algo en el mundo real que es activo (por ejemplo, el papel de una persona) y que pueda participar en la tarea (*controlador de tareas*).
  - un manejador artificial de todos los eventos del sistema de un caso de uso (*controlador de casos de uso*).

Además de los anteriores, será aplicado el patrón de creación **Singleton** (garantiza que una clase sólo tenga una instancia y proporciona un punto de acceso global a tal instancia).

3.3.2. Diagramas de Secuencia

Un Diagrama de Secuencia es una representación que muestra, en determinado escenario de un caso de uso, los eventos generados por actores externos, su orden y los eventos internos del sistema (LARMAN, 1999). En las figuras siguientes se presentan los Diagramas de Secuencia de ASADOHTML:

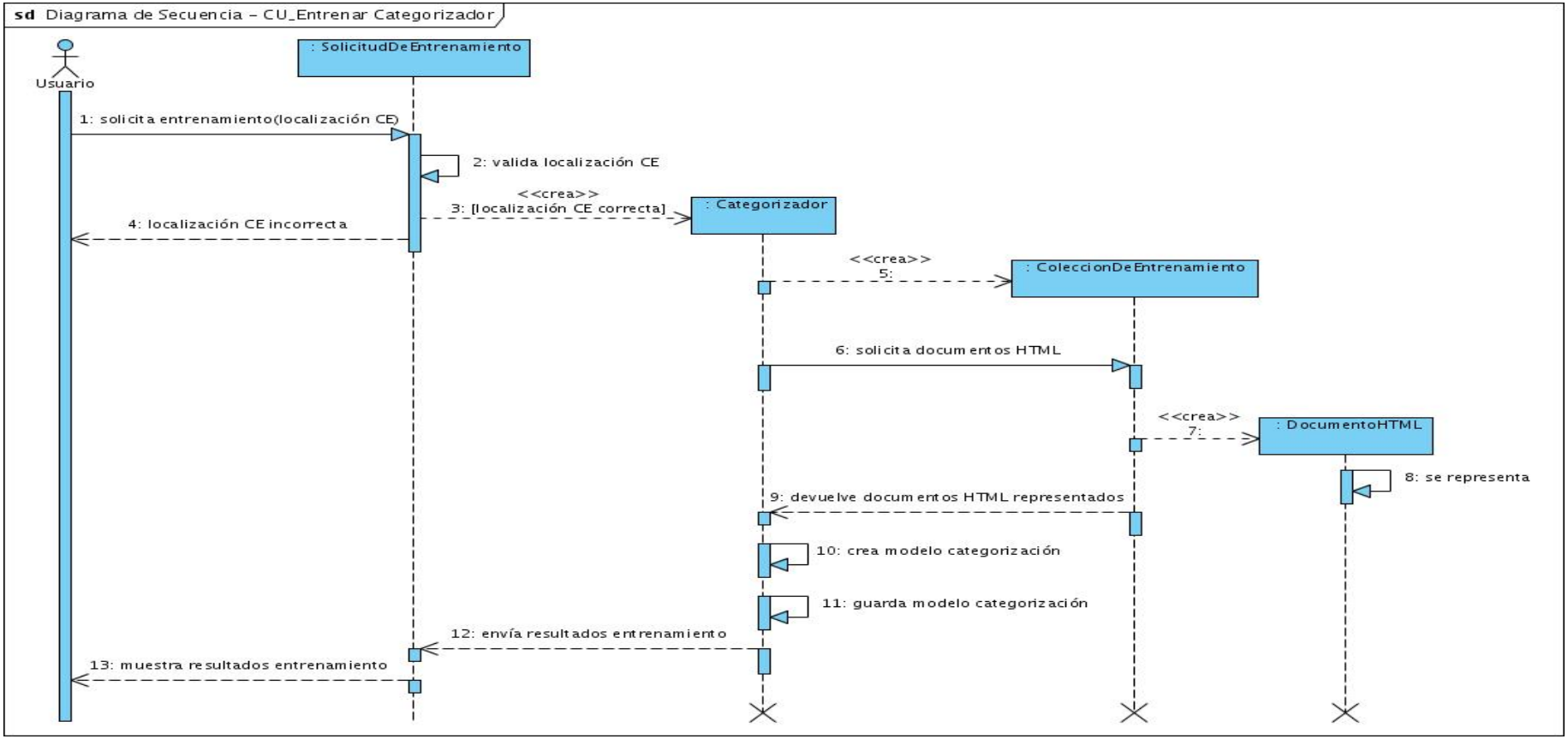


Figura 8: Diagrama de Secuencia - CU_Entrenar Categorizador.

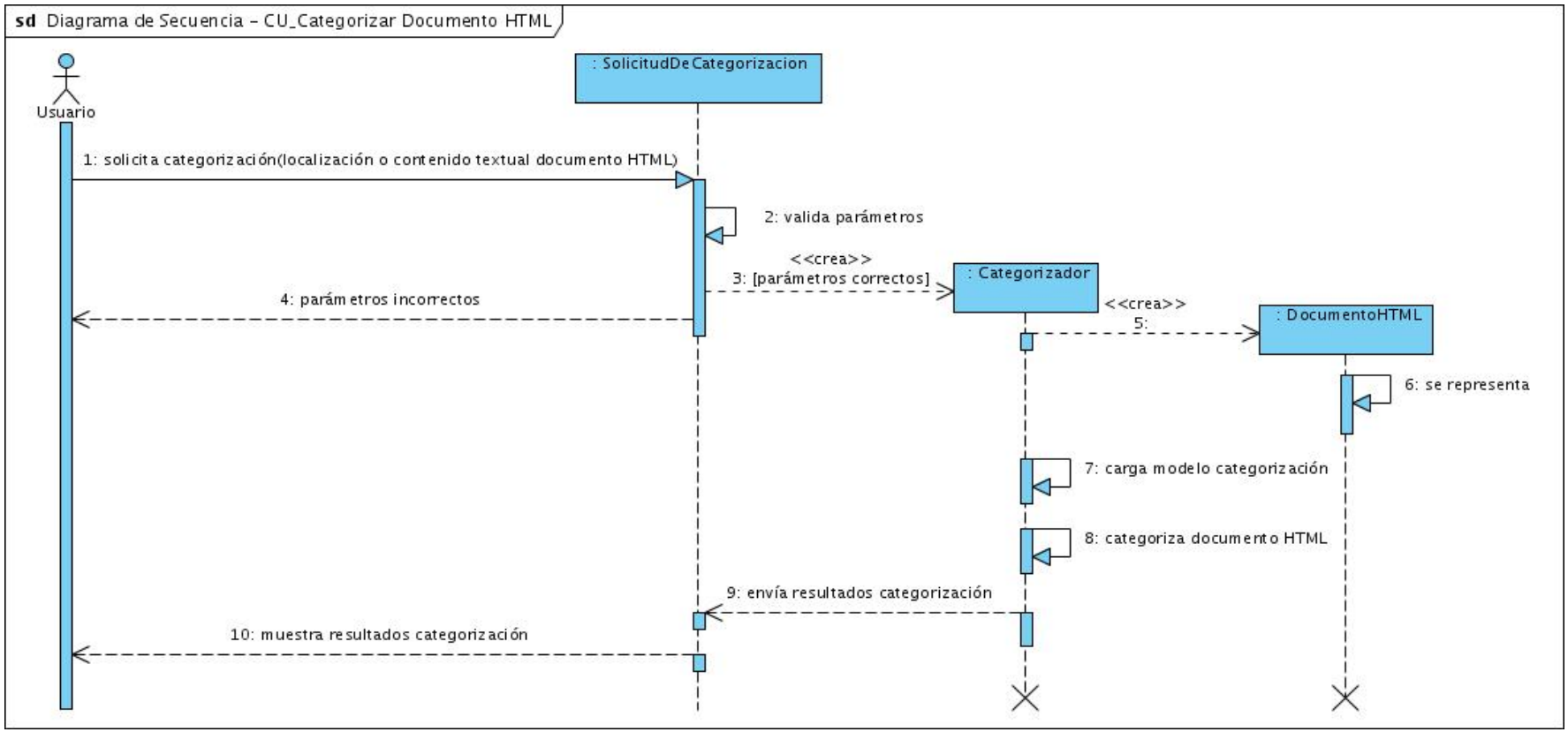


Figura 9: Diagrama de Secuencia - CU_Categorizar Documento HTML.

3.3.3. Diagrama de Clases del Diseño

Un Diagrama de Clases del Diseño describe gráficamente las especificaciones de las clases de un sistema y es una representación más concreta que los Diagramas de Clases del Análisis. Contiene información sobre clases, asociaciones, atributos y métodos. En las figuras siguientes se presenta el Diagrama de Clases del Diseño de ASADOHTML:

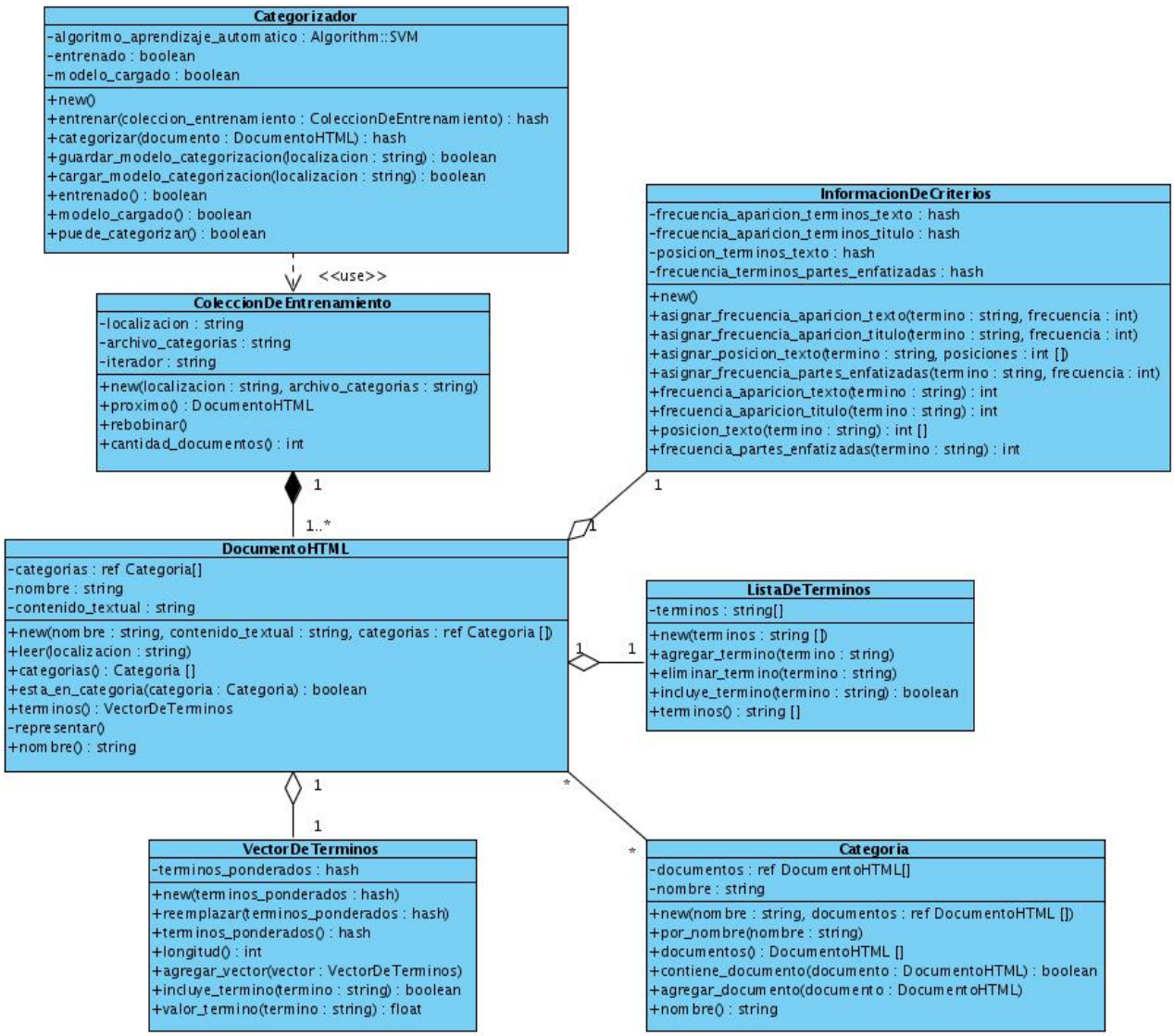


Figura 10: Diagrama de Clases del Diseño.

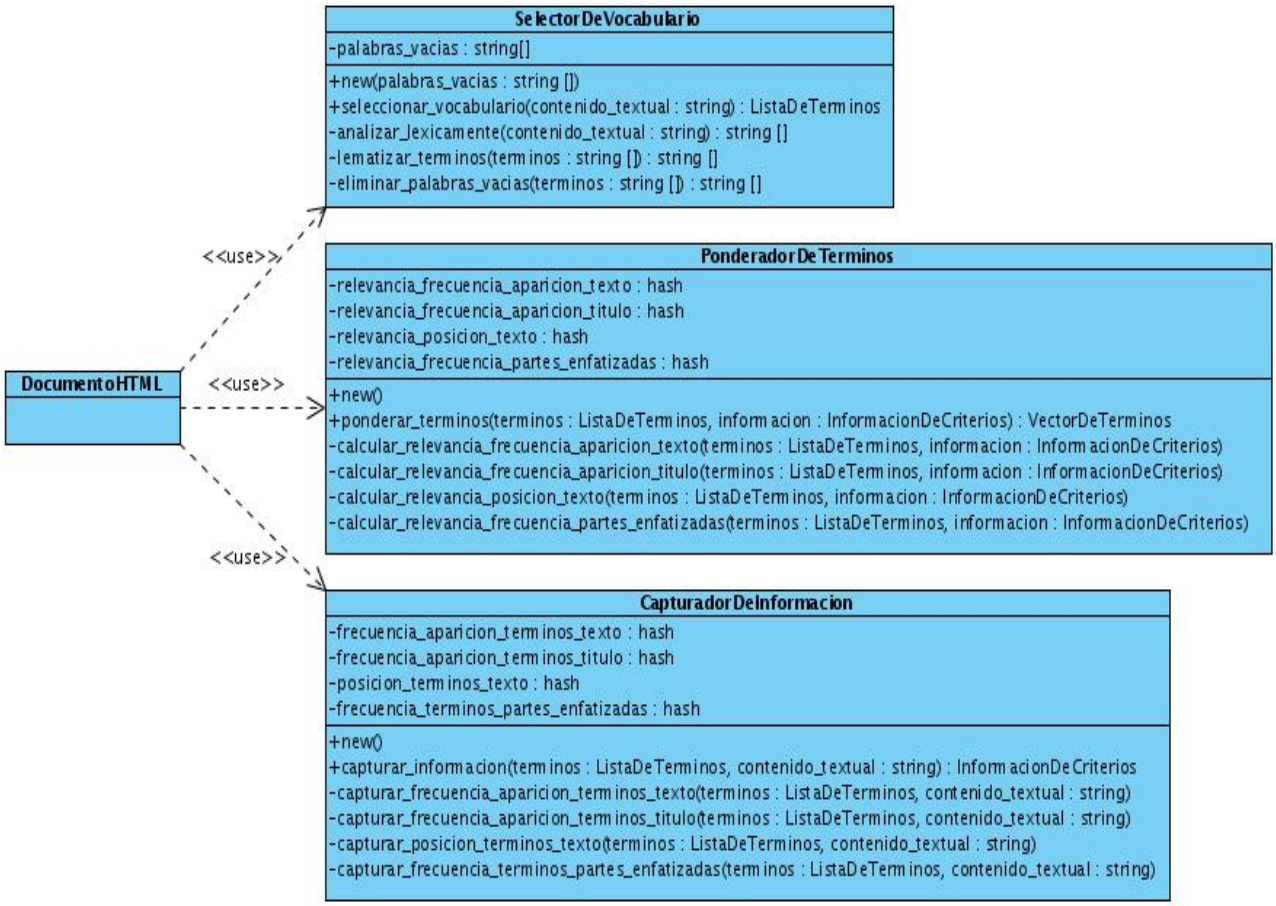


Figura 11: Diagrama de Clases del Diseño (continuación).

3.3.4. Descripción de las Clases

A continuación se describen las Clases del Diseño de ASADOHTML:

Nombre	`Categorizador`
Tipo de Clase	Controladora
Atributos	
Nombre	**Tipo**
`algoritmo_aprendizaje_automatico`	`Algorithm::SVM`
`entrenado`	`boolean`
`modelo_cargado`	`boolean`
Responsabilidades	
Nombre `new()`	
Descripción Crea un nuevo objeto `Categorizador` y lo devuelve.	
Nombre `entrenar( coleccion_entrenamiento : ColeccionDeEntrenamiento) : hash`	
Descripción Entrena al Categorizador, preparándolo para su labor de categorización. El parámetro coleccion_entrenamiento debe proveer un objeto de la clase `ColeccionDeEntrenamiento` poblada de varios documentos precategorizados. Devuelve estadísticas referentes al entrenamiento.	
Nombre `categorizar( documento : DocumentoHTML ) : hash`	
Descripción Asigna a un documento HTML un conjunto de categorías, cada una asociada a un valor numérico que indica el grado de pertenencia de tal documento a la misma.	
Nombre `guardar_modelo_categorizacion( localizacion : string ) : boolean`	
Descripción Guarda el modelo de categorización en la localización especificada. Devuelve verdadero si la operación se efectuó correctamente o falso en caso contrario.	
Nombre `cargar_modelo_categorizacion( localizacion : string ) : boolean`	
Descripción Carga el modelo de categorización desde la localización especificada. Devuelve verdadero si la operación se efectuó correctamente o falso en caso contrario.	
Nombre `entrenado() : boolean`	
Descripción Devuelve verdadero si el Categorizador está entrenado o falso en caso contrario.	
Nombre `modelo_cargado() : boolean`	
Descripción Devuelve verdadero si el Categorizador cargó un modelo de categorización o falso en caso contrario.	
Nombre `puede_categorizar() : boolean`	
Descripción Devuelve verdadero si el Categorizador está en condiciones de categorizar o falso en caso contrario.	

Tabla 14: Descripción de la Clase Categorizador.

Nombre	`ColeccionDeEntrenamiento`
Tipo de Clase	Controladora
Atributos	
Nombre	**Tipo**
`localizacion`	`string`
`archivo_categorias`	`string`
`iterador`	`string`
Responsabilidades	
Nombre `new( localizacion : string, archivo_categorias : string )`	
Descripción Crea un nuevo objeto `ColeccionDeEntrenamiento` y lo devuelve. Recibe como parámetros obligatorios la localización de la Colección de Entrenamiento y el nombre del archivo que contiene, para cada documento, las categorías asociadas.	
Nombre `proximo() : DocumentoHTML`	
Descripción Devuelve el próximo objeto `DocumentoHTML` de la Colección de Entrenamiento.	
Nombre `rebobinar()`	
Descripción Reubica el iterador en el primer documento de la Colección de Entrenamiento.	
Nombre `cantidad_documentos() : int`	
Descripción Devuelve la cantidad de documentos de la Colección de Entrenamiento.	

Tabla 15: Descripción de la Clase ColeccionDeEntrenamiento.

Nombre	`Categoria`
Tipo de Clase	Controladora
Atributos	
Nombre	**Tipo**
`documentos`	`ref DocumentoHTML[]`
`nombre`	`string`
Responsabilidades	
Nombre `new( nombre : string, documentos : ref DocumentoHTML[] )`	
Descripción Crea un nuevo objeto `Categoria` y lo devuelve. Recibe como parámetros el nombre de la categoría (obligatorio) y una lista (opcional) de referencias a objetos `DocumentoHTML` asociados a tal categoría.	
Nombre `por_nombre( nombre : string )`	
Descripción Devuelve el objeto `Categoria` con el nombre especificado o lo crea si no existe.	
Nombre `documentos() : DocumentoHTML[]`	
Descripción Devuelve la lista de objetos `DocumentoHTML` que pertenecen a la categoría.	
Nombre `contiene_documento( documento : DocumentoHTML ) : boolean`	
Descripción Devuelve verdadero si el documento especificado pertenece a la categoría o falso en caso contrario.	
Nombre `agregar_documento( documento : DocumentoHTML )`	
Descripción Agrega el documento especificado a la categoría.	
Nombre `nombre() : string`	
Descripción Devuelve el nombre de la categoría.	

Tabla 16: Descripción de la Clase Categoria.

Nombre	`DocumentoHTML`
Tipo de Clase	Controladora
Atributos	
Nombre	**Tipo**
`categorias`	`ref Categoria[]`
`nombre`	`string`
`terminos`	`VectorDeTerminos`
`contenido_textual`	`string`
`vocabulario`	`ListaDeTerminos`
`informacion`	`InformacionDeCriterios`
Responsabilidades	
Nombre `new( nombre : string, contenido_textual : string, categorias : ref Categoria[] )`	
Descripción Crea un nuevo objeto `DocumentoHTML` y lo devuelve. Recibe como parámetros el nombre del documento (obligatorio), el contenido textual del documento (obligatorio) y una lista (opcional) de referencias a objetos `Categoria` asociados a tal documento.	
Nombre `leer( localizacion : string )`	
Descripción Crea, a partir de la localización y el contenido textual de un documento HTML, un nuevo objeto `DocumentoHTML` y lo devuelve.	
Nombre `categorias() : Categoria[]`	
Descripción Devuelve la lista de objetos `Categoria` a las que pertenece el documento.	
Nombre `esta_en_categoria( categoria : Categoria ) : boolean`	
Descripción Devuelve verdadero si el documento pertenece a la categoría especificada o falso en caso contrario.	
Nombre `terminos() : VectorDeTerminos`	
Descripción Devuelve el Vector de Términos que representa al documento.	
Nombre `representar()`	
Descripción Representa el documento. Se invoca automáticamente al crear un objeto `DocumentoHTML`.	
Nombre `nombre() : string`	
Descripción Devuelve el nombre del documento.	

Tabla 17: Descripción de la Clase DocumentoHTML.

Nombre	`VectorDeTerminos`
Tipo de Clase	Entidad
Atributos	
Nombre	**Tipo**
`terminos_ponderados`	`hash`
Responsabilidades	
Nombre `new( terminos_ponderados : hash )`	
Descripción Crea un nuevo objeto `VectorDeTerminos` y lo devuelve. Recibe como parámetro (opcional) términos ponderados.	
Nombre `reemplazar( terminos_ponderados : hash )`	
Descripción Reemplaza los términos ponderados.	
Nombre `terminos_ponderados() : hash`	
Descripción Devuelve los términos ponderados.	
Nombre `longitud() : int`	
Descripción Devuelve la longitud del Vector de Términos.	
Nombre `agregar_vector( vector : VectorDeTerminos )`	
Descripción Agrega los términos ponderados del Vector de Términos especificado.	
Nombre `incluye_termino( termino : string ) : boolean`	
Descripción Devuelve verdadero si el término especificado pertenece al vector o falso en caso contrario.	
Nombre `valor_termino( termino : string ) : float`	
Descripción Devuelve el valor del término especificado.	

Tabla 18: Descripción de la Clase VectorDeTerminos.

Nombre	`SelectorDeVocabulario`
Tipo de Clase	Controladora
Atributos	
Nombre	**Tipo**
`palabras_vacias`	`string[]`
Responsabilidades	
Nombre `new( palabras_vacias : string[] )`	
Descripción Crea un nuevo objeto `SelectorDeVocabulario` y lo devuelve. Recibe como parámetro obligatorio una lista de palabras vacías.	
Nombre `seleccionar_vocabulario( contenido_textual : string ) : ListaDeTerminos`	
Descripción Devuelve la lista de términos a partir del contenido textual especificado.	
Nombre `analizar_lexicamente( contenido_textual : string ) : string[]`	
Descripción Analiza léxicamente el contenido textual especificado, devolviendo los términos que lo componen.	
Nombre `lematizar_terminos( terminos : string[] ) : string[]`	
Descripción Devuelve una lista con los términos lematizados.	
Nombre `eliminar_palabras_vacias( terminos : string[] ) : string[]`	
Descripción Elimina, de los términos especificados, las palabras vacías.	

Tabla 19: Descripción de la Clase SelectorDeVocabulario.

Nombre	`CapturadorDeInformacion`
Tipo de Clase	Controladora
Atributos	
Nombre	**Tipo**
`frecuencia_aparicion_terminos_texto`	`hash`
`frecuencia_aparicion_terminos_titulo`	`hash`
`posicion_terminos_texto`	`hash`
`frecuencia_terminos_partes_enfatizadas`	`hash`
Responsabilidades	
Nombre new()	
Descripción Crea un nuevo objeto `CapturadorDeInformacion` y lo devuelve.	
Nombre `capturar_informacion( terminos : ListaDeTerminos, contenido_textual : string ) : InformacionDeCriterios`	
Descripción Devuelve la información de criterios heurísticos de los términos en el contenido textual.	
Nombre `capturar_frecuencia_aparicion_terminos_texto( terminos : ListaDeTerminos, contenido_textual : string )`	
Descripción Captura la frecuencia de aparición de los términos en el contenido textual.	
Nombre `capturar_frecuencia_aparicion_terminos_titulo( terminos : ListaDeTerminos, contenido_textual : string )`	
Descripción Captura la frecuencia de aparición de los términos en el título.	
Nombre `capturar_posicion_terminos_texto( terminos : ListaDeTerminos, contenido_textual : string )`	
Descripción Captura la posición de los términos en el contenido textual.	
Nombre `capturar_frecuencia_terminos_partes_enfatizadas( terminos : ListaDeTerminos, contenido_textual : string )`	
Descripción Captura la frecuencia de los términos en las partes enfatizadas del contenido textual.	

Tabla 20: Descripción de la Clase CapturadorDeInformacion.

Nombre	`PonderadorDeTerminos`
Tipo de Clase	Controladora
Atributos	
Nombre	**Tipo**
`relevancia_frecuencia_aparicion_texto`	`hash`
`relevancia_frecuencia_aparicion_titulo`	`hash`
`relevancia_posicion_texto`	`hash`
`relevancia_frecuencia_partes_enfatizadas`	`hash`
Responsabilidades	
Nombre `new()`	
Descripción Crea un nuevo objeto `PonderadorDeTerminos` y lo devuelve.	
Nombre `ponderar_terminos( terminos : ListaDeTerminos, informacion : InformacionDeCriterios ) : VectorDeTerminos`	
Descripción Devuelve los términos ponderados según la información de criterios heurísticos.	
Nombre `calcular_relevancia_frecuencia_aparicion_texto( terminos : ListaDeTerminos, informacion : InformacionDeCriterios )`	
Descripción Calcula la relevancia de los términos, según su frecuencia de aparición en el texto.	
Nombre `calcular_relevancia_frecuencia_aparicion_titulo( terminos : ListaDeTerminos, informacion : InformacionDeCriterios )`	
Descripción Calcula la relevancia de los términos, según su frecuencia de aparición en el título.	
Nombre `calcular_relevancia_posicion_texto( terminos : ListaDeTerminos, informacion : InformacionDeCriterios )`	
Descripción Calcula la relevancia de los términos, según su posición en el texto.	
Nombre `calcular_relevancia_frecuencia_partes_enfatizadas( terminos : ListaDeTerminos, informacion : InformacionDeCriterios )`	
Descripción Calcula la relevancia de los términos, según su frecuencia en las partes enfatizadas del contenido textual.	

Tabla 21: Descripción de la Clase PonderadorDeTerminos.

Nombre	`ListaDeTerminos`
Tipo de Clase	Entidad
Atributos	
Nombre	**Tipo**
`terminos`	`string[]`
Responsabilidades	
Nombre `new( terminos : string[] )`	
Descripción Crea un nuevo objeto `ListaDeTerminos` y lo devuelve. Recibe como parámetro opcional una lista de términos.	
Nombre `agregar_termino( termino : string )`	
Descripción Agrega el término especificado a la lista de términos.	
Nombre `eliminar_termino( termino : string )`	
Descripción Elimina el término especificado de la lista de términos.	
Nombre `incluye_termino( termino : string ) : boolean`	
Descripción Devuelve verdadero si el término especificado pertenece a la lista de términos o falso en caso contrario.	
Nombre `terminos() : string[]`	
Descripción Devuelve la lista de términos.	

Tabla 22: Descripción de la Clase ListaDeTerminos.

Nombre	`InformacionDeCriterios`
Tipo de Clase	Entidad
Atributos	
Nombre	**Tipo**
`frecuencia_aparicion_terminos_texto`	`hash`
`frecuencia_aparicion_terminos_titulo`	`hash`
`posicion_terminos_texto`	`hash`
`frecuencia_terminos_partes_enfatizadas`	`hash`
Responsabilidades	
Nombre `new()`	
Descripción Crea un nuevo objeto `InformacionDeCriterios` y lo devuelve.	
Nombre `asignar_frecuencia_aparicion_texto( termino : string, frecuencia : int )`	
Descripción Le asigna a un término su frecuencia de aparición en el texto.	
Nombre `asignar_frecuencia_aparicion_titulo( termino : string, frecuencia : int )`	
Descripción Le asigna a un término su frecuencia de aparición en el título.	
Nombre `asignar_posicion_texto( termino : string, posiciones : int[] )`	
Descripción Le asigna a un término su frecuencia de aparición en cada una de las partes del texto.	
Nombre `asignar_frecuencia_partes_enfatizadas( termino : string, frecuencia : int )`	
Descripción Le asigna a un término su frecuencia de aparición en las partes enfatizadas del texto.	
Nombre `frecuencia_aparicion_texto( termino : string ) : int`	
Descripción Devuelve la frecuencia de aparición de un término en el texto.	
Nombre `frecuencia_aparicion_titulo( termino : string ) : int`	
Descripción Devuelve la frecuencia de aparición de un término en el título.	
Nombre `posicion_texto( termino : string ) : int[]`	
Descripción Devuelve la frecuencia de aparición de un término en cada una de las partes del texto.	
Nombre `frecuencia_partes_enfatizadas( termino : string ) : int`	
Descripción Devuelve la frecuencia de aparición de un término en las partes enfatizadas del texto.	

Tabla 23: Descripción de la Clase InformacionDeCriterios.

3.4. Tratamiento de Errores

Teniendo en cuenta que ASADOHTML será implementado en Perl, el tratamiento de posibles errores o excepciones en tiempo de ejecución debe hacerse usando la función `eval` seguida de un bloque de código. Los errores pueden ser propios del lenguaje (fuera de memoria y división por cero, entre otros) o generados por la función `die`. El siguiente ejemplo muestra como usar `eval` para tratar el error división por cero:

```
eval {
   $a = 10; $b = 0;
   $c = $a / $b;  # causa un error en tiempo de ejecución atrapado por eval
};
print $@;   # imprime "Illegal division by 0 at ejemplo.pl line 3"
```

Cuando el programa es compilado, Perl comprueba la sintaxis del bloque y genera código. Si encuentra un error en tiempo de ejecución, Perl ignora el resto del bloque `eval` y le asigna a la variable especial `$@` el texto del error correspondiente.

Para lanzar errores propios, se usará la función `die`. Perl sabe si un segmento de código se está ejecutando dentro de un `eval` y entonces, cuando `die` es invocado, simplemente asigna el texto de error (el argumento de `die`) a `$@` y salta a la sentencia que sigue al bloque `eval`. A continuación se muestra un ejemplo del uso de `die` para lanzar errores propios:

```
sub abrir_archivo {
   open(F, $_[0]) || die "El archivo no pudo ser abierto: $!";
}
$f = 'ejemplo.txt';
while (1) {
   eval {
      abrir_archivo($f);   # si abrir_archivo invoca a die, el programa no termina
   };
   last unless $@;   # no hubo error, salir del ciclo

   print "$f no existe. Por favor entre un nuevo archivo\n";
   chomp($f = <STDIN>);
}
```

Finalmente, los errores tratados deben registrarse en un archivo (*log file*). De este modo se dispondrá de información útil para el mantenimiento del sistema.

3.5. Concepción de la Ayuda

La ayuda de ASADOHTML debe ser simple e intuitiva para el usuario. Perl soporta un formato simple de marcado de texto llamado POD[43] que puede usarse, mezclado con el código fuente de ASADOHTML, para crear documentación incrustada. A continuación se muestra un ejemplo que evidencia lo anterior:

```
=head1
ejemplo.pl - Programa que calcula el poder de un equipo para la Euro 2008
=over 1
=item poder_de_equipo()
poder_de_equipo() devuelve el poder de un equipo en la escala del 1 al 10.
=cut
sub poder_de_equipo {
   my $equipo = shift;
      if ($equipo eq 'Alemania' || $equipo eq 'Holanda') {
         return 10;
      }
      return 0;
}
=head1
Final segura: Alemania vs. Holanda :)
=cut
```

POD puede ser convertido a varios formatos para ser adecuadamente impreso o visualizado. Para ello, se podrán utilizar las herramientas pod2text (lo convierte a texto plano), pod2man (lo convierte a un *manpage* de Unix), pod2html (lo convierte a HTML) y pod2latex (lo convierte a LaTeX).

43 http://www.cs.ait.ac.th/~on/O/oreilly/perl/prog3/ch26_01.htm

3.6. Conclusiones

Tras efectuar el Análisis de ASADOHTML se comprendieron sus requisitos de manera más precisa, analizándolos con mayor profundidad. Además, se consiguió preparar y simplificar las subsiguientes actividades del Diseño. Respecto a los Diagramas de Clases del Análisis que se obtuvieron, se concluye que pueden ser aplicados a varios diseños y cada una de sus clases de control, entidad e interfaz aislarán futuros cambios al comportamiento e información que representan.

Llevando a cabo el Diseño de ASADOHTML se encontró su forma y se obtuvo un sistema flexible a los cambios en los requisitos. Mediante los Diagramas de Secuencia se mostró la forma en que los objetos se comunican entre sí al transcurrir el tiempo, contribuyendo de manera importante a entender el comportamiento del sistema. Se evidenció que el lenguaje de programación seleccionado (Perl) provee mecanismos para tratar adecuadamente los errores y brindar la ayuda necesaria respecto al uso del sistema.

Finalmente se concluye que la habilidad más importante en el Análisis y Diseño orientados a objetos, asignar eficientemente las responsabilidades a los componentes del *software*, se cumplió satisfactoriamente. Además, se creó una entrada apropiada y un punto de partida para futuras actividades de implementación.

CONCLUSIONES

Una vez realizada la fundamentación teórica que sustentó este trabajo, definidas las características de ASADOHTML y efectuado el Análisis y Diseño del mismo, se obtuvieron resultados que le permiten a los autores presentar las siguientes conclusiones:

- El análisis de investigaciones similares, efectuadas en los diferentes ámbitos, arrojó que la mayoría poseen carácter privativo (imposibilitando el acceso a detalles de implementación) o no se adecuan a la finalidad requerida: categorizar documentos HTML para un *software* (MCADHTML) que creará y actualizará la BDUC de un Filtro de Contenidos. Por lo anterior, se evidenció la necesidad de concebir, analizar y diseñar un *software* de categorización (ASADOHTML).
- El estudio de características de la Web reflejó que su dinamismo, desmesurado tamaño actual, ritmo exponencial de crecimiento y la naturaleza de sus contenidos eran aspectos a considerar al definir las características de ASADOHTML; puesto que alertaban sobre el gran volumen de información que este último podría tratar. Todo sistema que se proponga lidiar con la información contenida en la Web estará a la sombra de dichos aspectos, por ello sugerimos que se les preste especial atención.
- HTML es el lenguaje de publicación más usado por la Web, por tanto, todo sistema que se proponga categorizar documentos de este medio (como ASADOHTML) deberá considerar los codificados en tal lenguaje y la valiosa información que proporcionan sus etiquetas.
- El uso de palabras aisladas y el Modelo de Espacio Vectorial para representar los documentos, en lugar de sintagmas y el Índice de Latencia Semántica, proporcionan mejores propiedades estadísticas y reducen drásticamente la complejidad computacional del problema (representación de documentos) respectivamente. Con respecto al uso de palabras aisladas, cabe destacar que deberá experimentarse con otro mecanismo que trate adecuadamente el *ruido* en los documentos, como por ejemplo, palabras escritas incorrectamente (errores ortográficos).
- La combinación de las fases Análisis Léxico, Lematización y Eliminación de Palabras Vacías es adecuada para seleccionar el vocabulario durante la representación de los documentos. Con respecto al Análisis Léxico, ha de efectuarse cuidadosamente cuando se aplica a documentos HTML en lugar de documentos en texto plano, ya que

los primeros incluyen notaciones propias del lenguaje que pueden afectar la correcta distinción de los términos que formarán parte de la representación. Por su parte, la Lematización y la Eliminación de Palabras Vacías reducen adecuadamente la cantidad de términos a tratar durante la representación.

- El uso de funciones de ponderación local, para representar los documentos de manera autocontenida, permite hacer frente al tamaño actual y futuro de la Web.
- El Algoritmo de Aprendizaje Automático SVM es independiente de la dimensión del espacio de características, evitando la complejidad computacional asociada a la selección de características.
- RUP es un marco de trabajo genérico que puede especializarse según las características específicas del desarrollo que se requiera. Fue adecuadamente configurado según las exigencias de ASADOHTML.
- Dadas las características distintivas de ASADOHTML, se concluye que podrá cumplir satisfactoriamente su labor dentro de MCADHTML.
- Tras efectuar el Análisis y Diseño de ASADOHTML, se obtuvo una entrada apropiada y un punto de partida para su futura implementación.

En resumen, las principales aportaciones de este trabajo son:

- Base teórica para el desarrollo de sistemas afines al que se diseñó.
- Análisis y Diseño del sistema que, en su futura implementación, contribuirá a crear y actualizar la BDUC de Filpacon; evitándose con ello, los inconvenientes generados por el dinamismo de la Web y la dependencia actual de dicho producto con las listas de URLs categorizadas por DMOZ, URLBlackList.com y Shalla.

Finalmente, los autores de este trabajo consideran que la idea que se propuso defender fue ratificada; es decir, la asignación de categorías temáticas al contenido textual de documentos HTML puede ser automatizada para contribuir al proceso de crear y actualizar la BDUC de Filpacon.

RECOMENDACIONES

- Efectuar la codificación o implementación de ASADOHTML a partir de lo pautado en este trabajo.
- Una vez implementado ASADOHTML, evaluar (cuantitativamente) su desempeño para diferentes escenarios o casos de pruebas.
- Realizar otras iteraciones de RUP sobre ASADOHTML para profundizar y mantener los artefactos obtenidos en este trabajo.
- Estudiar otros criterios que sean accesibles desde el código HTML y que puedan mejorar las representaciones de los documentos.
- Valorar el uso de *n-gramas* para reducir el *ruido* que introducen los errores ortográficos cuando se consideran palabras aisladas en la representación de documentos.

REFERENCIAS BIBLIOGRÁFICAS

ALONSO-ALLENDE, V. (2001). *Menores en la Red. Sistemas de filtro de contenidos nocivos*. [fecha de consulta: 4-febrero-2008]. Disponible en: http://www.informatica-juridica.com/trabajos/trabajosVarios.asp

ALPAYDIN, E. *Introduction to Machime Learning* [en línea]. Massachusetts: MIT Press, 2004 [fecha de consulta: 12-marzo-2008]. Disponible en: http://books.google.com/books?id=1k0_-WroiqEC&printsec=frontcover ISBN: 0262012111

BARKER, J. y KUPERSMITH, J. (2000). *Invisible or Deep Web*. [fecha de consulta: 1-febrero-2008]. Disponible en: http://www.lib.berkeley.edu/TeachingLib/Guides/Internet/InvisibleWeb.html

BERGMAN, M. K. *The Deep Web: Surfacing Hidden Value*. **Journal of Electronic Publishing** [en línea]. Agosto 2001, vol. 7, no. 1. [fecha de consulta: 3-febrero-2008]. Disponible en: http://hdl.handle.net/2027/spo.3336451.0007.104 ISSN: 1080-2711

BERMEJO, S. *Learning with nearest neighbour classifiers*. Tesis (Doctoral). Catalunya, España: Universitat Politécnica de Catalunya, 2000. [300] p. Disponible en: http://www.tdx.cat/TDX-0408103-145004/

BERNERS-LEE, T. [et al]. *World-Wide Web: The Information Universe*. **Electronic Networking: Research, Applications and Policy** [en línea]. Primavera 1992, vol. 2, no. 1. [fecha de consulta: 1-febrero-2008]. Disponible en: http://www.w3.org/History/1992/ENRAP/Article_9202.ps ISSN: 1051-4805

BHARAT, K. y BRODER, A. (1998). *A technique for measuring the relative size and overlap of public Web search engines*. [fecha de consulta: 3-febrero-2008]. Disponible en: http://www.ra.ethz.ch/CDstore/www7/1937/com1937.htm

CÁCERES, S. (2004). *La evolución de los contenidos en Internet*. [fecha de consulta: 3-febrero-2008]. Disponible en: http://www.fundacionorange.es/areas/28_observatorio/pdfs/7_SPC.pdf

CORTES, C. y VAPNIK, V. *Support-Vector Networks*. **Machine Learning** [en línea]. Septiembre 1995, vol. 20, no. 3. [fecha de consulta: 19-marzo-2008]. Disponible en: http://www.springerlink.com/content/w08253ul7m3780v8/

ISSN: 1573-0565

CRISTIANINI, N. y SHAWE-TAYLOR, J. *An Introduction to Support Vector Machines* [en línea]. Cambridge: Cambridge University Press, 2000 [fecha de consulta: 19-marzo-2008]. Disponible en: http://books.google.com/books?id=L-2Vqx56J5UC&printsec=frontcover&hl=es
ISBN: 0521780195

FERNÁNDEZ-VALMAYOR, A. [et al]. *Lenguajes de programación, lenguajes de marcado y modelos hipermedia: una visión interesada de la evolución de los lenguajes informáticos*. **Estudios de Lingüística del Español (ELiEs)** [en línea]. 2006, vol. 24. [fecha de consulta: 7-febrero-2008]. Disponible en: http://elies.rediris.es/elies24/fernandezvalmayor.htm
ISSN: 1139-8736

FIGUEROLA, C. G. [et al]. *Spanish Monolingual Track: The Impact of Stemming on Retrieval*. **Lecture Notes in Computer Science** [en línea]. 2002, vol. 2406. [fecha de consulta: 10-marzo-2008]. Disponible en: http://www.springerlink.com/content/d9jyl3y1rvwlem2j/
ISSN: 1611-3349

FIGUEROLA, C. G., ZAZO, A. F. y ALONSO, J. L. (2000). *Categorización automática de documentos en español: algunos resultados experimentales*. [fecha de consulta: 2-marzo-2008]. Disponible en: http://reina.usal.es/pub/figuerola2000categorizacion.pdf

FRESNO, V. D. *Representación Autocontenida de Documentos HTML: una propuesta basada en Combinaciones Heurísticas de Criterios*. Tesis (Doctoral). Móstoles, España: Universidad Rey Juan Carlos, Escuela Superior de Ciencias Experimentales y Tecnología, Departamento de Ingeniería Telemática y Tecnología Electrónica, 2006. 271 p.
Disponible en: http://www.escet.urjc.es/~vfresno/phd_sp.html

GULLI, A. y SIGNORINI, A. (2005). *The Indexable Web is more than 11.5 billion pages*. [fecha de consulta: 3-febrero-2008]. Disponible en: http://www.cs.uiowa.edu/~asignori/web-size/

JACOBSON, I., BOOCH, G. y RUMBAUGH, J. *El Proceso Unificado de Desarrollo de Software*. 1a. ed. Madrid: Addison Wesley, 2000. 464 p.
ISBN: 8478290362

JOACHIMS, T. (1997). *Text Categorization with Support Vector Machines: Learning with Many Relevant Features*. [fecha de consulta: 20-marzo-2008].

Disponible en: http://www.cs.cornell.edu/People/tj/publications/joachims_97b.pdf

KOSALA, R. y BLOCKEEL, H. (2000). *Web Mining Research: A Survey*. [fecha de consulta: 23-marzo-2008]. Disponible en: http://www.sigkdd.org/explorations/issue2-1/kosala.pdf

KRAAIJ, W. *TNO at CLEF-2001: Comparing Translation Resources*. **Lecture Notes in Computer Science** [en línea]. 2002, vol. 2406. [fecha de consulta: 10-marzo-2008]. Disponible en: http://www.springerlink.com/content/f3u10vhbr4xeg0gl/fulltext.pdf

LANDAUER, T. K., FOLTZ, P. W. y LAHAM, D. *An Introduction to Latent Semantic Analysis*. **Discourse Processes** [en línea]. 1998, vol. 25, no. 2-3. [fecha de consulta: 6-marzo-2008]. Disponible en: http://lsa.colorado.edu/papers/dp1.LSAintro.pdf ISSN: 0163-853X

LARMAN, C. *UML y Patrones. Introducción al análisis y diseño orientado a objetos*. 1a. ed. México: Prentice Hall, 1999. 536 p. ISBN: 9701702611

LAWRENCE, S. y GILES, C. L. *Accessibility of information on the web*. **Nature** [en línea]. Julio 1999, vol. 400. [fecha de consulta: 3-febrero-2008]. Disponible en: http://clgiles.ist.psu.edu/papers/Nature-99.pdf

LEWIS, D. D. (1990). *Representation Quality in Text Classification: An Introduction and Experiment*. [fecha de consulta: 7-marzo-2008]. Disponible en: http://www.aclweb.org/anthology-new/H/H90/H90-1057.pdf

LUHN, H. P. (1957). *A Statistical Approach to Mechanized Encoding and Searching of Literaty Information*. [fecha de consulta: 5-marzo-2008]. Disponible en: http://www.research.ibm.com/journal/rd/014/ibmrd0104D.pdf

MIRANDA, R. (2005). *Los menores en la Red: comportamiento y navegación segura*. [fecha de consulta: 15-febrero-2008]. Disponible en: http://www.financialtech-mag.com/_docum/50_Tendencias_05.pdf

O'NEILL, E. T., LAVOIE, B. F. y BENNETT, R. *Trends in the Evolution of the Public Web 1998-2002*. **D-Lib Magazine** [en línea]. Abril 2003, vol. 9, no. 4. [fecha de consulta: 3-febrero-2008]. Disponible en: http://www.dlib.org/dlib/april03/lavoie/04lavoie.html ISSN: 1082-9873

PORTER, M. F. *An algorithm for suffix stripping*. **Program** [en línea]. Julio 1980, vol. 14, no. 3. [fecha de consulta: 10-marzo-2008].

Disponible en: http://telemat.det.unifi.it/book/2001/wchange/download/stem_porter.html
PURESIGHT (2008). *PureSight Web Filter*. [fecha de consulta: 30-marzo-2008].
Disponible en: http://www.icognito.com/technology/index.shtml
PURESIGHT (2004). *Multilayer Dynamic Internet Content Filtering: An Overview of Next Generation Filtering Technology*. [fecha de consulta: 24-febrero-2008].
Disponible en: http://www.icognito.com/wpapers/puresight_whitepaper.pdf
SALTON, G., WONG, A. y YANG, C. S. *A Vector Space Model for Automatic Indexing*. **Communications of the ACM** [en línea]. Noviembre 1975, vol. 18, no. 11.
[fecha de consulta: 6-marzo-2008].
Disponible en: http://aidb.cs.iitm.ernet.in/cs625/salton-VectorSpaceModel.pdf
ISSN: 0001-0782
SEBASTIANI, F. *Machine Learning in Automated Text Categorization*. **ACM Computing Surveys (CSUR)** [en línea]. Marzo 2002, vol. 34, no. 1.
[fecha de consulta: 1-marzo-2008].
Disponible en: http://www.isti.cnr.it/People/F.Sebastiani/Publications/ACMCS02.pdf
ISSN: 0360-0300
STARYNKEVITCH, B. [et al] (2002). *POESIA Software Architecture*.
[fecha de consulta: 31-marzo-2008].
Disponible en: http://www.poesia-filter.org/pdf/Deliverable_3_1.pdf
TOPTENREVIEWS (2008). *Internet Filter Software Review 2008*.
[fecha de consulta: 20-febrero-2008].
Disponible en: http://internet-filter-review.toptenreviews.com/
VAPNIK, V. N. *The Nature of Statistical Learning Theory* [en línea]. [Berlín]: Springer, 2000
[fecha de consulta: 19-marzo-2008].
Disponible en:
http://books.google.com/books?id=sna9BaxVbj8C&printsec=frontcover&hl=es
ISBN: 0387987800
VENEGAS, R. *Clasificación de textos académicos en función de su contenido léxico-semántico*. **Revista Signos** [en línea]. 2007, vol. 40, no. 63.
[fecha de consulta: 1-marzo-2008].
Disponible en:
http://www.scielo.cl/scielo.php?pid=S0718-09342007000100012&script=sci_arttext
ISSN: 0718-0934
VILLATE, J. (2001). *Libertad de expresión en Internet*. [fecha de consulta: 15-febrero-2008].

Disponible en: http://www.cibersociedad.net/archivo/articulo.php?art=37

W3C (1999). *HTML 4.01 Specification*. [fecha de consulta: 10-febrero-2008].
Disponible en: http://www.w3.org/TR/1999/PR-html40-19990824/

WIKIPEDIA (2008). *World Wide Web*. [fecha de consulta: 1-febrero-2008].
Disponible en: http://es.wikipedia.org/wiki/World_Wide_Web

YANG, Y. *An Evaluation of Statistical Approaches to Text Categorization*. **Information Retrieval** [en línea]. Abril 1999, vol. 1, no. 1-2. [fecha de consulta: 15-marzo-2008].
Disponible en: http://www.springerlink.com/content/x3n6633584015p59/

YIMING, Y. *An evaluation of statistical approaches to text categorization*. **Journal of Information Retrieval** [en línea]. 1999, vol. 1, no. 1/2.
[fecha de consulta: 15-marzo-2008].
Disponible en: http://www.cs.cmu.edu/~yiming/publications.html
ISSN: 1386-4564

GLOSARIO DE TÉRMINOS

- **Ingeniería del Conocimiento** - Disciplina que forma parte de la Inteligencia Artificial y cuyo fin es el diseño y desarrollo de Sistemas Expertos. Para ello, se apoya en metodologías instruccionales y en las ciencias de la computación y de las tecnologías de la información, intentando representar el conocimiento y razonamiento humanos en un determinado dominio, dentro de un sistema artificial.
- **Inteligencia Artificial** - Rama de la Informática que desarrolla procesos que imitan la inteligencia de los seres vivos. La principal aplicación de esta ciencia es la creación de máquinas para la automatización de tareas que requieran un comportamiento inteligente.
- **Internet** - Red de ordenadores a nivel mundial. Ofrece distintos servicios, como el envío y recepción de correo electrónico, la posibilidad de ver información en las páginas web, de participar en foros de discusión, de enviar y recibir ficheros, de charlar en tiempo real, entre otros.
- **Procesamiento de Lenguaje Natural** - Disciplina que forma parte de la Inteligencia Artificial. Se ocupa de la formulación e investigación de mecanismos eficaces computacionalmente para la comunicación entre personas o entre personas y máquinas por medio de lenguajes naturales.
- ***Software* libre** - *Software* que brinda libertad a los usuarios sobre su producto adquirido y por tanto, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente.
- ***Software* privativo** - *Software* no libre, es decir, *software* o programa al que por medio de una licencia no libre se le ha privado de ciertas libertades de uso, modificación o distribución.
- **URL** - Secuencia de caracteres, de acuerdo a un formato estándar, que se usa para nombrar recursos, como documentos e imágenes en Internet, según su localización.