Universidad de la República
Facultad de Ingeniería
Instituto de Computación

Tesis de Grado
"Web Classification Wrappers "

Versión 3.0
Marzo de 2002

Laura Linzo
Marcela Pardo
Luis Rodríguez

Tutora: Regina Motz

1. Introducción

La World Wide Web (WWW) ha crecido vertiginosamente en los últimos años y es actualmente una fuente fundamental de información en prácticamente todas las áreas de interés. Los sistemas de manejo de la información global permiten a los usuarios realizar búsquedas y acceder a documentos. En este proceso se obtienen diferentes enlaces (links) a documentos que pueden contener o no la información relacionada al tema de la búsqueda, y el mismo tema puede estar tratado en distintos documentos desde diferentes puntos de vista. Para explotar esta información el usuario necesita explorarla y analizarla.

El objetivo principal de este trabajo es facilitar al usuario el acceso a la información correcta brindando herramientas para el análisis del contenido de páginas html. Este análisis está orientado a realizar clasificación automática de las páginas de acuerdo a distintos criterios dados por el usuario y a gerenciar el mantenimiento de ésta clasificación ante cambios en las páginas.

El presente trabajo se desarrolla en el marco de un proyecto del grupo de Concepción de Sistemas de Información del Instituto de Computación (CSI) **Proyecto EDW** [1] que plantea la propagación automática de cambios en la Data Warehouses (DW). Por cambios, se entienden los efectos de agregar, borrar y cambiar fuentes Web y datos, en el esquema de la DW. Dicho trabajo propone una arquitectura wrapper/mediador, que minimiza el impacto de los cambios de las fuentes Web en el esquema DW. La Figura 1 ilustra la arquitectura propuesta en el proyecto.

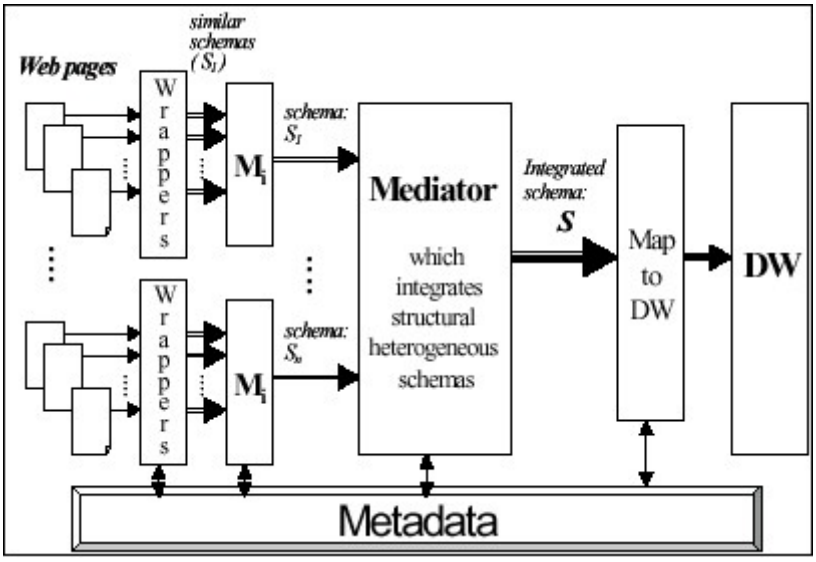


Fig. 1 – Arquitectura del Proyecto sobre Evolución de DW del CSI.

Se puede observar en la Figura 1 que la arquitectura del proyecto EDW parte de una clasificación previa de páginas. Estas páginas se encuentran agrupadas para su tratamiento por diferentes wrappers. La razón del agrupamiento inicial de páginas se debe a cierto grado de similaridad dentro de criterios dados por el usuario. En el proyecto EDW [1] se muestra cómo a partir de esta clasificación se simplifica la propagación de cambios hacia el data warehouse.

Nuestro trabajo desarrolla la primer parte de este proyecto..
En primer lugar, se brinda una herramienta para realizar la clasificación inicial de las páginas HTML de acuerdo a criterios dados por el usuario. En este sentido dentro del trabajo se definieron pautas para los criterios y se ofrece un prototipo para su uso. De esta forma se obtiene un conjunto de páginas agrupadas según un cierto criterio, permitiendo al usuario obtener información más detallada.

En segundo lugar, se brinda una especificación y prototipo para la extracción de información de las páginas clasificadas.
Con el detalle de que información se desea de cada subgrupo brindado por el usuario se extrae la información requerida de las páginas agrupadas y luego se convierte la información obtenida en documentos XML.

Por último, se realiza el análisis de los posibles cambios en las páginas html, se presentan herramientas que detectan algunos de estos cambios y se reflexiona sobre las formas de absorber los cambios dentro de la clasificación.

Resumiendo, el trabajo se divide en los siguientes tres módulos:

1. El módulo de ***Clasificación*****,** que consiste en clasificar las páginas en diferentes grupos.

2. El módulo de ***Extracción*****,** que se encarga de extraer de cada página la información requerida por el usuario. Esta tarea se ve facilitada por la clasificación anterior, ya que la información a extraer es relativa a cada grupo. Esta información es retornada como documentos XML.

3. El módulo de ***Evolución****,* que se encarga de mantener la clasificación y los documentos XML actualizados.

A continuación se presenta la organización de este documento:

Este documento consta de 7 capítulos, 6 apéndices y una bibliografía.

El Capitulo 1 realiza una introducción general al trabajo.

El Capítulo 2 presenta los conceptos preliminares necesarios para comprender el trabajo propuesto. Si el lector tiene conocimientos sobre *wrappers*, *information retrieval* o *expresiones regulares*, puede omitir la lectura de estos puntos en este capítulo. Los apéndices A, B y C explican más detalladamente cada uno de estos conceptos.

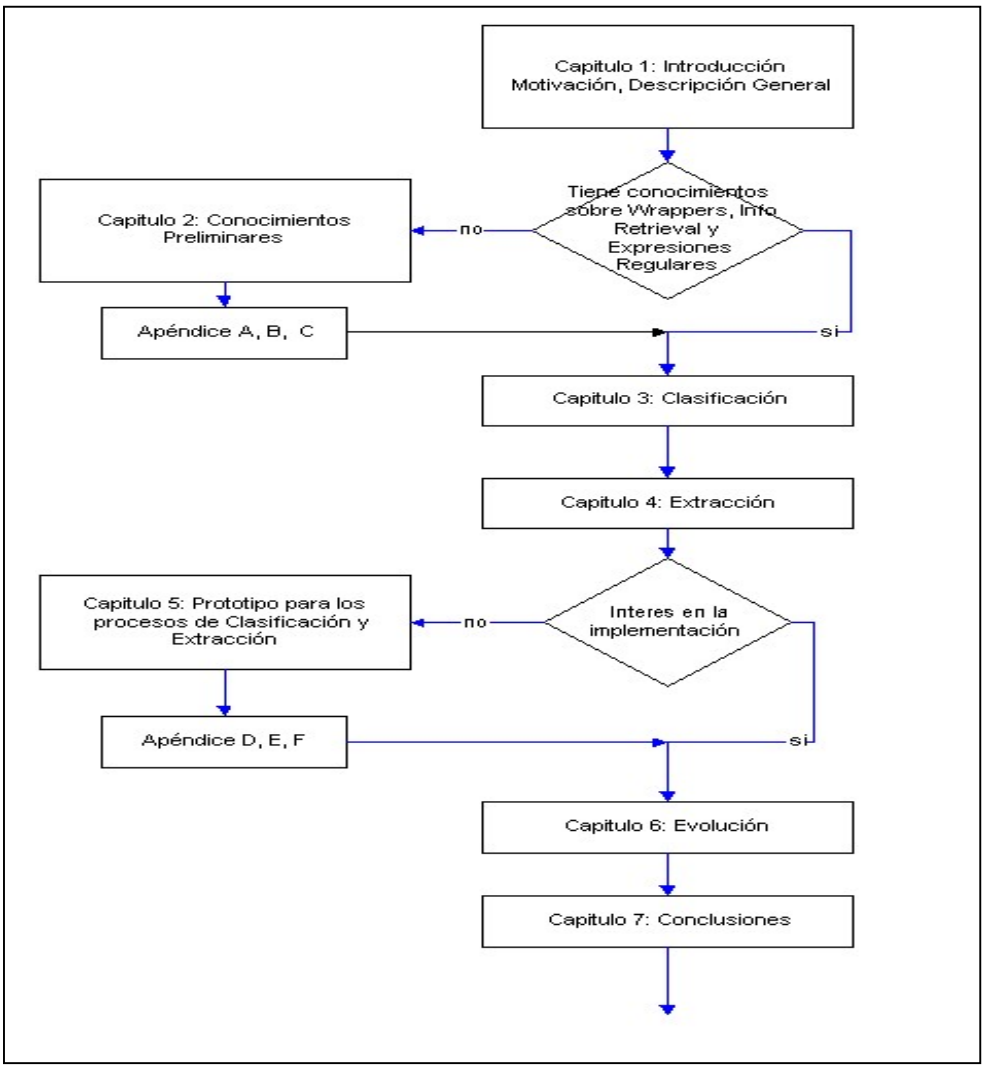
En el Capítulo 3 y 4 se explican los procesos de Clasificación y Extracción, respectivamente.

El Capítulo 5 describe brevemente el prototipo implementado para este proyecto (LPR01). En el apéndice D se presenta la especificación UML del prototipo, en el apéndice F se presentan los juegos de pruebas realizados y en el apéndice E se describe la base de datos.
Si el lector no desea adquirir conocimientos sobre la implementación del prototipo, puede omitir la lectura de este capítulo, sin influir en la comprensión del resto del documento.

En el Capitulo 6 se analiza el proceso de evolución.

En el Capítulo 7 se presentan las conclusiones, evaluación y trabajos futuros.

Finalmente se presenta un diagrama de flujo que especifica como puede ser leído este documento.



Junto con este Informe se entrega un CD con la siguiente documentación de proyecto: *manual de instalación del prototipo*, *cronograma del desarrollo del trabajo*, y *manual de usuario*.

2. Conceptos Preliminares

En este capítulo se presentan brevemente los conocimientos sobre los que se apoya el trabajo. Estos son: Wrappers, Information Retrieval y Expresiones regulares.

En la sección 2.1 se presenta una breve descripción de los extractores de información denominados *Wrappers*. Esta herramienta es utilizada en el proyecto para extraer información de las páginas Web.

En la sección 2.2 se presenta muy brevemente la metodología *Information Retrieval*.

En la sección 2.3 se explican que son las *Expresiones Regulares* y para que sirven.

2.1. Wrappers

Los wrappers tienen la finalidad de convertir datos de las fuentes de origen hacia un Modelo de Datos Común, también convierten consultas de aplicaciones globales en consultas específicas de las fuentes de Información.

En este proyecto se utilizó la herramienta W4F (Word Wide Web Wrapper Factory) para realizar la extracción de información sobre las diferentes páginas.
Por más información sobre W4F, ver Apéndice B.

2.2. Information Retrieval

Típicamente Information retrieval retorna documentos completos en respuesta a la información que el usuario necesita. Muchas veces, el usuario va a preferir examinar pequeñas porciones de un documento.

Estudio de los sistemas para indexarlos, hacer búsquedas y recuperar datos, en particular texto y otras formas no estructuradas. [2]

Los sistemas booleanos fueron los primeros en ser desarrollados hace 30 años, en un tiempo que el poder de cómputo era mínimo comparado con hoy. Por esto, los sistemas requerían que los usuarios suministraran de suficientes restricciones sintácticas en sus consultas para limitar el número de documentos recuperados, y estos documentos recuperados no están ordenados en relación a la consulta del usuario. Aunque los sistemas boléanos son muy poderosos en la búsqueda on-line para bibliotecarios y otros intermediarios entrenados, provee de un pobre servicio a los usuarios finales. Estos usuarios finales están familiarizados con la terminología del conjunto de datos que están buscando, pero pierden el entrenamiento y práctica necesaria para obtener constantemente buenos resultados. La aproximación de rankeo (ranking approach) para retornar información, parece ser más orientada a usuarios finales. Esta aproximación permite al usuario ingresar una simple consulta tal como una oración o frase y recuperar una lista de documentos rankeados en orden de relevancia. [3]
En este sentido es que se usan técnicas de IR dentro del trabajo.

2.3. Expresiones regulares

Una expresión regular es un patrón de texto consistente en una combinación de caracteres alfanuméricos y caracteres especiales denominados meta caracteres. Una “close relative” es en definitiva la expresión comodín que se usa usualmente en el manejo de archivos. El patrón es usado para realizar match contra cadenas de texto. El resultado de un match puede ser exitoso o no, aunque cuando un match es exitoso no todos los patrones deben coincidir.

Se encuentra que las expresiones regulares son usadas en tres maneras diferentes: match de texto regular, buscar y reemplazar y splitting. Este último es básicamente lo mismo que match inverso ie. todo lo que no hace match con la expresión regular.

Debido a la versatilidad de las expresiones regulares, son ampliamente usadas en el procesamiento de texto y parsing. [4]

El proyecto se basa en el contenido de las páginas y para definirlo se utilizan expresiones regulares estilo Java. [5]

3. Clasificación

En este capitulo se trata el problema de "clasificar" un conjunto de páginas Web. Estas páginas son obtenidas a partir de una búsqueda realizada sobre un *dominio* determinado, por ejemplo, *"Medicamentos"*.
Intuitivamente la palabra "clasificar" significa, determinar a que grupo pertenece un cierto objeto. Por lo tanto, la idea aquí es agrupar las diferentes páginas web según sea su contenido, en diferentes grupos que llamaremos *subdominios*.
Dado un *dominio*, se divide este, obteniéndose un conjunto de *subdominios*, que pueden ser excluyentes o no.
Para decidir cuando una página pertenece o no a un cierto *subdominio*, se debe determinar que "*tipo de información*" especifica dicho *subdominio*. Por ejemplo, analizando una página web se puede decidir a través de su contenido, si esta trata sobre el tema *"tratamientos"*, o *"venta de medicamentos"*
Por lo tanto para realizar la "clasificación" se debe especificar que información define un *subdominio* determinado, esta información es lo que llamaremos "criterios de clasificación". Utilizando los *criterios de clasificación*, se decide si una página clasifica o no en un subdominio determinado.

Dada una *página web*, y un conjunto de *subdominios* puede ocurrir lo siguiente:

- la página clasifica en un subdominio determinado.
- la página clasifica en más de un subdominio.
- la página no clasifica, y en ese caso se descarta.

En la siguiente figura se presenta un esquema del proceso de clasificación.

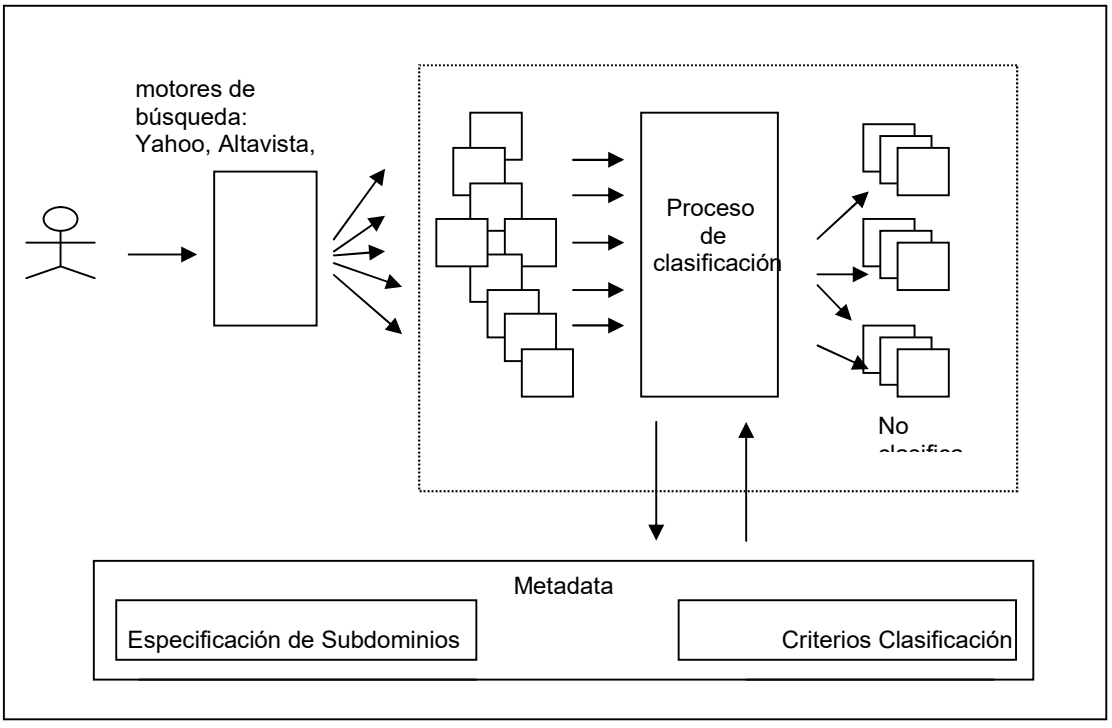


Fig. 2 – Arquitectura del proceso de clasificación

El usuario realiza una consulta con algún motor de búsqueda. Con las páginas obtenidas en la búsqueda, la especificación de subdominios y los criterios de clasificación, se realiza el proceso de clasificación, obteniéndose las páginas clasificadas.

3.1. Solución Propuesta

Como se mencionó con anterioridad, la solución que se propone está basada en el contenido de las páginas.
Se contemplan dos aspectos de los documentos: el contenido propiamente dicho (o sea las palabras o caracteres que se encuentran en el documento), y la presentación.

Con presentación nos referimos a la forma en como se ve el texto, o sea negrita o subrayado, el tipo de letra, el tamaño de la misma. Se tratará más en detalle en la sección 3.2.4.

Para realizar la clasificación, se utiliza la definición de los subdominios, y una asignación de pesos a la presentación (conjunto de tags). Dicha asignación de pesos se utiliza para el dominio en su totalidad.

La definición de un subdominio incluye una lista de expresiones regulares, un valor mínimo y otro máximo.
Estos dos últimos para determinar si la página pertenece o no al subdominio.

Con la información de la presentación y la especificación de los subdominios , se calcula un valor total para cada página, y luego se compara este con los valores máximo y mínimo del subdominio particular para determinar si la página pertenece a este o no.

Para calcular el peso de cada página, se suman los pesos individuales de cada ocurrencia de una de las expresiones regulares, multiplicado por el valor que le corresponda según su presentación.

Desde la sección 3.2.2 hasta la sección 3.2.5 inclusive, se explican los algoritmos y las estructuras de datos utilizados con mayor detalle.

En este proyecto los subdominios son organizados en forma de árbol (ver ejemplo en Figura 3), la idea es que solo las páginas que clasificaron dentro del subdominio padre se evalúan en el hijo.

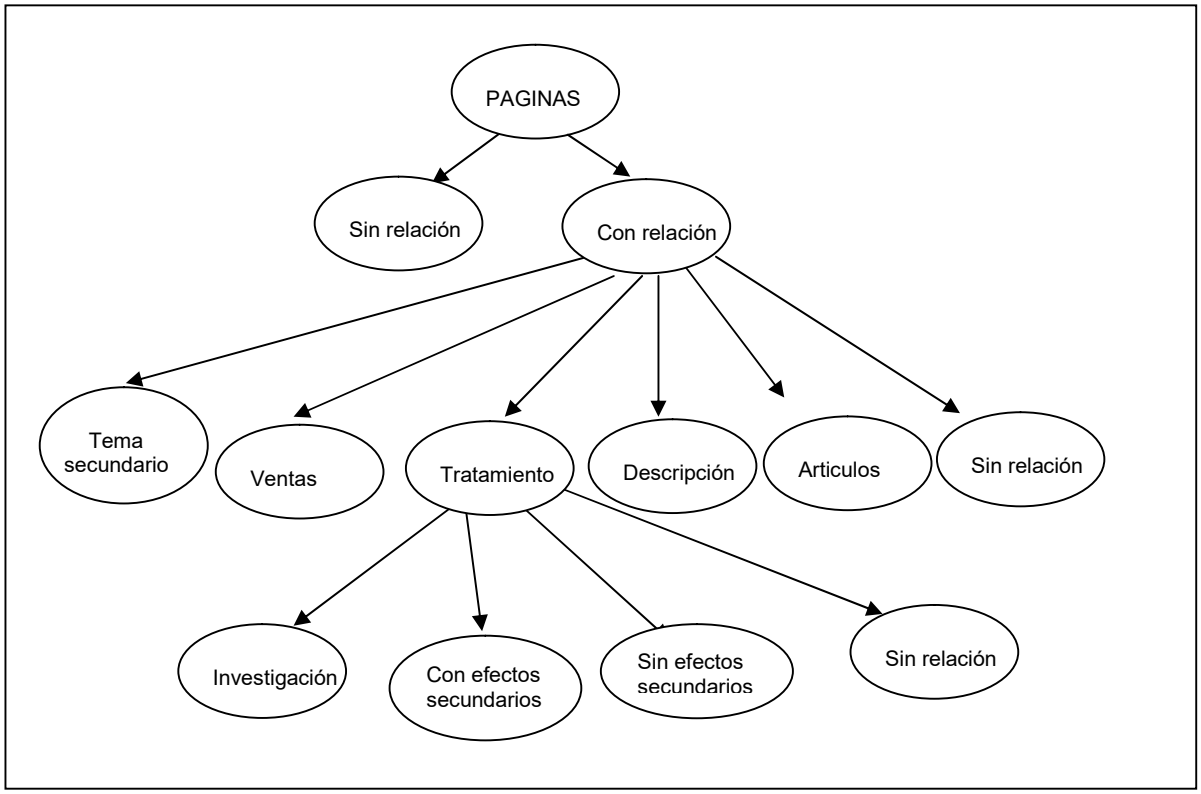


Fig. 3 – Ejemplo de Arbol de clasificación

3.1.1. ¿POR QUÉ UN ÁRBOL?

En principio, se puede pensar en clasificar utilizando solamente un conjunto de dominios sin estructura. Pero la estructura de árbol nos brinda las siguientes ventajas:

- Optimiza la clasificación, de dos formas:
    - No es necesario que todas las páginas se evalúen para cada dominio. El proceso de clasificación es bastante costoso, si existe una clasificación que al inicio divide las páginas en dos grandes grupos excluyentes, se reduce a la mitad la cantidad de evaluaciones necesarias. Si los nodos fueran excluyentes esto reduce el número de clasificaciones en orden logarítmico con respecto a la opción del conjunto de dominios.
    - Los conjuntos de expresiones regulares son de menor tamaño. Al tener varias instancias de clasificación (niveles del árbol), las expresiones regulares se dividen entre todas ellas, probablemente a lo largo de la clasificación de una página, en ella se busquen menos expresiones regulares que las que se evalúan utilizando la otra opción.

- La clasificación que se obtiene es más fina, ya que en un determinado nivel del árbol, ciertas expresiones regulares pueden tener más significado que en niveles anteriores. Esto es porque al estar dentro de un nivel del árbol dado, implica que clasificó en ese nodo, por lo tanto de ahora en más se puede analizar la página sabiendo que está dentro de un determinado contexto. Por ejemplo, se sabe que una página pertenece al nodo “Universitario”, en adelante se puede interpretar la palabra carrera, sabiendo que se refiere a una carrera universitaria y no a un evento deportivo.

- La estructura de árbol es más intuitiva, por lo tanto resulta más sencillo para el usuario armar la clasificación. Esto no es un punto menor, ya que la construcción de los criterios de clasificación por parte de un usuario experto en el dominio es fundamental para que la clasificación tenga éxito.

3.1.2. SUB-DOMINIOS

Un subdominio es una división conceptual de las páginas.
Por ejemplo, páginas que traten sobre el tema “investigación”.
Un usuario experto en el tema define los subdominios, especificándolos en la Metadata.

3.1.3. CRITERIO DE CLASIFICACIÓN

Un criterio de clasificación esta asociado a un subdominio especifico y permite determinar que paginas pertenecen a este. Esta compuesto por expresiones regulares que tienen un cierto peso asociado. Un usuario experto en el tema es quien brinda las expresiones regulares.

3.1.4. PRESENTACIÓN

En un documento *HTML* se encuentran diferentes tipos de datos: texto, imágenes, sonido, etc. Estos datos, a la vez, aparecen con determinada presentación; como por ejemplo: títulos, subtítulos, párrafos, negrita, subrayado.
Debido a que este proyecto esta orientado al contenido de la página, solo se consideran los datos que aparecen como texto en el documento. Se considera importante tanto el contenido como la presentación de este. Por ejemplo, cuando se observa una página, el título de esta, aporta información sobre lo que trata. Pero esta información no sólo esta dada por el contenido, sino también, por destacarse del resto del texto. Esto no solo se da para el título del documento, sino también para los subtítulos, títulos de las tablas, listas, etc.

Por lo tanto se debe idear un mecanismo, para manejar la información que nos aporta la presentación del documento. A continuación vemos los posibles enfoques que han surgido en el desarrollo del proyecto:

1. El primero y el de más sencilla implementación, es considerar los tags entre los que se encuentra la expresión regular que se está buscando.
   Esto es, si está entre los tags de título, tiene mas relevancia a la hora de decidir la clasificación de la página que se está evaluando.

   Se pueden considerar otros tags distintos a título, por ejemplo los tags "*Hx*", que definen los subtítulos de una página.

   <TITLE> texto </TITLE>
   <H1> texto </H1>
   <H2> texto </H2>
   <H3> texto </H3>
   <H4> texto </H4>
   <H5> texto </H5>
   <H6> texto </H6>

   Y asignar a cada opción un peso.

   El problema con este mecanismo es que los documentos *HTML* no siempre utilizan este tipo de tags. El mismo efecto que produce un tag *H1* por ejemplo, puede ser causado utilizando el tag *font* con *size* mayor que el resto del texto del documento.

   Para solucionar este problema, se puede definir que combinación de tags con sus respectivos atributos hacen que el texto comprendido sea considerado de mayor o menor peso. Por ejemplo puede considerarse el tag *font* con sus distintos atributos:

   ** texto **

   El problema que surge a partir de esto, es que no se puede definir valores fijos de atributos para determinar cuando un texto es considerado un título o subtítulo, ya que para diferentes paginas un texto entre tags *font* con atributo, por ejemplo, *size* 18 puede ser un título y en otra página, puede ser simplemente un texto cualquiera.

2. Un mecanismo que considere este problema es:
   Analizar todos los tags *font* que se encuentran en esta página, y asignarle al de mayor tamaño, el mayor peso, y así sucesivamente.
   Este método tiene el inconveniente de que pueden existir otros criterios para resaltar el texto. Por ejemplo el color.
   No alcanza para solucionar este problema, agregar el atributo *color* del tag *font* al algoritmo de asignación de pesos, esto lo mejora pero existen otras formas de resaltar texto a los ojos de un lector.

3. La solución es hacer este mecanismo de asignación de pesos, parametrizable, es decir, permitir al usuario definir que es considerado un título dentro de los documentos, que es un subtítulo, que es un texto destacado, etc. y asignar a estos conceptos un peso.
   Por ejemplo un usuario puede especificar como título todo texto que aparezca entre los siguientes tags *HTML*: *title, h1, font size 3* y asignarle como peso un valor determinado. Este concepto se aplica para todo el conjunto de páginas.
   Este mecanismo es un poco rígido, ya que el conjunto de páginas sobre el cual se desea hacer la clasificación puede ser demasiado heterogéneo como para aplicar este tipo de solución.

4. Puede aplicarse el mismo criterio (volver nuestras definiciones paramétricas) pero de una forma más dinámica, por ejemplo: especificar como título el tag *font* más grande de la página o el tag *font* de tamaño siguiente en negrita.

Por más información sobre tags, ver Apéndice A.
Actualmente se utilizo la opción 1 dada la limitación de tiempo del proyecto y la simplicidad de implementación.

3.1.5. PROCESO DE CLASIFICACION

Este módulo implementa el proceso de clasificación de páginas HTML en diferentes subdominios. Recibe como entradas: los criterios de clasificación para cada subdominio, el conjunto de tags que se consideran relevantes dada la presentación de las paginas, el árbol de clasificación y el conjunto de páginas HTML a clasificar, y devuelve como resultado información sobre como quedaron clasificadas las diferentes páginas.

El proceso se puede dividir en diferentes etapas: **extracción de información, cálculo de pesos, y clasificación.**

En la etapa de **extracción de información** se implementan mecanismos para extraer bloques de información de las páginas web, a partir de ciertos *criterios de extracción* que recibe como entrada.

El procedimiento **cálculo de pesos**, se encarga de asociarle a cada página un cierto valor, que determinará su clasificación en los diferentes *subdominios*.
Por último se desarrolla el procedimiento **clasificación** que determina si una página clasifica o no dentro de un *subdominio* determinado.

La estrategia planteada para resolver este proceso consiste en recorrer una única vez el *árbol de clasificación* , ya que es un proceso sumamente costoso (tanto la creacion como la ejecucion de los wrappers son procesos costosos).
La recorrida del arbol se realiza en profundidad, visitando primero al nodo padre, y los hijos se recorren de izquierda a derecha.

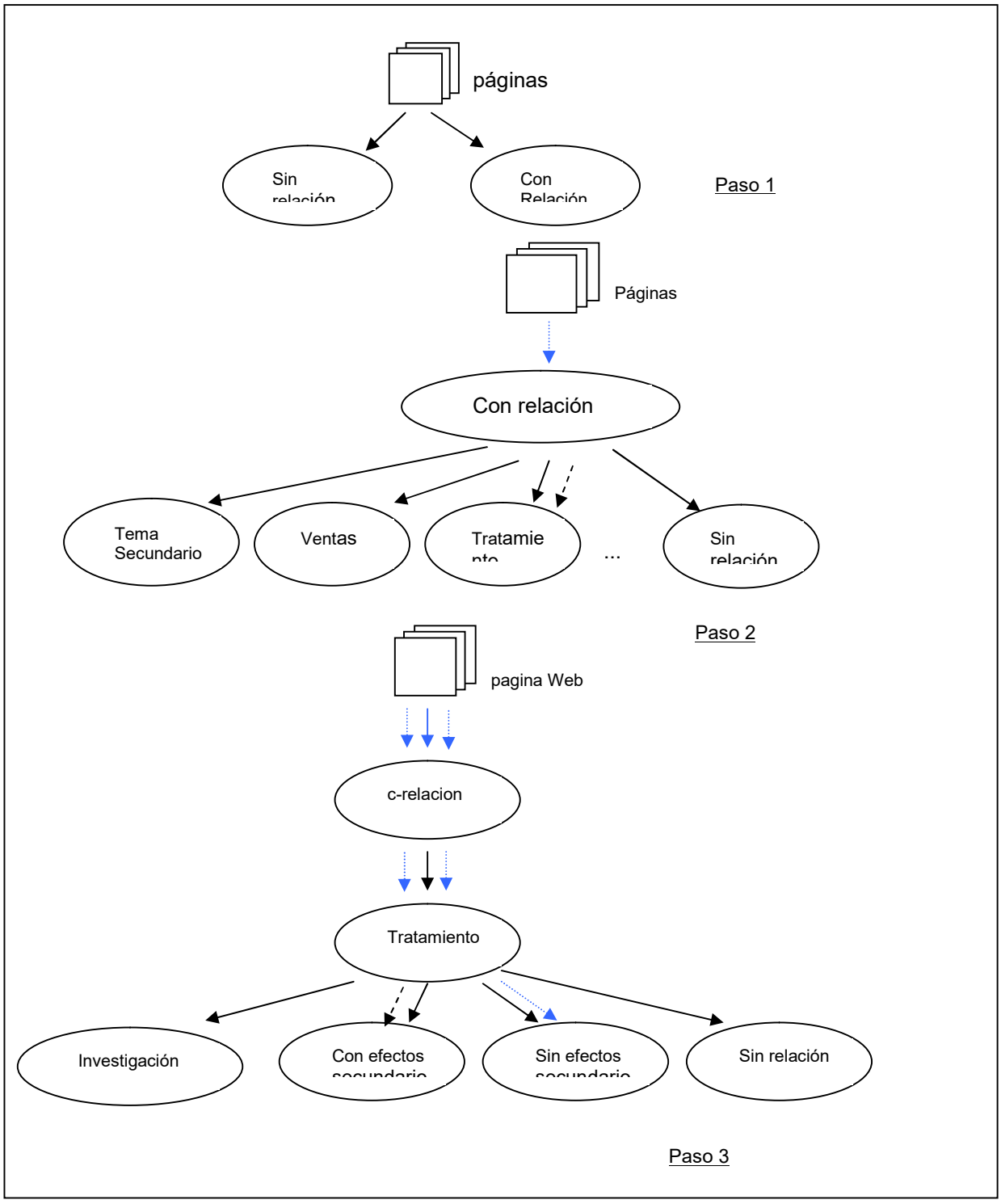


Fig. 4 – Proceso de clasificación

En la figura 4 se muestra como se realiza el proceso de clasificación.

En el paso 1 se clasifican todas las páginas teniendo en cuenta si tienen relación o no con el dominio especificado por un usuario experto.

Las páginas que clasificaron en el subdominio "*ConRelacion*" son clasificadas nuevamente en subdominios más específicos, las restantes son descartadas.

En cada paso del proceso de clasificación puede ocurrir lo siguiente con cada una de las páginas:

1) clasifica en un subdominio
   (ver figura 4 – paso 2)

2) clasifica en varios subdominios
   (ver figura 4 – paso 3)

3) no clasifica. En este caso se descarta para ese subdominio.

3.1.5.1. Proceso - Extracción de información

En esta etapa se desarrolla un mecanismo para realizar, la extracción de información de paginas Web. Para implementar este tipo de proceso, se utilizó la herramienta W4F [6][7] que ofrece una forma rápida para escribir dichos componentes especializados denominados *wrappers*. Esta herramienta divide el proceso de desarrollo del *wrapper* en una fase de extracción y otra de mapeo. Debido a la necesidad de realizar los *wrappers* en forma dinámica, se desarrollan un conjunto de clases Java que encapsulan algunas de las funcionalidades que ofrece dicha herramienta.
Por lo tanto a partir de estas clases Java se implementan diferentes procedimientos para: *generar, compilar y ejecutar los wrappers* en forma dinámica, a medida que se recorre el *árbol de clasificación.*

Por cada *subdominio (nodo del árbol de clasificación)* se construye un wrapper especifico utilizando la información que éste contiene.
Para generar cada uno de los *wrappers* se especifica que información (*expresiones regulares*) se quiere extraer de las paginas para los diferentes *subdominios*, y sobre que parte de la página (*tags*) se quiere extraer dicha información. Esta información es lo que denominamos *criterios de extracción*.

Procedimiento: *Generación de un wrapper*

Este procedimiento permite escribir los diferentes sectores que especifican un wrapper a través de un conjunto de clases Java.

Entrada:
- Un conjunto de *expresiones regulares* asociadas a un *subdominio* determinado.
- Un conjunto de *tags HTML* mediante los cuales se realizara la extracción.
- Nombre del wrapper

Salida:
- Un archivo de descripción, con extensión w4f que contiene toda la información para construir un wrapper.

Seudocódigo: *Generación de reglas de extracción*

A continuación se presenta el seudocódigo para la construcción de reglas de extracción:

```
Por cada expresión regular
  Por cada tag html
    genero regla de extracción de la siguiente manera:
        rule_i = html.tag.txt  = expresion regular
```

Procedimiento: *Compilación de un wrapper*

En este paso se implementa un procedimiento que permite compilar un wrapper utilizando una clase *Java*.

Entrada:

- nombre y ubicación del wrapper

Salida: La compilación del wrapper genera los siguientes archivos:

- *WraperName.Java* – contiene el código Java generado automáticamente.
- *WrapperName.class* – es la clase compilada para el wrapper.
- *WrapperName_get.class* – es una clase interna correspondiente al método retrieval.

Procedimiento: *Ejecución wrappers*

Este procedimiento implementa un mecanismo para ejecutar un wrapper a partir de una clase java.

Entrada:

- url de la página sobre la cual se va a aplicar el wrapper generado en el paso anterior.

Salida:

- Estructura auxiliar que almacena la información extraída de una página HTML.

3.1.5.2. Proceso - Cálculo de pesos y clasificación.

Este procedimiento se encarga de calcular el peso de una página a partir de cierta información que recibe como entradas. El resultado de éste sirve como entrada al proceso *clasificación* que se encarga de determinar si una página clasifica o no dentro de un *subdominio* especifico.

Procedimiento: *Cálculo de pesos*

Entrada:

- Estructura que devuelve el proceso de ***ejecución de un wrapper***

Salida:

- Peso de la página.

Seudocódigo: *Cálculo de pesos*

peso_total = 0
Para cada *expresión regular e* asociada a un nodo *i* del *árbol de clasificación*
    Para cada *tag HTML t*
        contar_ocurrencias de la *expresión regular e* en el tag *t* en una página.
        peso_de_expresion_regular = multiplicar la *cantidad_ocurrencias* de *e* por el peso del tag *t* por el peso de la expresion regular *e*.
        peso_total = peso_total + *peso_total_expresion_regular*
    fin de para cada tag
fin para cada expresión regular

Procedimiento: *clasificación*

Entrada:

- Salida del procedimiento *calculo de pesos* (peso de la página)

Salida:

- Devuelve si una pagina clasifica en un *subdominio* determinado.

Seudocódigo: *Clasificación*

Para cada *nodo i* del *árbol de clasificación*
Para cada *pagina p* pertenecientes al nodo *i*
peso_pagina = calculo_pesos (p)
Si *peso_pagina* cae en el *rango_de_clasificación*
pagina *p* clasifica en el nodo *i*

Esquema: *Proceso de clasificación*

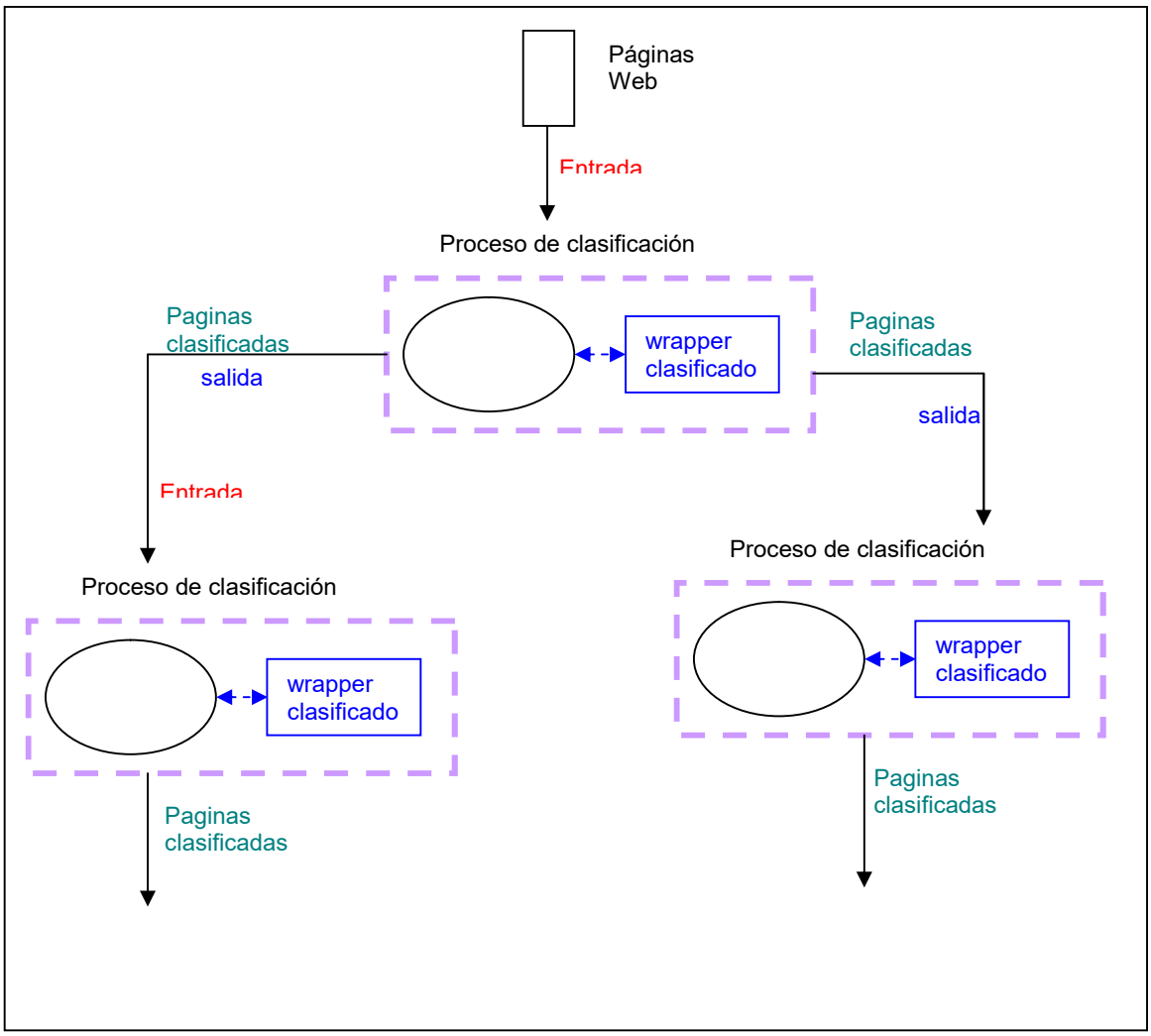


Fig. 5 – Esquema del proceso de clasificación

Procedimiento: *Proceso de clasificación:*

Entradas

- Arbol de clasificación
  - Especificación de cada *sub-dominio*
  - El *rango de valores* que determinan que una página clasifique o no en un *sub-dominio*
  - Wrapper asociado a cada *sub-dominio*.
- Un conjunto de *tags html* que determinan sobre que lugar de la página se extraerá la información.
- Paginas a clasificar.

Salida

- Estructura que almacena como quedo clasificada cada pagina HTML en los diferentes *subdominios.*

Seudocódigo – *Proceso de clasificación*

Para cada nodo *i* del *árbol de clasificación*
    Para cada página *j* asociada al nodo *i*
        Si no *existe_wrapper* para el nodo *i*
            generar_y_compilar_wrapper para el nodo *i*
        ejecutar_wrapper asociado al nodo *i*, sobre la página *j*
        calcular_peso de la página *j*
        Si *clasifica* la página *j* en el nodo *i*
            agregar página *j* al conjunto de páginas clasificadas para el nodo *i*.
    fin Para cada pagina
fin para cada nodo

- *existe_wrapper (nodo i)* : Determina si para el nodo *i*, ya fue creado un wrapper.
- *generar_y_compilar_wrapper (nodo i, informacion)*: Crea un nuevo wrapper para el nodo *i* a partir de cierta información que recibe como entrada, y lo compila.
- *ejecutar_wrapper (nodo i, pagina j)*: Ejecuta el wrapper asociado al nodo *i*, sobre la página *j*.
- *calcular_pesos (pagina j)*: Le asigna un determinado valor a la página j.
- *clasifica (nodo i, pagina j)* : Determina si la página *j* clasifica o no en el nodo *i*.
- *agregar (nodo i, pagina j)*: Agrega la página *j* al conjunto de páginas clasificadas en el nodo *i*

3.1.6. USO DE LA HERRAMIENTA W4F.

Resulto ser una buena herramienta y la única dificultad encontrada con respecto a este proyecto, es la imposibilidad de controlar la extracción de información desde tags anidados.

A continuación se explica este problema a través de un ejemplo.

Sea el siguiente fragmento de código HTML:

```
<table>
        <tr>
                <td>información a extraer</td>
        </tr>
</table>
```

Si se busca todas las ocurrencias de la palabra información que se encuentren dentro de un td, y luego todas las ocurrencias de la misma palabra dentro del tag tr, el fragmento de texto del ejemplo se obtendrá como resultado las dos veces. Esto no es un resultado erróneo, pero en el contexto de nuestro proyecto esto implica que esa ocurrencia se tomara en cuenta dos veces para calcular el peso total de la página.

4. Extracción

En este capitulo se trata el problema de extracción de información de paginas clasificadas en los diferentes subdominios.

Para realizar esta extracción es necesario que el usuario proporcione una especificación de la información que desea extraer de cada subdominio.

4.1. SOLUCIÓN PROPUESTA

Para cada subdominio se especifica que información se desea extraer. Esto se realiza definiendo un conjunto de conceptos sobre los diferentes subdominios.

Por ejemplo, dado el subdominio “ventas”, se desea extraer de las paginas que clasificaron en él, la información asociada con el concepto “precio”.

Observación: un mismo concepto puede estar asociado a más de un subdominio.

A partir de la definición de los conceptos, del resultado de la clasificación y de las paginas HTML, se extrae la información deseada y se almacena en formato XML.

4.1.1. CONCEPTOS

Los conceptos son la unidad de extracción. Su definición contiene un nombre y un conjunto de criterios de extracción.

Los criterios de extracción son la estructura de datos utilizada para especificar que información se desea extraer.

Un criterio de extracción consta de una expresión regular principal, un conjunto de expresiones regulares complementarias, un valor máximo y un valor mínimo.
Dado un criterio de extracción, se busca la expresión regular principal en el documento, y se extraen todos los fragmentos de texto en los que se encuentre. Luego se utiliza el conjunto de expresiones regulares complementarias para evaluar el contexto. La forma de evaluarlo es buscar cada una de las expresiones regulares complementarias y sumar el peso asociado a estas por cada ocurrencia, si el valor obtenido para todo el fragmento, luego de evaluar todo el conjunto, se encuentra entre los valores establecidos (entre el valor mínimo y el valor máximo) para el criterio de extracción, este fragmento se considera parte del resultado de la extracción, sino es desechado.

4.1.2. PROCESO DE EXTRACCIÓN

Este módulo implementa el proceso de extracción de información. Recibe como entradas el conjunto de paginas clasificadas en los diferentes subdominios, el conjunto de conceptos asociados a cada uno de ellos, y devuelve como resultado la información extraída de las diferentes paginas en formato XML.

El proceso se puede dividir en dos etapas: **extracción de información** y **exportación de información**.

El proceso de **extracción de información** se encarga de extraer la información definida por los conceptos.

En la etapa de **exportación de información** se transforma la información obtenida en el proceso anterior, a documentos XML.

Para la implementación de este módulo se reutilizaron varias estructuras de datos, así como algoritmos, implementados para el módulo de clasificación.

4.1.2.1. Proceso - Extracción de información

En esta etapa, como se dijo anteriormente, se realiza la extracción de información de las paginas según el resultado de la clasificación.
Para determinar que información se extrae de cada subdominio, se asignan conjuntos de conceptos a cada uno de ellos.
Para cada uno de estos conceptos se generan wrappers de extracción, que luego se ejecutan sobre las paginas pertenecientes a dicho subdominio.
Estos wrappers son los utilizados para realizar la extracción de información necesaria para la evaluación de los criterios de extracción.

Procedimiento: Extracción de información

Entrada:

- Resultado del proceso de clasificación
- Definición de los conceptos
- Relación entre conceptos y subdominios

Salida:

- Estructura que mantiene la información sobre la extracción para cada uno de los subdominios.

Seudocódigo:

```
Para cada subdominio s
        Para cada concepto c de s
                Si no existe wrapper w asociado a c
                        Genero wrapper a partir de los criterios de extracción de c
                Para cada pagina p de s
                        Ejecuto wrapper w sobre p
                        Filtro el resultado de la ejecución del wrapper w
                        Agrego resultado del filtro al resultado final
                Fin para
        Fin para
Fin para
```

4.1.2.2. Proceso - Exportación de información

Este proceso se encarga de transformar la información obtenida a partir del proceso de extracción de información en documentos XML. Este se puede dividir en dos partes: parseo de la información extraída a un objeto DOM y generación del documento XML. La información extraída es exportada a un único documento.

A continuación se presenta la especificación del documento XML generado por este proceso.

```
<ExtractionResult>
    <Concept name="nombre del concepto">
        <Criteria regexp="expresion regular principal">
            <Information>
                Información extraída
            </Information>
            ...
            <Information>
                Información extraída
            </Information>
        </Criteria>
    </Concept>
    …
    …

    <Concept name="nombre del concepto">
        <Criteria regexp="expresion regular principal">
            <Information>
                Información extraída
            </Information>
            ...
            <Information>
                Información extraída
            </Information>
        </Criteria>
    </Concept>
</ExtractionResult>
```

El elemento *ExtractionResult* es el elemento raíz del documento.
El elemento *Concept* contiene una secuencia de elementos *Criteria* que se corresponden con los criterios de extracción asociados a cada concepto. Este elemento tiene como atributo el nombre del concepto.
El elemento *Criteria* contiene una secuencia de elementos *Information* que mantienen la información extraída de la pagina. Este elemento tiene como atributo el nombre de la expresión regular principal del criterio de extracción.

Procedimiento: Parseo de información extraida

Entrada:

- Resultado del proceso de extracción de información.

Salida:

- Objeto DOM que almacena la información extraída por el ***proceso de extracción***

Seudocódigo:

Para cada elemento de la estructura *e* (resultado del proceso de extracción)
    Agrego la información extraída al objeto ***document***
Fin para

5. Prototipo

5.1. Descripción

En esta sección se presenta la arquitectura de un sistema que implementa las propuestas presentadas en las secciones *Clasificación* y *Extracción* descritas anteriormente. Este sistema propone un mecanismo que permite clasificar un conjunto de paginas web pertenecientes a un mismo dominio (proceso de clasificación) en diferentes subdominios, y según el resultado de la clasificación permite extraer la información que se considera relevante (proceso de extracción). Dicha información es retornada como documentos XML.

El prototipo [LPR01] desarrollado para el proyecto " Web Classification Wrappers " fue implementado utilizando el lenguaje java (jdk versión 1.3.1). Las siguientes características del lenguaje motivaron para que fuera elegido para la implementación de dicho prototipo:

- multiplataforma
- facilidad del lenguaje para el manejo de la herramienta W4F
- conocimiento previo del lenguaje

Para almacenar la información utilizada por los procesos de clasificación y extracción, se utiliza una base de datos. A continuación se presenta el Modelo Entidad-Relación (MER) utilizado y en el Apéndice F se presenta la documentación de dicha base.

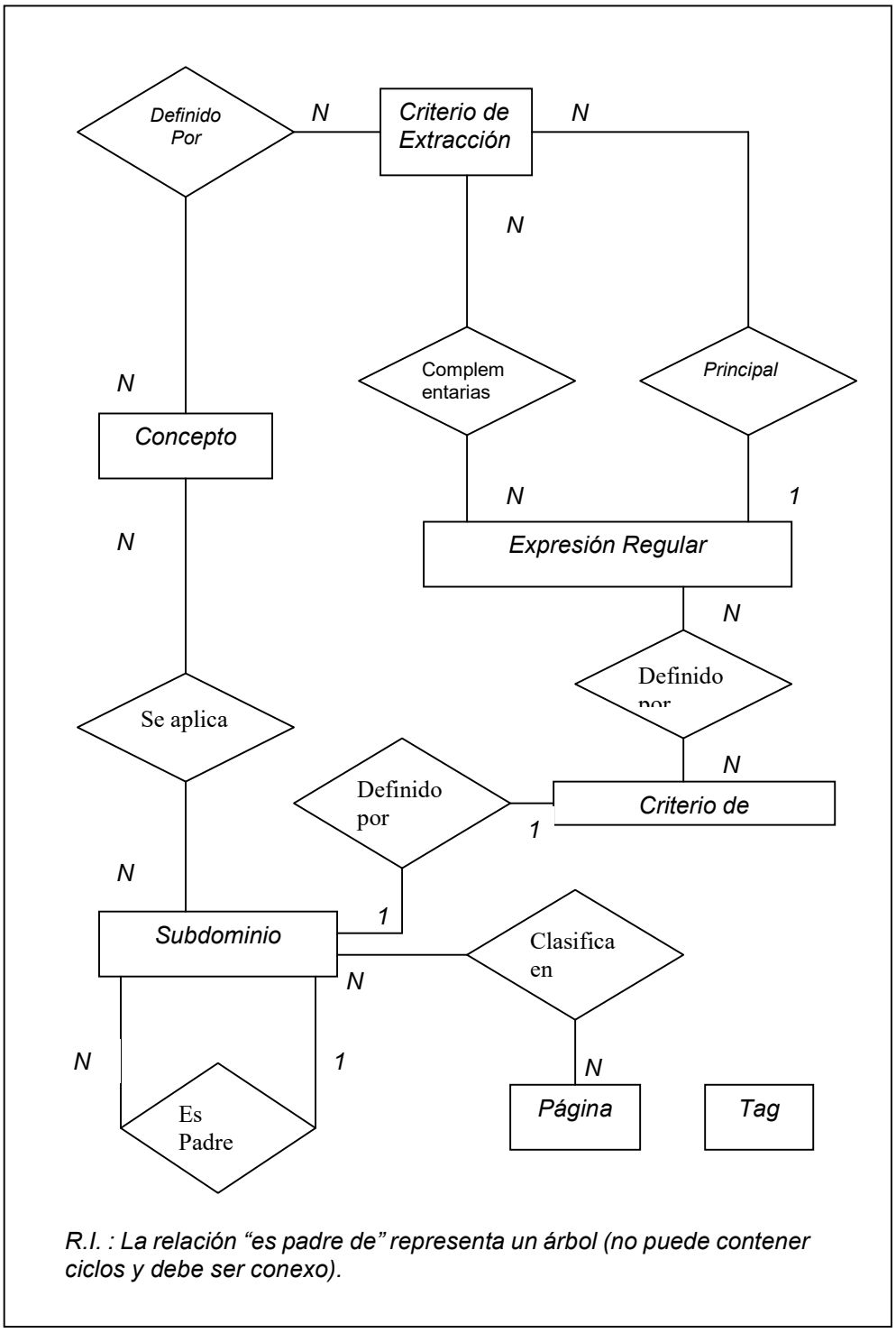

Definido Por
N
Criterio de Extracción
N
N
Complem entarias
Principal
N
Concepto
N
1
N
Expresión Regular
N
Definido por
Se aplica
N
Definido por
Criterio de
1
N
1
Subdominio
Clasifica en
N
N
1
N
Es Padre
Página
Tag
R.I. : La relación “es padre de” representa un árbol (no puede contener ciclos y debe ser conexo).

Fig. 6 - MER

La entidad “Tag” representa los tags del lenguaje HTML, que son considerados en los procesos.
La entidad “Subdominio representa los subdominios en los que se quiere clasificar las páginas.
La auto-relación “Es Padre de” representa el árbol de clasificación.
La entidad “Página” representa las páginas que se utilizan en los procesos. La relación “Clasifica en” entre “Página” y “Subdominio” representa el resultado del proceso de clasificación. Una página puede estar en ningún o varios subdominios y un subdominio puede tener ninguna o varias páginas.
La entidad “Criterios de Clasificación” contiene la información necesaria para determinar si una página pertenece o no a un subdominio. Cada subdominio se asocia a un solo criterio de clasificación, lo cual se refleja en la relación “Definido por” con la entidad “Subdominio”.
La entidad “Expresión Regular” representa las expresiones regulares que se utilizan en los procesos. La relación “Definido por” con la entidad “Criterio de Clasificación” representa cuales son las expresiones regulares que se utilizan para cada criterio de clasificación.
Los conceptos que se van a extraer de un determinado subdominio se representan con la entidad “Conceptos” y la relación “Se aplica”. Dicha relación refleja que un subdominio puede tener asociado varios conceptos y un concepto puede tener asociado varios subdominios.
La entidad “Criterio de extracción” contiene la información necesaria para determinar que se va a extraer. La relación “Definido por” con la entidad “Conceptos” representa cuales son los conceptos que se utilizan para extraer y la relación “Complementarias” indica cuales expresiones regulares se utilizan para el proceso de extracción.
La relación “Principal” entre las entidades “Criterio de extracción” y “Expresión Regular” representa el proceso de extracción. Un criterio de extracción está asociado a una expresión regular principal, que busca en las páginas que clasificaron en el determinado dominio. El proceso de extracción va a extraer la información donde se encuentre esta expresión regular, acompañada por otras expresiones regulares complementarias.

5.2. Limitaciones

El prototipo tiene las siguientes limitaciones:

- Interface. La interfaz que ofrece el prototipo brinda las funcionalidades básicas para su ejecución. No se desarrolla una interfaz amigable al usuario, debido a que este proyecto no es un producto final.

- Los procesos no detectan automáticamente si la información almacenada en la base de datos cambio, por lo tanto cada vez que se modifique dicha información se deben borrar los wrappers para que sean regenerados a partir de la nueva información.

La especificación del diseño y la implementación del prototipo se encuentra disponible en el Apéndice D.

6. Evolución

Dada la volatilidad de la web, es fundamental poseer un mecanismo para detectar y manejar los cambios en la fuente de datos.

Dependiendo del tipo de cambio que sufren las paginas, sé debería ejecutar nuevamente el proceso de clasificación y/o el proceso de extracción.

En este capitulo analizamos primero los posibles tipos de cambios que pueden sufrir las páginas, anotamos una reseña de herramientas para detección de cambios y evaluamos las posibilidades de aplicar reingeniería sobre los procesos de clasificación y extracción anteriores para gerenciar el mantenimiento de los cambios.

6.1.1. ANALISIS DE LOS TIPOS DE CAMBIOS

- Multimedia
  Cambios en las imágenes, sonido, vídeo, applet, o cualquier otra información que no sea texto.

- Reestructuración de la página

  - Diseño
    Este tipo de cambios se refiere a modificaciones en la apariencia de la página, por ejemplo alteración de fuentes (color, tamaño, etc.), transformación de tablas en listas, etc.

  - Movimiento dentro de la página
    Se refiere a los cambios en los que se mueve el contenido de un tag a otro. En este tipo de cambio se mantiene la misma estructura y el mismo contenido, pero este último se cambia de lugar.

- Contenido
  Estos son los cambios en el texto, sin considerar Tags HTML.

- Metadata
  Cambios en la metadata de la página.

6.1.2. RELEVANCIA DEL CAMBIO

La clasificación, como ya se mencionó, es un proceso costoso, por lo tanto el objetivo es, reclasificar la menor cantidad de veces posible. Se sabe que existen cambios en las páginas que no afectan el resultado de la clasificación. Para manejar esta situación se introduce el concepto de *relevancia de un cambio*.
La *relevancia de un cambio* es la probabilidad de que este afecte el resultado de la clasificación de la página.
Cuanto mayor probabilidad tenga un cambio en la página de afectar el resultado de la clasificación, mayor relevancia tendrá.

Un cambio sin probabilidades de influir en la clasificación es irrelevante.
La importancia de este concepto es que, ante un cambio de esta índole no es necesario reclasificar la página.

Frente a esta situación, se plantea el problema de cuando reclasificar una página. Para aclarar esta situación se estudia la relevancia de cada tipo de cambio.

- Multimedia
  Estos cambios son irrelevantes, ya que la clasificación se basa en el texto.

- Reestructuración de la página

  - Diseño
    Los cambios en el diseño de la página son relevantes si se cambian los tags que se consideran en el proceso de clasificación.

    Aunque un cambio dentro de los tags considerados no necesariamente afecta el resultado de la clasificación. Un ejemplo de esto puede ser el cambio de texto de una lista a una tabla. En un contexto en el cual tablas y listas tienen asignado el mismo peso, este cambio tampoco debe provocar una reclasificación.

  - Movimiento dentro de la página
    Las expresiones regulares tienen asignado un determinado peso en la página, según el lugar de esta en que se encuentren. Al modificar la ubicación de una expresión regular podría también cambiarse el peso asignado a ella, afectando de esta forma el resultado de la clasificación.

    En este punto se presentan dificultades como por ejemplo, detección de permutaciones en las columnas de una tabla. Es decir, se mueven los datos de una columna a otra, si ambas columnas tienen los mismos atributos asociados, este cambio no debe provocar una reclasificación.

- Contenido
  La relevancia de este tipo de cambios depende de que, se modifiquen en la página expresiones regulares que son consideradas en el proceso de clasificación.

- Metadata
  Estos cambios son irrelevantes, ya que la clasificación se basa en el texto.

6.1.3. HERRAMIENTAS DE DETECCIÓN DE CAMBIOS

A continuación se mencionan algunas de las principales herramientas consideradas en el relevamiento realizado [8] :

- **Mind-It**

Esta herramienta monitorea las páginas web registradas y notifica vía e-mail cuando una página ha sido modificada, esta tarea la realiza una componente llamada **URL-minder**, luego otra componente llamada **Highlights** resalta los cambios de la página. **URL-minder** está disponible on-line para monitorear páginas de Internet. [9]

- **HtmlDiff**

Esta es una herramienta que compara dos páginas html y el resultado es una nueva página con los cambios resaltados. Permite realizar algún seteo como el de ignorar blancos e indicar si es case sensitive o no. Disponible on-line para comparar páginas de Internet. [10]

- **TopBlend**

Es una nueva implementación en Java de HtmlDiff que destaca un nuevo algoritmo de comparación y la posibilidad de ver los cambios uno por uno. [11]

- **WebGUIDE (Web Graphical User Interface to a Difference Engine)**

Esta es una herramienta para detectar cambios en páginas web y en la estructura de los sitios, es decir en los links existentes, soporta comparación recursiva de documentos, y los usuarios pueden explorar las diferencias entre páginas respecto a dos fechas. Las diferencias entre páginas son resumidas automáticamente en una nueva página HTML y las diferencias en la estructura de links es mostrada mediante una representación gráfica. [12]

- **WebCQ (CONTINUAL QUERY)**

WebCQ está diseñado para detectar cambios en páginas Web eficientemente y proveer una notificación personalizada de que y como han cambiado las páginas de interés. Los requerimientos de monitoreo de cambios indicados por los usuarios son modelados como consultas continuas sobre la Web. Puede monitorear varios tipos de cambios en páginas estáticas o dinámicas y ofrece la posibilidad de personalizar la notificación de los cambios y resumir los cambios que se quieren monitorear. [13][14]

- **NiagaraCQ**

El objetivo de este proyecto es permitir la ejecución de consultas estilo base de datos sobre archivos XML. Provee mecanismos para encontrar los archivos XML relevantes a la consulta dada y tratar con fuentes de datos remotas que proveen un acceso a los datos impredecible, con fuentes que son un stream infinito o ambas. [15][16][17]

- **WebBeholder**

El objetivo de este proyecto es brindar un servicio para encontrar y visualizar cambios sobre la web en base a una estructura de una comunidad de agentes cooperativos. Varios agentes y componentes en la comunidad interactúan con otros para lograr la meta de los usuarios del sistema. El sistema consiste de un agente proveedor de servicio que observa y detecta los cambios sobre la Web, un número de agentes que representan cada usuario y un número de mediadores para negociar con el agente proveedor de servicio por los requerimientos de los agentes personales. [18]

- **HTML Compare**

Esta herramienta compara dos versiones de una página html y crea una nueva resaltando las diferencias. También puede comparar dos directorios en cuyo caso crea una página en la que indica los archivos nuevos, los borrados y para los modificados tiene un link a la versión vieja y otro a la nueva, si se trata de un archivo html realiza la comparación de los mismos colocando un link a la página resultado. Los archivos que se van a comparar deben estar en el PC o en una red local. [19]

- **URL y Warning**

No muestra estructura html. Se registra URL y compara versiones de la misma página. Versión on-line, se puede bajar un cliente. No permite seteos en que cambios monitorear. [20]

- **WebSecretary**

No maneja páginas HTML. En base a fecha de modificación o algún checksum, manda mail avisando que hubo cambio, pero no cual. [21]

- **ARANEUS**

Herramienta que permite manejar información de páginas Web como de Base de Datos pero aun no maneja la parte de evolución.
El objetivo de este proyecto es manejar la información de bases de datos y de la Web en el mismo estilo de base de datos. El enfoque es generalizar la idea de vistas a la Web, como herramienta de reestructuración e integración. El sistema es diseñado para soportar varias clases de aplicaciones como ser: (i) el acceso de alto nivel a los datos en la web, (ii) el diseño, implementación y mantenimiento de sitios Web, (iii) aplicaciones cooperativas sobre la Web. Las vistas de sitios web externos son materializadas localmente. [22]-[33]

- **TracerLock**

Esta herramienta monitorea las páginas de los resultados del buscador de AltaVista a una consulta dada y notifica cuando aparecen nuevos resultados, además proporciona los links más relevantes a la información consultada. [34]

- **The Informant**

Esta herramientas tiene dos posibilidades:

1. Monitorear las páginas de los resultados de AltaVista, Lycos, Excite e Infoseek permitiendo especificar hasta tres consultas y el período de tiempo para realizar el chequeo. Monitorea las 10 páginas de resultados más relevantes y notifica vía e-mail si alguna nueva página se encuentra en las 10 páginas de resultado o si las páginas existentes han sido modificadas.
2. Monitorea páginas web, se pueden indicar hasta 4 páginas y el período de tiempo para realizar el chequeo, notifica vía e-mail cuando alguna de ellas cambia. [35]

6.1.4. Resumen de la Comparación de herramientas

	Notificación	Comparación	On-line
Mind-it	X	X	X
HTML-Diff		X	X
TopBlend		X	X
WebGuide		X	X
WebCQ	X	X	X
HTML Compare	X	X	
URL Warning		X	
Web Secretary	X		X
The Informant	X		X

Fig. 6 - Cuadro comparativo

En la figura 6, se visualiza un cuadro comparativo de las herramientas de detección de cambios relevadas. Con X se indica si dichas herramientas envían una notificación, comparan diferencias entre dos versiones de un documento y si es posible aplicar estas herramientas sobre páginas on-line.

6.1.5. Conclusion

La cualidad más importante que buscamos a la hora de elegir una herramienta de detección de cambios, es que envíen una notificación. Al tener la certeza de que cambió una página, tenemos las siguientes opciones:

- Correr los procesos de clasificación y extracción nuevamente.
- Correr solamente el proceso de extracción.
- No hacer nada

La opción a elegir va a depender del tipo de cambio que ocurrió (ver sección 6.2). Las herramientas que realizan comparaciones nos pueden ayudar en esta toma de decisión.

La propiedad de aplicar las herramientas a páginas on-line es importante para nuestro proyecto, ya que pretendemos que se pueda aplicar tanto localmente como en la web.

Del cuadro comparativo de la Figura 6, observamos que de las herramientas relevadas, las que cumplen estas tres propiedades deseables son Mind-it y WebCQ. Ambas herramientas notifican los cambios ocurridos, pero WebCQ también ofrece la posibilidad de personalizar que cambios monitorear, lo cual es mejor desde el punto de vista de la performance.

En conclusión, la herramienta que más se ajusta a los requerimientos de nuestro proyecto es WebCQ.

7. Conclusiones

7.1. CONCLUSIÓN

Se completo exitosamente el desarrollo del prototipo LPR01, que implementa los módulos de clasificación y extracción. Se realizo el análisis del módulo de evolución, quedando pendiente el diseño y la implementación.

Puede destacarse como contribución importante, las estructuras definidas como mecanismo de especificación de la información deseada por el usuario. Dichas estructuras son el “árbol de clasificación”, y el “concepto”, definidos respectivamente en la sección de clasificación y extracción.

Otro aporte de interés, es la implementación de clases java para encapsular el manejo de W4F, que permiten la generación de los wrappers necesarios para extraer información de los documentos HTML. Esto es de gran utilidad, ya que la herramienta se basa en el análisis del contenido de las páginas.

En las VII Jornadas de Informática e Investigación Operativa realizadas en el Instituto de Computación, se realizó una presentación del proyecto. En dicha presentación se exhibió el problema, los objetivos, la arquitectura, el estado del proyecto y una demo del prototipo.

7.2. EVALUACIÓN

El prototipo LPR01 se implementó en Java. Para un conjunto de pruebas realizadas la generación de los documentos XML resultó satisfactoria, sin embargo la herramienta necesita ser testeada con distintos dominios. Como posible extensión se propone trabajar sobre sitios completos, siguiendo links.

Una característica a resaltar de dicho prototipo es que puede ser utilizado tanto localmente, como directamente en la web.

El Apéndice E contiene los Juegos de prueba.

7.3. TRABAJO FUTURO

Futuros desarrollos incluyen:

- La incorporación de la herramienta Tidy [36], que pasa documentos HTML a XML, solucionando las dificultades que puede acarrear trabajar sobre documentos HTML.
- Elección de una buena herramienta para la parte de Evolución, que sea en lo posible de distribución gratuita y que su utilización sea fácil de automatizar.
- Incorporación de la herramienta WordNet [43], que permite aumentar el vocabulario necesario para definir los subdominios.

Apéndice A - HTML (HyperText Markup Language)

Es un lenguaje de marcas que permite escribir documentos. Dichos documentos tienen embebidas marcas en el texto, que definen un significado. Estas marcas se leen con un browser que representa el contenido en forma adecuada. A estas marcas se le denominan tags. [37]

1.1 Marcas (Tags)

Las marcas delimitan elementos de un documento como cabeceras, párrafos, etc., y se utilizan para dar un tratamiento diferente al texto que se encuentra entre las marcas.
En HTML las marcas se delimitan con los signos < (inferior a) y > (superior a). Un texto marcado tendrá por tanto este aspecto:

> ... texto normal <marca> texto afectado por la marca </marca>
> resto del texto ...

Ejemplo: Resaltar un texto con negrita

Para ello se emplea la marca <B> y queda de la siguiente forma:

> ... texto normal <B> texto en negrita </B> resto del texto ...

Atributos de las marcas

Algunas marcas pueden admitir atributos, pudiendo tener cada uno de estos atributos un valor.

Estructura de los documentos

<HTML>
<HEAD>
<TITLE> Bienvenido a nuestra página de Taller V </TITLE>
</HEAD>
Cuerpo del documento
...
</HTML>

La cabecera se emplea para facilitar información acerca del documento y está delimitada por <HEAD> y </HEAD>. Dentro de la cabecera podemos destacar el título que indica el nombre del documento <TITLE> y </TITLE>.
El resto del documento residirá entre las marcas <BODY> y </BODY>.

A continuación describiremos algunos elementos que pueden aparecer dentro del cuerpo:

Encabezado – los encabezados se emplean para dividir los documentos en secciones o mas concretamente para marcar los títulos de esas secciones. Las marcas son del tipo <H#> título </H#>, donde # puede ser un número cualquiera entre 1 y 6.

<H1>Tamaño mayor </H1>
<H6>Tamaño menor </H6>

Definición de bloques – para definir y separar bloques de texto se emplean una serie de marcas que definen párrafos, texto preformateado o bloques con significado especial como direcciones o citas.

Marcas de bloques:

- <P> se utiliza para separar párrafos. Dado que para el HTML todo el texto es continuo, necesitamos algún mecanismo para indicar el principio y el fin de un párrafo. Las marcas inicial y final son <P> y </P>

- <PRE> El texto insertado entre las marcas <PRE> y </PRE> será visualizado por el browser, respetando el formato con el que fue escrito en el fichero fuente HTML.

- <ADDRESS> empleada para indicar que un texto representa una dirección o una firma. Generalmente se activa en cursiva y suele estar tabulado.

- <BLOCKQUOTE> Se suele representar con tabulaciones a la izquierda y derecha y en cursiva. En sistemas que no permiten representar en cursiva se puede emplear algún tipo de símbolo al principio de las líneas.

-
 Este elemento solo tiene marca inicial e indica un salto de línea.

- <HR> Solo tiene marca inicial y se emplea para representar una línea horizontal.

Comentarios - Todo texto que empieze por <!...comentario...> será ignorado por el browser, y por lo tanto no será visible. Esto sirve al autor del documento para comentar su fichero fuente.

1.2 Problemas que tiene HTML

Define más la presentación que el contenido.
No es fácilmente procesable por “máquinas”.
Problemas de internacionalización.
Su estructura es “caótica”.
Su interpretación es ambigua según el SW utilizado.
Solo tiene un uso: páginas web.

Apéndice B - Reseña sobre W4F

World Wide Web Wrapper Factory (W4F) es una herramienta para construir componentes de software que integra publicaciones Web dentro de aplicaciones. Esto es necesario para aplicaciones tradicionales de toma de decisiones y para aplicaciones Web, que es un área nueva y extremadamente activa.
W4F ofrece a los desarrolladores una herramienta rápida para escribir dichos componentes especializados denominados wrappers.
W4F divide el proceso de desarrollo del wrapper en una fase de extracción y otra de mapeo. Ambas son guiadas por la estructura de los datos, ambas involucran programación con descripciones de alto nivel y ambas son semi-automatizadas por interacción con wizards diseñados cuidadosamente. Por su fácil reutilización, todo esto hace que el proceso de escritura del wrapper sea tanto rápido como flexible. [6][7]

1.1 Cómo escribir un wrapper

Lo primero que se debe hacer es escribir un archivo de descripción. Un archivo de descripción tiene la extensión .w4f y contiene toda la información para construir un wrapper.
Dicho archivo se divide en varias secciones. Cada sección comienza con un nombre de sección seguida de "{" y termina con "}". Hay cinco secciones predefinidas, pero se pueden definir secciones.
Las secciones predefinidas son:

OPTIONS – Se pueden especificar aquí varias opciones para controlar el wrapper, como el proxy, la implementación del DOM a usar, tiempo de respuesta, etc.

SCHEMA – Aquí es donde se especifica como la información extraída va a ser mapeada en objetos Java. Es el equivalente a la declaración de variables en lenguajes de programación.

EXTRACTION_RULES – Aquí se escriben las reglas de extracción, usando HEL (HTML Extraction Language). Ellas especifican como navegar el árbol HTML para extraer información.

RETRIEVAL_RULES – Aquí se especifica como retornar la información, de la web, o de un sistema de archivos local.

JAVACODE – Todo lo que se escriba en esta sección va a ser insertado "como es" en el código Java generado. Por ejemplo, es útil para escribir un método main.

Ninguna sección es obligatoria. Cualquier combinación de secciones válidas va a generar algún código Java válido. Pero el wrapper no va a ser muy útil, si no tiene definido una sección OPTIONS.
El orden de las diferentes secciones que son especificadas debe ser el siguiente:

- OPTIONS
- SCHEMA
- EXTRACTION_RULES
- RETRIEVAL_RULES
- Cualquier otra sección definida por el usuario en cualquier orden
- JAVACODE

1.2 Compilación del wrapper

Una vez que se escribe el wrapper, se puede compilar. Por ejemplo, si se guarda el wrapper en el archivo MyWrapper.w4f, para compilarlo basta con tipear w4f MyWrapper.w4f
La compilación va a crear tres archivos:

- MyWraper.Java – contiene el código Java generado automáticamente.
- MyWraper.class – es la clase compilada para el wrapper.
- MyWraper_get.class – es una clase interna correspondiente al método retrieval.

1.3 Escritura de Retrieval Rules

Una retrieval rule especifica como va a ser retornado el documento. Como hay varias maneras de retornar un solo documento (por ejemplo, desde un file system o sobre la web), se pueden especificar tantas reglas como se quieran en esta sección.
La versión corriente de W4F soporta cuatro métodos retrieval:

- File GET
- HTTP HEAD
- HTTP GET
- HTTP POST

Ejemplo:

```
RETRIEVAL_RULES
{
getAltavista ( ) {
                    METHOD: GET ;
                    URL: http://www.altavista.com ;
                  }
}
```

1.4 Escritura de Retrieval Rules

El corazón del wrapper es probablemente esta sección. Aquí es donde se usa el poderoso lenguaje de extracción de W4F para extraer información de páginas HTML.
Esta sección va a tener la siguiente estructura en el archivo:

```
EXTRACTION_RULES
{
...
}
```

Dentro de esta sección se pueden escribir tantas reglas de extracción como se quieran.
La sintaxis para definir una regla de extracción es:

<rulename> = <extraction rule>

la convención para una rulename sigue la convención de Java para nombres de variables.
La extraction rule se define utilizando HEL (HTML Extraction Language).

1.5 Utilización del Schema

Se puede ver la sección Schema del archivo de descripción del wrapper como la declaración de variables en lenguajes de programación. Aquí se declaran variables como se hace en Java. Luego, se escribe una regla de extracción con el mismo nombre para cada variable declarada.

Ejemplo:

SCHEMA

```
{
   String title;
   Int http_code;
}

EXTRACTION_RULES
{
  title = html.head.title.txt;
  http_code = html.getAttr(http_code);
}
```

1.6 La sección Options

Esta sección se utiliza para definir varios parámetros que controlan el comportamiento de W4F:

- usar rasgos de Java OO
- implementación del DOM
- usar un proxy
- chequear que se tiene la página esperada
- etc.

Apéndice C - XML

XML es un subconjunto de SGML (Standard Generalised Mark-up Language), simplificado y adaptado a Internet. No es, como su nombre podría sugerir, un lenguaje de marcado. XML es un meta-lenguaje que nos permite definir lenguajes de marcado adecuados a usos determinados. Además, XML es un estándar internacionalmente reconocido, no pertenece a ninguna compañía y su utilización es libre. [38]

1.1 Ventajas de XML

Su estructura hace que sea fácilmente entendible tanto por humanos como por software.
Separa radicalmente la información o el contenido de su presentación o formato.
Diseñado para ser utilizado en cualquier lenguaje o alfabeto.
Permite poderosas técnicas de extracción de información y data-mining.
Las estrictas reglas para la composición de un documento XML permiten su fácil análisis sintáctico.

1.2 XML “bien-formado”

Se dice que un documento XML es “bien formado” cuando cumple una serie de reglas descritas en la especificación oficial de XML v1.0
Esta propiedad de los documentos XML es útil cuando se extrae información en base a la estructura de la página web. [42]

2 Tidy

Herramienta que convierte páginas en formato HTML a XML. [36]
Corrige error tales como: falta de tags de terminación, tags de terminación en orden incorrecto, mezcla de tags, etc.

3 Regular Expressions

3.1 Cuantificadores

El contenido de una expresión es una combinación de caracteres alfanuméricos y meta caracteres. Un carácter alfanumérico es tanto una letra del alfabeto

abc

o un número

123
Actualmente en el mundo de las expresiones regulares, cualquier caracter que no sea un meta carácter va a realizar match consigo mismo (usualmente denominado caracter literal), aunque la mayoría del tiempo nos preocuparemos por los caracteres alfanuméricos. Un caracter especial es la retrobarra \, esta convierte cualquier metacaracter en caracter literal, y caracteres alfanuméricos en una especie de metacaracter o secuencia. Los metacaracteres son:

\ | ( ) [{ ^ $ * + ? . < >

el carácter de puntuación o punto, este no va a realizar match con un punto en una linea. Es un meta carácter especial que matchea cualquier carácter. Se usa cuando se quiere encontrar el fin de la linea o el decimal en un número de punto flotante. Por ej., tomemos la expresión
1.23
va a matchear el número 1.23 en un texto , pero también lo siguiente:

1x23
1 23
1-23

Para que solo matchee la expresión de número flotante tenemos que poner
1\.23

dos metacaracteres recurrentes son
* y +

se denominan cuantificadores y le indica a la máquina que busque varias ocurrencias de un carácter. El carácter * matchea cero o más ocurrencias de un carácter en una fila, el carácter + es similar, pero matchea uno o mas.

En las expresiones regulares tenemos la posiblididad de matchear lo que se denomina el string vacío, que es un string con tamaño cero.

El metacaracter ? indica a la máquina realizar o no el matching (cero o uno). Por ejemplo la expresión:
cows?
va a matchear cualquiera de estas líneas:
cow
cows

{n,m}

la n y la m son el tamaño mínimo y máximo del cuantificador, respectivamente. Por ejemplo

{1,5}
significa matchear uno o hasta cinco caracteres. Se puede saltear m para permitir matcheo infinito:
{1,}
que machea uno o mas caracteres. Esto es exactamente lo que hace el carácter +.
La última cosa que se puede hacer con el cuantificador es no poner la coma,
{5}
que significa machear 5 caracteres, ni mas ni menos.

3.2 Afirmaciones (assertions)

El siguiente tipo de metacaracteres son afirmaciones, van a matchear si un afirmación dada es verdadera. El primer par de afirmaciones son
^ y $
que matchean el principio de una linea y el fin de la línea. Notar que algunas expresiones regulares permiten cambiar su comportamiento, por lo que matchearían el principio y el final del texto. Estas afirmaciones también matchean un string de largo cero, o en otras palabras matchean una posición.
Por ejemplo si escribimos la expresión:
^The
va a matchear cualquier linea que comience con la palabra The.

< y >
matchea al principio y el final de la palabra. Por ejemplo:

cow
va a matchear con cualquiera de las siguientes palabras

cow
coward
cowage
cowboy
cowl

3.3 Grupos y alternanciones

Los cuantificadores solo trabajan sobre los caracteres en la izquierda
Se pueden formar grupos, o subexpresiones , usando los paréntesis de apertura y cierre:

(y)

El (empieza la subexpresión y el) la termina. También es posible tener una o mas subexpresiones dentro de una subexpresión. La subexpresión va a matchear si el contenido matchea. Mezclando esto con cuantificadores y afirmaciones podemos hacer:

(?ho)+
que matchea todas las siguientes lineas

Ho
ho ho
ho ho ho
hohoho
Otro uso de las subexpresiones es extraer una parte del matcheo si matchea, esto se usa a menudo en la conjunción con secuencias.

Las alternaciones permiten matchear entre muchas palabras, el carácter de alternación es |

Un simple uso es:

Bill|Linus|Steve|Larry
Va a matchear tanto Bill, Linus, Steve o Larry, y mexclando esto con subexpresiones y cuantificadores podemos hacer:

cow(ard|age|boy|l)?
que matchea cualquiera de las siguientes palabras pero ninguna otra
cow
coward
cowage
cowboy
cowl

3.4 Secuencias

Los caracteres de secuencia son
[y]
cualquier carácter puesto entre la secuencia de paréntesis rectos es tratada como un carácter literal, incluso metacaracteres. El único carácter especial es el – que denota el rango de caracteres y el ^ que es usado para negar una secuencia. La secuencia es similar a la alternación, la similitud es que solo uno de los items de la lista va a matchear. Por ejemplo

[a-z]
va a matchear cualquier carácter minúscula del alfabeto. Otra secuencia común es
[a-zA-Z0-9]
que matchea cualquier carácter minúscula, mayúscula, asi como también números. Las secuencias también se combinan con cuantificadores y afirmaciones para producir búsquedas elaboradas. Por ejemplo

<[a-zA-Z]+>
matchea todas las palabras. Esto va a matchear

cow
Linus
regular
expresión

Pero no va a matchear

200
x-files
C++

3.5 Comodines

*.jpg
matchea cualquier texto que termina con.jpg. también se pueden especificar paréntesis con caracteres, por ejemplo

*.[ch]pp
matchea cualquier texto que termina con .cpp o .hpp.

Apéndice D - PROTOTIPO

1.1 Diseño

1.1.1 Diagrama - Proceso de clasificación

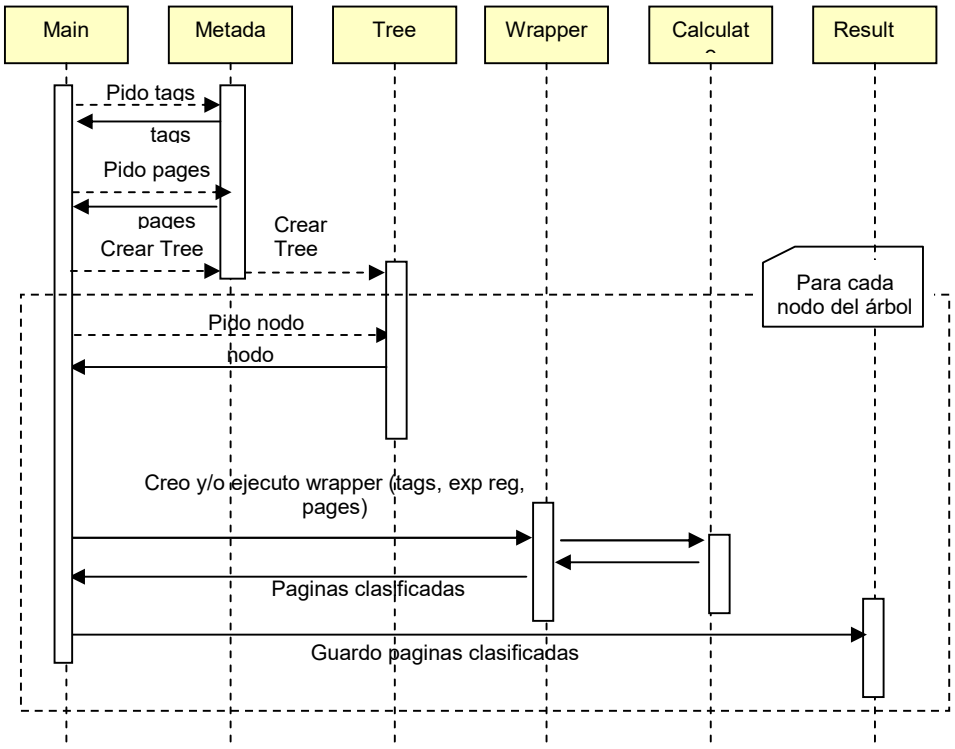


Fig – Proceso de clasificación

El diagrama de la figura muestra los objetos que intervienen en el proceso de clasificación y su interacción.

1.1.2 Diagrama - Proceso de extracción

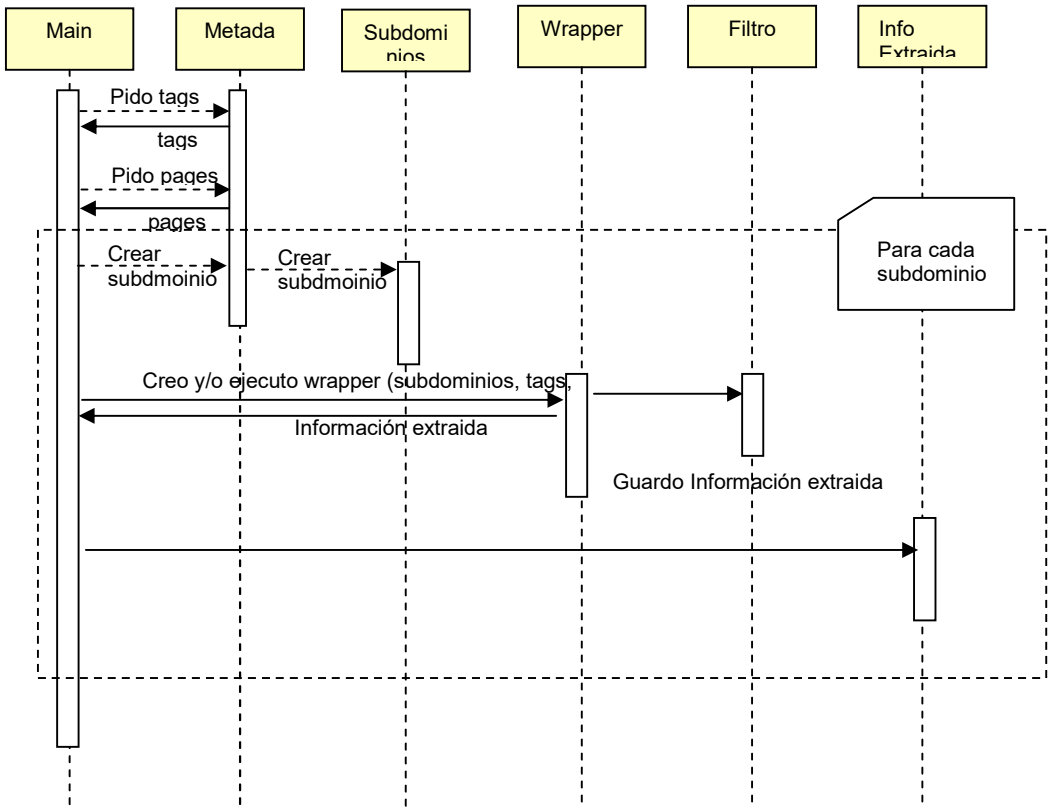


Fig – Proceso de extracción

El diagrama de la figura muestra los objetos que intervienen en el proceso de extracción y su interacción.

1.2 Implementación

La implementación de este sistema consta de dos módulos principales: proceso de clasificación y proceso de extracción. En la sección 3.1.5 *Proceso de Clasificación* y 4.1.2 *Proceso de Extracción* fueron descriptos los procedimientos principales de dichos procesos.

1.2.1 Organización de las clases

A continuación se presenta como fueron organizadas las clases dentro del proyecto:
Este mantiene la siguiente estructura de paquetes:

- ***classification***: Este paquete contiene las clases que implementan el proceso de clasificación.
- ***extraction***: Este paquete contiene las clases que implementan el proceso de extracción
- ***database***: Este paquete contiene las clases que recuperar y almacenar la información en las base de datos
- ***wrapper:*** Este paquete contiene el conjunto de clases que encapsulan la funcionalidad que ofrece la herramienta W4F.
- ***util***: Este paquete contiene las clases de uso común entre los diferentes procesos

Por información mas detallada con respecto a los métodos que ofrece cada clase referirse al archivo *javadoc.html* encontrado en el CD de instalación.

Apéndice E - JUEGO DE PRUEBAS

1.1 Juego de prueba 1 - Aspirina

El dominio de este juego de prueba es la Aspirina.

1.1.1 Páginas

Se realizó una búsqueda con el programa Copérnico y se seleccionó un subconjunto de páginas del resultado de la búsqueda.
Las URL's de las páginas seleccionadas son:

http://www.elmundosalud.com/elmundosalud/especiales/aspirina/aspirina.html
http://salud.enlaweb.cl/pages/aAspirin.htm
http://www2.el-mundo.es/salud/310/24N0114.html
http://www.tuotromedico.com/temas/previene_aspirina_infarto.htm
http://www.angelfire.com/biz2/vitaminas/
http://www.bayer.cl/productos/salud/listaprecios.htm
http://www.etsit.upm.es/~teatro/obras/aspirina.html
http://www.infodoctor.org/bandolera/b92s-3.html
http://www.euros.net/estepona/aspirina/index.html
http://www.ofermed.com/
http://www.el-mundo.es/salud/278/29N0122.html
http://www.bayer.es/medicinas/aspirina/aspmain/xxaspirina.html
http://www.coresalud.com/Divulg/aspirina.htm
http://www.mty.itesm.mx/dae/dsa/aspirina.htm
http://neurologia.rediris.es/congreso-1/conferencias/h-general-1.html
http://www.terra.com.hn/salud/articulo/html/sal670.htm
http://www.mipediatra.com.mx/infantil/aspirina.htm
http://www.unav.es/cun/noticias/aspirina.html
http://www.clarin.com.ar/diario/97-04-13/e-04201d.htm

1.1.2 Subdomios

En la Figura E1, se visualizan los subdominios elegidos para el dominio Aspirina. Estos son: con relación, ventas, tratamiento, descripción, artículo, investigación, efectos secundarios.

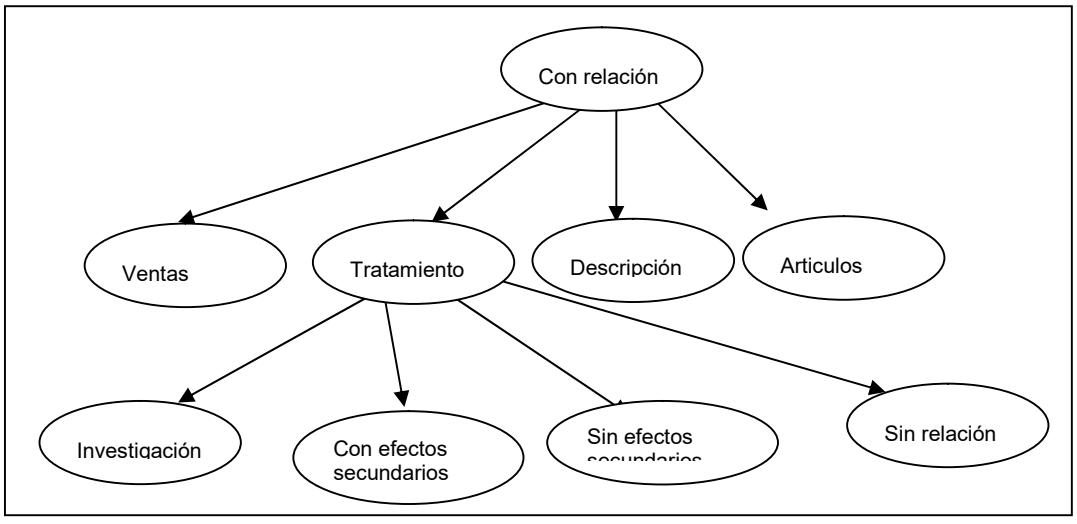


Fig. E1 – Subdominios para Aspirina

1.1.3 Expresiones Regulares

Las expresiones regulares se encuentran almacenadas en la tabla reg_exp de la base de datos MedicamRE. A modo de ejemplo, las expresiones regulares asociadas al subdominio investigación son:

reg_expresion	set_id	weigth
([Aa]nti)	9	1
([Cc]ientífico)(|s)	9	1
([Ee]studi)(ado|e|o|os)	9	1
(acción)	9	1
(ácido acetil salicílico)	9	2
(agente)	9	1
(descubrieron)	9	1
(efecto)	9	1
(evitar)	9	1
(experimento)(|s)	9	1
(fármaco)(|s)	9	1
(investiga)(r|ción|dor|dores|ciones)	9	1
(poder)	9	1
(preven)(ción|ir|tiva|tivo)	9	1
(previene)(|n)	9	1
(protege)	9	1
(prueba)(|s)	9	1
(publica)(ción|ciones|do)	9	1
(reduce)	9	1
(resultado)(|s)	9	1
(riesgo)	9	1

1.1.4 Conceptos

En este juego de prueba utilizamos el concepto “Precio de Venta”

1.1.5 Criterio de extracción

Se extrae información de las páginas que clasificaron en el subdominio “ventas”, relacionadas con la expresión regular principal ([Tt][Aa][Bb])(|[Ss]|[Ll][Ee][Tt][Aa][Ss]) y con las expresiones regulares secundarias (\$) y ([Aa][Ss][Pp][Ii][Rr][Ii][Nn][Aa]).

1.2 Juego de prueba 2 – Libros Informática

El dominio de este juego de prueba es libros de informática.

1.2.1 Páginas

Se realizaron búsquedas con el motores de búsqueda y se seleccionó un subconjunto de páginas del resultado de la búsqueda.
Las URL's de las páginas seleccionadas son:

http://www.cyberlibro.com/busqueda_avanzada2.asp?menu=1&buscar=Lenguajes
http://libros2.elcorteingles.es/secciones/temas/tema.asp?CODITEMA=30&DNOMTEMA=Inform%E1tica
http://www.tizaymouse.com.ar/Libros_IE.htm
http://www.anayamultimedia.es/cgigeneral/novedades.pl?id_sello_editorial_web=23
http://www.precioalavista.com/c317.html
http://www.precioalavista.com/c395.html
http://www.precioalavista.com/c337.html
http://www.ra-ma.es/catalogo/bdv.htm
http://www.readysoft.es/mail/compravenda/0695.html
http://www.cyberlibro.com/
http://www.cyberlibro.com/busqueda_avanzada2.asp?menu=1&buscar=Lenguajes&offset=10
http://www.cyberlibro.com/busqueda_avanzada2.asp?menu=1&buscar=Sistemas%20operativos
http://www.cyberlibro.com/busqueda_avanzada2.asp?menu=1&buscar=Aplicaciones
http://www.cyberlibro.com/busqueda_avanzada2.asp?menu=1&buscar=Internet
http://www.infotodo-tiendas.com/rastro/c20.html
http://www.cocodrilolibros.com/
http://www.comerciovasco.com/cgi-bin/uncgi/buscar?prod=libros+inform%E1tica
http://www.discolibro.com/listadoLibros.jsp?cod_categoria=21
http://www.discolibro.com/listadoLibros.jsp?cod_categoria=585
http://www.discolibro.com/listadoLibros.jsp?cod_categoria=587
http://www.lalibreriadigital.com/

1.2.2 Subdomios

En la Figura E2, se visualizan los subdominios elegidos para el dominio Libros de Informática.

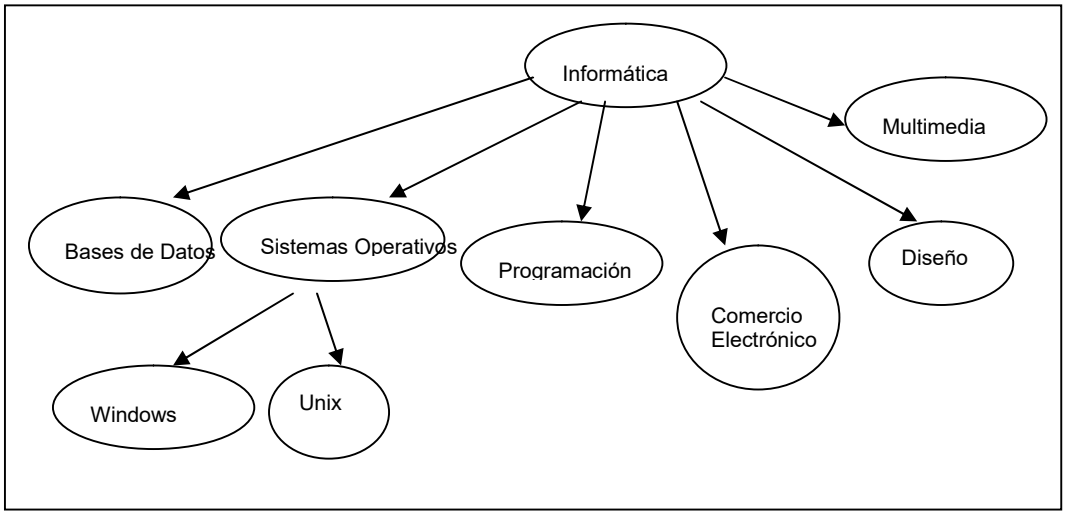


Fig. E1 – Subdominios para Informática

1.2.3 Expresiones Regulares

Las expresiones regulares se encuentran almacenadas en la tabla reg_exp de la base de datos LibrosInf. A modo de ejemplo, las expresiones regulares asociadas al subdominio investigación son:

reg_expresion	set_id	weigth
([Cc]omercio [Ee]lectrónico]	7	2
([Ee]-)	7	2
([Cc]iber)	7	1

1.2.4 Conceptos

En este juego de prueba utilizamos los conceptos “Windows_XP” y “Precios_libros_bd”.

1.2.5 Criterio de extracción

Para el concepto “Windows_XP” se extrae información de las páginas que clasificaron en el subdominio “Windows”, relacionadas con la expresión regular principal (Win)(|dows) y con las expresiones regulares secundarias (XP) y (Home Edition).

Para el concepto “Precios_libros_bd” se extrae información de las páginas que clasificaron en el subdominio “Base de Datos”, relacionadas con la expresión regular principal ([Bb][Aa][Ss][Ee][Ss] [Dd][Ee] [Dd][Aa][Tt][Oo][Ss]) y con las expresiones regulares secundarias ([Pp]tas|euros) y [0-9]{1, }.

Apéndice F - BASE DE DATOS

1.1 Introducción

En esta sección se describe la base de datos (BD) utilizada en este proyecto. Dicha base almacena toda la información necesaria por los procesos de clasificación y extracción, a excepción de las páginas.

1.2 Conexión a la base de datos

Para configurar la conexión a la BD ha ser utilizada por el prototipo debe realizar los siguientes pasos:

1. Crear un nuevo *Data Source Name* en el *ODBC* referenciando la base de datos deseada.
2. En el archivo *config.txt* debe hacerse referencia al *driver* utilizado y al *Data Source Name* creado en el paso anterior.

Nota: Puede utilizarse cualquier motor de BD para el cual se disponga de un *driver JDBC* apropiado. Referirse al documento de instalación (sección 4, Generación de conexiones ODBC) por una descripción más detallada sobre este proceso.

1.3 Descripción de las tablas

Tabla:	***pages***

Descripción: Esta tabla contiene el nombre del archivo .html correspondiente a las diferentes páginas que se consideran durante la ejecución de los procesos de clasificación y extracción.

Campos:	**(path_id, page_name, page_id)**
path_id:	Referencia al campo *path_id* en la tabla *path*.
page_name:	Nombre de la página (archivo .html).
page_id:	Identificador único de la página.

Tabla:	***path***

Descripción: Esta tabla contienen las URLs de las páginas que se consideran durante la ejecución de los procesos de clasificación y extracción.

Campos:	**(path_id, path)**
path_id:	Identificador único de cada path.
Path:	*url* menos el nombre de la página (archivo .html)

Tabla:	***tags***

Descripción: En esta tabla se almacenan los tags que se tomarán en cuenta en los procesos de clasificación y extracción, con un peso asociado a cada uno de ellos. Este peso es utilizado en los diferentes procesos para calcular el peso de cada página en un dominio dado.

Campos:	**(tag, weigth)**
tag:	Tag html.
weigth:	Peso asociado al tag html.

Tabla:	***subdomains***

Descripción: Esta tabla almacena información sobre los diferentes subdominios definidos por el usuario.

Campos:	**(subdomain_id, subdomain_name, min, max)**
subdomain_id:	Identificador único del subdominio.
subdomain_name:	Nombre del subdominio.
min:	Valor mínimo para que una página clasifique dentro del subdominio.
max:	Valor máximo para que una página clasifique dentro del subdominio.

Tabla: ***tree***

Descripción: Esta tabla contiene información sobre la organización de los diferentes subdominios. Deben ingresarse las referencias a estos de forma tal que estructuralmente formen un árbol, para que el proceso de clasificación funcione de forma correcta.

Campos:	**(father_id, child_id)**
father_id:	Referencia al campo *subdomain_id* de la tabla *subdomains*.
child_id:	Referencia al campo *subdomain_id* de la tabla *subdomains*.

Tabla: ***reg_exp***

Descripción: Esta tabla almacena todas las expresiones regulares necesarias durante los procesos de clasificación y extracción. El campo *set_id* se utiliza para hacer referencia a un grupo de expresiones regulares.

Campos:	**(reg_expresion, set_id, weigth)**
reg_expresion:	Expresión regular
set_id:	Identificador de un grupo de expresiones regulares.
weigth:	Peso asociado a la expresión regular.

Tabla: ***concepts***

Descripción: Esta tabla almacena los conceptos definidos por el usuario. Estas definiciones se utilizan durante el proceso de extracción.

Campos:	**(concept_name, concept_id)**
concept_name:	Nombre del concepto
concept_id:	Identificador único del concepto.

Tabla: ***extraction_criterias***

Descripción: Esta tabla almacena los criterios de extracción utilizados durante el proceso de extracción de información.

Campos:	**(extraction_criteria_id, main_reg_exp, max, min)**
extraction_criteria_id:	Identificador único del criterio de extracción.
main_reg_exp:	Expresión regular principal. Es una referencia al campo *set_id* de la tabla *reg_exp*.
max:	Peso máximo para el contexto aceptado por este criterio de extracción. Si no se quiere especificar este valor se debe asignar el valor 0.
min:	Peso mínimo para el contexto aceptado por este criterio de extracción.

Tabla: ***subdomain_concepts***

Descripción: Esta tabla almacena la relación entre los conceptos y los subdominios. Esta relación indica cuales conceptos se extraerán de cada subdominios.

Campos:	**(subdomain_id, concept_id)**
subdomain_id:	Referencia al campo *subdomain_id* en la tabla *subdominios*.
concept_id:	Referencia al campo *concept_id* en la tabla *concepts*.

Tabla: ***classification_result***

Descripción: En esta tabla se almacena el resultado del proceso de clasificación. Esta información es utilizada por el proceso de extracción para saber a que subdominio pertenece cada página.

Campos:	**(subdomain_id, page_id)**
subdomain_id:	Referencia al campo *subdomain_id* de la tabla *subdominio*.
page_id:	Referencia al campo *page_id* de la tabla *pages*.

Tabla: ***concept_extraction_criteria***

Descripción: Esta tabla almacena los criterios de extracción asociados a cada concepto. Por una explicación más detallada ver el documento *Informe Final.doc*.

Campos:	**(concept_id, extraction_criteria_id)**
concept_id:	Referencia al campo *concept_id* de la tabla *concepts*.
extraction_criteria_id:	Referencia al campo *extraction_criteria_id* de la tabla *extraction_criterias*.

Tabla: ***extraction_criteria_set***

Descripción: Esta tabla almacena la relación entre los grupos de expresiones regulares y los criterios de extracción. Cada criterio de extracción tiene un grupo de expresiones regulares asociadas, que son sus expresiones regulares complementarias. Estas son utilizadas para evaluar el contexto del concepto y decidir si será incluido o no en la extracción.

Campos:	**(set_id, extraction_criteria_id)**
set_id:	Referencia al campo *set_id* de la tabla *reg_exp*.
extraction_criteria_id:	Referencia al campo *extraction_criteria_id* de la tabla *extraction_criteria*.

Tabla: ***subdomain_set***

Descripción: Esta tabla almacena la relación entre los grupos de expresiones regulares y los diferentes subdominios. Cada subdominio tiene asociado un grupo de expresiones regulares, esto forma los criterios de clasificación para cada subdominio.

Campos:	**(set_id, subdomain_id)**
set_id:	Referencia al campo *set_id* de la tabla *reg_exp*.
subdomain_id:	Referencia al campo *subdominio_id* de la tabla *subdominios*.

1.4 ¿ Como crear una nueva base de datos ?

En esta sección se describe el proceso para generar una nueva base de datos y cargarla con información.

Primero que nada se debe generar el conjunto de tablas especificadas en la sección anterior. Se recomienda exportar el conjunto de tablas de alguna de las base de datos de ejemplo a una nueva base de datos.

A continuación se detallan los pasos a seguir para ingresar la información en las diferentes tablas:

Algunos de los objetos almacenados en la base, son utilizados por ambos procesos (Extracción y Clasificación), estos son:

- Las URLs de las páginas (tablas *pages* y *path*)
- Los Tags con sus pesos asociados (tabla *tags*)
- Los subdominios (tabla *subdomains*)

Se recomienda ingresar la información de la siguiente manera:

1) *Subdominios*

Se debe ingresar en la tabla *subdomains* la información correspondiente al nombre del subdominio (campo *subdomain_name*) y su identificación (campo *subdomain_id*). El resto de los campos se deja en blanco por el momento.

2) *Tags y Pages*

Esta información puede ingresarse en cualquier momento, ya que no tiene relación directa con ningún otro dato. Ver sección descripción de tablas (tags, pages) para mayor nivel de detalle.

En las siguientes secciones se explica como agregar la información utilizada por cada uno de los procesos.

1.4.1 Proceso de Clasificación

A continuación se presentan los conceptos a ingresar en la base de datos que son utilizados exclusivamente en este proceso.

1.4.1.1 Criterios de clasificación (tabla *subdomains*)

Ingresar los valores de los siguientes campo *max* y *min* de la tabla *subdomains*. Estos datos indican el rango de pesos para el cual serán aceptadas las páginas en cada subdominio.
Para completar los criterios de clasificación debe ingresarse la información de las expresiones regulares asociadas a ellos, esto se explica a continuación.

1.4.1.2 Expresiones regulares para clasificación (tablas *reg_exp* y *subdomain_set*)

Las expresiones regulares deben ingresarse en la tabla *reg_exp*. Esta tabla consta de tres campos: *reg_expresion*, *weight*, *set_id*.
En el campo *reg_expresión* debe ingresarse el valor de la expresión regular. Para tener información sobre como escribir una expresión regular, ver apéndice C – sección 1.3.

En el campo *weigth* debe ingresarse el peso (valor numérico) que se le va a asociar a cada expresión regular.
El campo *set_id* especifica a que grupo pertenece una expresión regular. Las expresiones regulares pertenecientes a un mismo grupo, deben ingresarse con la misma identificación (*set_id*). Luego en el campo *set_id* de la tabla *subdomain_set* se realiza la asignación de estos grupos de expresiones regulares (especificados en el campo *set_id* de la tabla *reg_exp*) a los diferentes subdominios.

1.4.1.3 El árbol de clasificación (tabla *tree*)

Como se menciono en secciones anteriores, los subdominios están organizados en forma de árbol. La tabla *tree* contiene información sobre dicha estructura. Aquí se debe especificar la relación de padre-hijo entre los diferentes subdominios. Los campos *father_id* y *child_id* de la tabla *tree* son referencias al campo *subdomain_id* de la tabla *subdomains*.

Importante: Esta relación debe representar un árbol para que el proceso de clasificación funcione correctamente (no debe contener ciclos y debe ser conexa).

1.4.2 Proceso de Extracción

A continuación se presentan los conceptos a ingresar en la base de datos que son utilizados exclusivamente en este proceso.

1.4.2.1 Conceptos (tablas *concepts*, *concepts_extraction_criterias* y *subdomain_concepts*)

La tabla *concepts* almacena información sobre los conceptos que se definen para el proceso de extracción. En esta se debe ingresar información sobre el nombre del concepto (campo *concept_name*) y el identificador de este (campo *concept_id*). El valor del campo *concept_id* debe ser único.

En la tabla *subdomain_concepts*, se especifica que conceptos están asociadas a cada subdominio. Esto se realiza asignando al campo *concept_id* de la tabla *subdomain_concepts* el identificador del concepto que se quiere asociar a cada subdominio.

En la tabla *concepts_extraction_criterias* se almacena que criterios de extracción tiene asociado cada concepto, esto es, se asigna a cada concepto (campo *concept_id* de la tabla *concepts_extraction_criterias*) uno o más criterios de extracción (campo *extraction_criteria_id*) especificados en la tabla *extraction_criterias*.

1.4.2.2 Criterios de extracción (tabla *extraction_criterias*)

La tabla *extraction_criteria* almacena información sobre los criterios de extracción que son utilizados en este proceso. A cada criterio de extracción se le debe asignar un identificador único (valor numérico), un valor máximo y otro mínimo (valores numérico), una expresión regular principal (referencia al campo *set_id* de la tabla *reg_exp*) y un conjunto de expresiones regulares complementarias que se utilizarán para evaluar el contexto.

La expresión regular principal indica la información que se buscará en las páginas.
Para cada ocurrencia de ésta se utiliza el conjunto de expresiones regulares complementarias para determinar el peso del contexto de esta ocurrencia.
Los valores máximo (campo *max*) y mínimo (campo *min*) indican el rango de pesos aceptado para dicho contexto.

Importante: El grupo asociado a la expresión regular principal debe contener un solo elemento.

1.4.2.3 Expresiones regulares de extracción (tablas *reg_exp* y *extraction_criteria_set*)

En la tabla *reg_exp*, como se explico anteriormente, se ingresan las expresiones regulares que se van a utilizar en los diferentes procesos.

La tabla *extraction_criteria_set* almacena información sobre los expresiones regulares complementarias que tiene cada criterio de extracción.
El campo *set_id* es una referencia, al campo *set_id* de la tabla *reg_exp* de esta forma se especifica el conjunto de expresiones regulares complementarias, y el campo *extraction_criteria_id* es una referencia, al campo *extraction_criteria_id* de la tabla *extraction_criteria* e indica que criterio de extracción tiene asociado este conjunto de expresiones regulares complementarias.

Referencias

[1] Adriana Martota, Regina Motz, Raúl Ruggia
Managing Source Schema Evolution in Web Warehouses
http://www.fing.edu.uy/inco/grupos/csi/Publicaciones/pub_csi2001/ webdw.pdf
Ultima visita: 14/03/2002

[2] Glossary for Information Retrieval
http://www.cs.jhu.edu/~weiss/glossary.html
Ultima visita: 08/12/2001

[3] Donna Harman
Ranking Algorithms
*********falta nombre del libro

[4] Regular Expressions explained
http://www.zez.org/article/articleprint/11/
Ultima visita: 09/02/2002

[5] Regular Expressions in Java
http://www.javaregex.com/
Ultima visita: 09/02/2002

[6] Arnaud Sahuguet, Fabien Azavant
Building Intelligent Web Applications Using Lightweiht Wrappers
Department of Computer and Information Science, University of Pennsylvania, USA

[7] Arnaud Sahuguet, Fabien Azavant
W4F User Manual, Tropea Inc., 2000
http://www.tropea-inc.com/
Ultima visita:

[8] Miriam Steiner
Mantenimiento de bases de datos alimentadas con páginas web
Proyecto de Maestría de la carrera Ingeniero en Computación
Facultad de Ingeniería, Universidad de la República, URUGUAY, 2001

[9] Mind-It
http://www.netmind.com

[10] HtmlDiff
http://www.htmldiff.com

[11] TopBlend
Yih-Farn Chen, Fred Douglis, Huale Huang, Kiem-Phong Vo: TopBlend: An Efficient Implementation of HtmlDiff in Java. AT&T Labs - Research Technical Report 00.5.1. - 2000.

[12] WebGUIDE
Fred Douglis, Thomas Ball, Yih-Farn Chen, Eleftherios Koutsofios: WebGUIDE: Querying and Navigating Changes in Web Repositories. WWW5 / Computer Networks 28(7-11): 1335-1344 - 1996.

[13] WebCQ
http://www.cc.gatech.edu/projects/disl/WebCQ - http://www.cse.ogi.edu/~lingliu/CQ/

[14] Ling Liu, Calton Pu, Wei Tang
WebCQ: Detecting and Delivering Information Changes on the Web. CIKM 2000: 512-519 - 2000.

[15] NiagaraCQ
http://www.cs.wisc.edu/niagara/

[16] Jeffrey F. Naughton, David J. DeWitt, David Maier, Ashraf Aboulnaga, Jianjun Chen, Leonidas Galanis, Jaewoo Kang, Rajasekar Krishnamurthy, Qiong Luo, Naveen Prakash, Ravishankar Ramamurthy, Jayavel Shanmugasundaram, Feng Tian, Kristin Tufte, Stratis Viglas, Yuan Wang, Chun Zhang, Bruce Jackson, Anurag Gupta, Rushan Chen: The Niagara Internet Query System. IEEE Data Engineering Bulletin 24(2): 27-33 - 2000.

[17] Jianjun Chen, David J. DeWitt, Feng Tian, Yuan Wang: NiagaraCQ: A Scalable Continuous Query System for Internet Databases. SIGMOD Conference 2000: 379-390 - 2000.

[18] WebBeholder
Santi Saeyor, Mitsuru Ishizuka: WebBeholder: A Revolution in Tracking and Viewing Changes on The Web by Agent Community Dept. of Information and Communication Engineering, Faculty of Engineering, University of Tokyo. 7- 3- 1 Hongo, Bunkyo- ku, Tokyo 113- 8656, JAPAN.

[19] HTML Compare
http://www.htmlcompare.com

[20] URL y Warning
http://www.urlywarning.com

[21] WebSecretary
http://homemade.hypermart.net/websec

[22] Araneus
http://www.dia.uniroma3.it/Araneus/

[23] G. Mecca, P. Atzeni, A. Masci, P. Merialdo, G. Sindoni: From Databases to Web-Bases: The ARANEUS Experience - Technical Report n. 34-1998 - Dipartimento di Informatica e Automazione, Universita' di Roma Tre, May, 1998

[24] Giansalvatore Mecca, Paolo Atzeni, Alessandro Masci, Paolo Merialdo, Giuseppe Sindoni: The Araneus Web-Base Management System. SIGMOD Conference 1998: 544-546 - 1998.

[25] Paolo Atzeni, Giansalvatore Mecca, Paolo Merialdo: To Weave the Web. VLDB 1997: 206-215 - 1997.

[26] Stéphane Grumbach, Giansalvatore Mecca: In Search of the Lost Schema. ICDT 1999: 314-331 - 1999.

[27] Grammars have Exceptions Valter Crescenzi, Giansalvatore Mecca Grammars Have Exceptions - Information Systems, Special Issue on Semistructured Data, 1998.

[28] Giansalvatore Mecca, Paolo Atzeni: Cut and Paste. JCSS 58(3): 453-482 - 1999.

[29] Giansalvatore Mecca, Alberto O. Mendelzon, Paolo Merialdo: Efficient Queries over Web Views. EDBT 1998: 72-86 - 1998.

[30] Paolo Atzeni, Alessandro Masci, Giansalvatore Mecca, Paolo Merialdo, Elena Tabet: ULIXES: Building Relational Views over the Web. ICDE 1997: 576 - 1997.

[31] Giansalvatore Mecca, Paolo Merialdo, Paolo Atzeni, Valter Crescenzi: The ARANEUS Guide to Web-Site Development. SEBD 1999: 167-177 - 1999.

[32] Paolo Atzeni, Giansalvatore Mecca, Paolo Merialdo: Design and Maintenance of Data-Intensive Web Sites. EDBT 1998: 436-450 - 1998.

[33] Giuseppe Sindoni: Incremental Management of Hypertextual Views. WebDB 1998: 98-117 - 1998.

[34] TracerLock
http://www.tracerlock.com

[35] The Informant
http://informant.dartmouth.edu/

[36] Clean up your Web pages with HTML TIDY
http://www.w3.org/People/Raggett/tidy/
Ultima visita: 20/07/2001

[37] Guía Rápida HTML
http://gias720.dis.ulpgc.es/Guias/Cursos/Tutorial_html
Ultima visita: 05/08/2001

[38] Extensible Markup Language (XML)
http://www.w3.org/XML
Ultima visita: 23/05/2001

[39] The XML cover pages
http://www.oasis-open.org/cover
Ultima visita: 23/05/2001

[40] http://www.xml.com
Ultima visita: 23/05/2001

[41] XML in 10 points
http://www.w3.org/XML/1999/XML-in-10-points
Ultima visita: 23/05/2001

[42] Extensible Markup Language (XML) 1.0 (Second Edition)
http://www.w3.org/TR/REC-xml
Ultima visita: 23/05/2001

[43] WordNet
http://www.ac-toulouse.fr/wordnet/wordnet.htm
Ultima visita: 11/06/2001