Instituto de Computación – Facultad de Ingeniería
Universidad de la República

Tesis de Maestría
en Ingeniería en Computación

Extracción de Información de la Web Basado en Ontologías

Alvaro Fernández Peirano
(alfernan@adinet.com.uy)

Director: Regina Motz

Montevideo, Uruguay
Diciembre de 2004

Extracción de Información de la Web Basado en Ontologías
Alvaro Fernández Peirano

ISSN 1510-7264
Tesis de Maestría en Ingeniería en Computación
Instituto de Computación – Facultad de Ingeniería
Universidad de la República

Montevideo, Uruguay, Diciembre de 2004

Resumen

En este trabajo se presenta un proceso que permite automatizar la extracción de información de la Web. A este tipo de procesos se les denomina Wrappers. Un wrapper es un programa que se conecta con una fuente de información, extrae los datos relevantes en base a reglas predefinidas por el usuario y devuelve la información encontrada en un formato estructurado.

El wrapper propuesto en este trabajo está guiado por consultas del usuario y basado en Ontologías. Una Ontología permite definir y trabajar sobre un vocabulario común para referirse a los diferentes conceptos del dominio, es decir, asociarle un significado inequívoco a cada concepto.

La Web es un gran repositorio de información que cambia constantemente, tanto en la actualización de contenido como en estructura. Por lo tanto un wrapper que tome como fuente de información la Web, debe de adaptarse a los constantes cambios que esta sufre. La técnica de wrappers basados en Ontologías es especialmente útil para adaptarse a los cambios estructurales debido a que no utiliza la estructura de la fuente para extraer información. El wrapper propuesto en este trabajo toma como base la técnica "Basada en Ontologías", focalizando su extracción en fuentes escritas en lenguaje HTML. Dado que HTML es un lenguaje standard que mantiene cierta estructura, el wrapper se extiende de manera que utiliza el significado de los diferentes ítems (tags) del lenguaje HTML al momento de realizar la extracción de modo de incrementar la inteligencia del mismo.

En el trabajo se proponen y desarrollan diferentes aspectos que hacen al proceso de búsqueda y extracción de información a los efectos de resolver el problema planteado:

a. *Desarrollo de una metodología para que dada una ontología se seleccionen solo las clases y relaciones que interesan.*
b. *Desarrollo de un algoritmo que permite la clasificación de páginas HTML en relevantes y no relevantes.*
c. *Desarrollo de una heurística para la distinción en documentos HTML entre tablas de contenidos y tablas de presentación.*
d. *Desarrollo de una heurística para determinar el alcance de los conceptos en la página HTML.*

INDICE

FIGURAS

TABLAS

1 Introducción

Con la expansión de la Web, los usuarios tienen acceso a una enorme cantidad de información representada generalmente como páginas HTML [1] o en el mejor de los casos en XML [2]. Para acceder a esta información, el usuario generalmente comienza a navegar los enlaces de algún sitio que clasifique la información de la Web o se realiza una búsqueda por palabras claves, en ambos casos el usuario navega los distintos enlaces decidiendo si esa página contiene información relevante o no sobre el tema buscado. Esta metodología de búsqueda de información de la Web presenta serias limitaciones y contratiempos. Navegar por los enlaces es una tarea pesada y generalmente el usuario termina perdiéndose ya que los resultados de la búsqueda suelen devolver demasiada información, generalmente mucho más de la que el usuario puede manejar.

Tal cual se desarrolla en [3], muchos de los sitios Web disponibles actualmente son diseñados y desarrollados por gente que no está adecuadamente entrenada ni preparada en el desarrollo de sitios Web y por ende generan contenido que es difícil de ser utilizado, es decir, es difícil de buscar información en dichos sitios.

Para poder extraer información de la Web de forma automática o semiautomática, un usuario, con conocimientos avanzados en informática puede construir un programa especializado que identifique los datos de interés en base a una serie de reglas gramaticales, que los extraiga y realice una transformación a una estructura de datos apropiada para su posterior manipulación. A estos programas se les denomina wrappers.

No es deseable que los wrappers sean generados por el usuario debido a que escribir reglas gramaticales de extracción es un trabajo tedioso, consume mucho tiempo y requiere una gran experiencia en el dominio de la aplicación. La dificultad se incrementa aún más cuando se posee un gran número de fuentes de información heterogéneas de las cuales extraer la información o cuando estas fuentes son muy dinámicas en su evolución, como es el caso de la información accesible vía Web.

Un primer problema a resolver cuando se desea extraer información, es el hecho de poder trabajar sobre un vocabulario común para referirse a los diferentes conceptos del dominio, es decir, asociarle un significado inequívoco a cada concepto. Este problema se resuelve definiendo una *Ontología* sobre el dominio que se desee trabajar.

Tal como se define en [4] una Ontología ha de verse como un entendimiento común y compartido de un dominio, que puede compartirse entre científicos y sistemas computacionales, entre personas y sistemas heterogéneos, es decir, es una base de datos que describe conceptos del mundo real o de algún dominio, algunas de sus propiedades y como los conceptos se relacionan entre si.

Este trabajo aborda el problema a través del análisis y desarrollo de metodologías para realizar la búsqueda y extracción de información de la Web en forma automática. La metodología presentada está dirigida por ontologías.

El Instituto de Computación (InCo) de la Facultad de Ingeniería está desarrollando un proyecto llamado "Web WareHouse" y consiste en construir un repositorio de datos de la Web acerca de un determinado dominio de interés, similar a lo que se conoce como una Data WareHouse de un Sistema de Información. Para ello, el proyecto ha sido dividido en varias etapas. La primera etapa consiste en construir wrappers que permitan extraer la información de un determinado dominio de interés y corresponderla a una estructura de datos manipulable. La segunda etapa consiste en la construcción de un módulo que se encargue de integrar la salida de cada uno de los wrappers. Finalmente, en la tercera etapa se realiza el mapeo de estas estructuras, obtenidas en la etapa anterior, a una Data WareHouse. Todo el proceso está apoyado por una meta data común a cada uno de los módulos que integran el proyecto.

El trabajo desarrollado en este documento se corresponde con la primera etapa del proyecto de la construcción de una "Web WareHouse", es decir, con la construcción de wrappers y su salida en una estructura de datos manipulable.

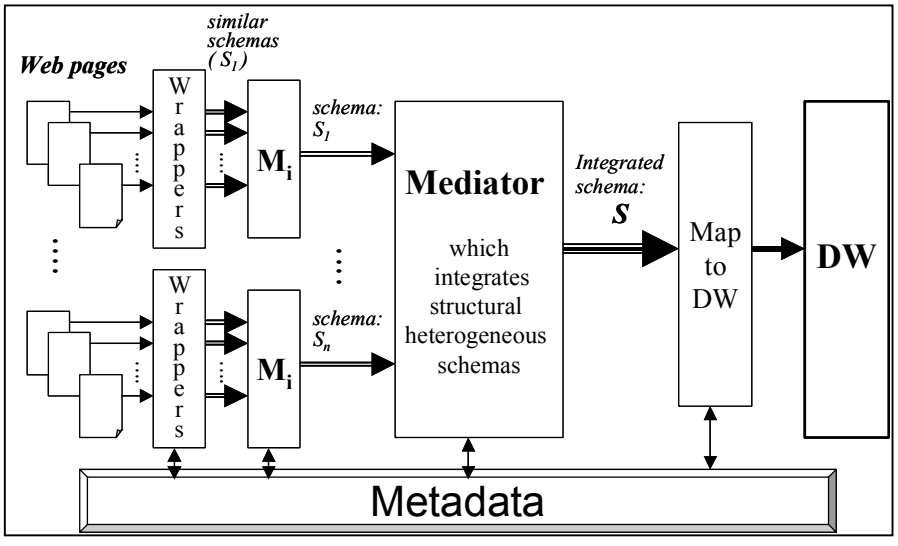

Figura 1 – Contexto del Trabajo en el InCo

El resto del trabajo se organiza como sigue. En la Sección 2 se hace referencia a una clasificación de técnicas de construcción de wrappers y se presentan las características de de los wrappers que se identifican como más relevantes. En la Sección 3 se presentan los diferentes aspectos que hacen al proceso de búsqueda y extracción de información. Se desarrolla una metodología para que dada una ontología se seleccionen solo las clases y relaciones que interesan. Se desarrolla un algoritmo que permite la clasificación de páginas HTML en relevantes y no relevantes. Se desarrolla una heurística para la distinción en documentos HTML entre tablas de contenidos y tablas de presentación. Y también se desarrolla una heurística para determinar el alcance de los conceptos en la página HTML. En la Sección 4 se presenta un ejemplo completo de funcionamiento del sistema. En la Sección 5 se presenta pruebas realizadas y datos experimentales que sustentan lo desarrollado en el trabajo. Finalmente en la Sección 6 se presentan algunas conclusiones y trabajos futuros.

2 Sobre la generación de Wrappers

Tal cual se mencionó en la introducción un wrapper es un programa que extrae datos desde una fuente de información en base a reglas predefinidas por el usuario y transforma esta información datos estructurados, en nuestro caso XML.

A continuación se describe la clasificación de técnicas de generación de wrappers presentada por Alberto H. F. Laender *et al.* para introducir al lector en las técnicas de generación de wrappers. Por un análisis más detallado de las propiedades cualitativas de cada técnica se recomienda referirse a [5].

2.1 Clasificación de técnicas de desarrollo de Wrappers

Existen varias técnicas y enfoques que son utilizados para la generación de wrappers. En esta sección se aborda una clasificación basada en la técnica que se utiliza para escribir las diferentes reglas de extracción según presentado en [5].

Lenguajes para desarrollo de Wrappers. Esta técnica de generación de wrappers propone la definición de lenguajes propietarios especialmente diseñados para la extracción de información. Son una alternativa a los lenguajes de propósito general como por ejemplo Perl o Java. Tienen la desventaja de que el usuario debe de aprender este lenguaje propietario y para escribir las reglas de extracción se tiene que basar en la estructura del documento, pero por otro lado suelen ser más potentes que los lenguajes de propósito general. Esta técnica está muy ligada a la estructura del documento, por lo que en el caso de las fuentes HTML que sufren cambios constantemente ya sea estructurales y de contenido, la adaptación a los cambios resulta compleja. Un ejemplo de este tipo de wrappers es Jedi [6].

Herramientas HTML Dependientes. Esta técnica se basa en la estructura del documento HTML para la extracción de la información. Como representación de la estructura del documento HTML (representación jerárquica de los tags HTML) se utiliza DOM [7] a partir del cual se genera un árbol a ser parseado durante el proceso de extracción. Este tipo de técnica es soportada con una interfaz gráfica, en donde el usuario marca los sectores del documento que le son de interés y la herramienta infiere el wrapper. Para realizar la inferencia, esta técnica se apoya en "wizards" que forman parte de la interfaz gráfica. Este tipo de técnica permite que un usuario poco experimentado pueda generar con relativa facilidad un wrapper, pero como en el punto anterior, esta técnica está muy ligada a la estructura del documento por lo tanto la adaptación a los cambios resulta compleja. Para lograr mayor flexibilidad en este tipo de técnica se suele incorporar en la herramienta un lenguaje propietario para permitir extender las reglas inferidas de la interfaz gráfica, es decir, se incorpora un lenguaje propietario tal cual se estudió en el punto anterior. Un ejemplo de este tipo de wrappers es W4F [8].

Procesamiento de Lenguaje Natural. Se aplican reglas basadas en la sintaxis y en la semántica para identificar la información relevante en un documento escrito en lenguaje natural. Esta técnica se basa en filtros, en marcas de partes de textos y en marcas lexicográficas y de semántica para construir relaciones entre las oraciones y los elementos de la misma. Por ejemplo, infiere el wrapper a partir de la identificación de los verbos, adjetivos y substantivos de una porción de texto. Los wrappers son construidos a partir de ejemplos que el usuario ingresa y en el cual marca los sectores del texto que le son de interés, para lo cual es necesario contar con una interfase gráfica, y a partir de algoritmos de procesamiento de lenguaje natural la herramienta infiere las reglas y genera el wrapper. Esta técnica tiene la ventaja de no depender de la estructura de la fuente, por lo tanto se puede decir que se adapta a los cambios con facilidad y que su desempeño es ideal para fuentes con textos semiestructurados (fuentes HTML o no), es decir, textos escritos en lenguaje natural, como por ejemplo, artículos, noticias, informes, etc. Sin embargo poseen la desventaja de que su implementación depende exclusivamente del idioma en el que se desee extraer información, y no resulta portable el wrapper cuando se desea extraer información en otro idioma. Otro problema que tiene esta técnica es la ambigüedad de las palabras utilizadas para definir y referirse a los conceptos de un dominio. Un ejemplo de este tipo de wrappers es RAPIER [9].

Técnicas de Inducción. Esta técnica se focaliza en la generación de delimitadores basados en una serie de ejemplos suministrados por el usuario. A diferencia del punto anterior que se basa en delimitaciones lingüísticas, propias del lenguaje natural (verbos, sustantivos, etc.), esta técnica delimita las porciones de texto de interés de acuerdo a su formato, por ejemplo, números con decimales, palabras que comienzan en mayúsculas, etc. Esta técnica se apoya en interfases gráficas para facilitarle al usuario la entrada de ejemplos a partir de los cuales inferir las reglas de extracción y generar el wrapper. Si bien esta técnica no se basa en la estructura del documento, si se basa en el formato de la información que se desea explorar, por lo tanto se puede decir que este tipo de técnicas no se adapta con facilidad a los cambios que la fuente pueda sufrir. Un ejemplo de este tipo de wrappers es WIEN [10].

Técnicas basadas en Modelado. Esta técnica toma como entrada una estructura, conforme a un modelo de datos (tuplas, listas, tablas, etc.), que define los objetos de interés dentro de la fuente. Esta técnica es útil cuando la fuente presenta la información en forma estructurada o semiestructurada. Ejemplos de este tipo de fuente pueden ser las páginas que presentan la información en forma de grilla, como por ejemplo, los sitios de venta electrónica que generalmente muestran una lista de artículos para que el usuario seleccione lo que desea comprar. Para escribir la estructura el usuario debe de descomponer el documento jerárquicamente, resaltando las regiones de interés y describiendo su semántica. Para ello es fundamental contar con una buena interfaz gráfica que de soporte a la construcción de la estructura. Esta técnica se basa en la estructura del documento y en el formato de la información que se desea explorar, por lo tanto se puede decir que este tipo de técnicas no se adapta con facilidad a los cambios que la fuente pueda sufrir. Un ejemplo de este tipo de wrappers es NoDoSE [11].

Técnicas basadas en Ontologías: Los puntos anteriores se basan, de alguna forma u otra, en la estructura de la fuente y/o en la estructura de presentación de la información buscada dentro del documento. En el caso del wrapper basado en "Procesamiento de Lenguaje Natural", si bien su implementación no se basa en la estructura del documento, la misma queda sujeta al idioma que se utilice, si la fuente cambia de idioma, este tipo de técnicas no permiten su reutilización, así como también existe el problema de la ambigüedad de las palabras utilizadas para definir y referirse a los conceptos de un dominio. Las ontologías proporcionan una vía para representar el conocimiento en la Web y son un enfoque importante para capturar semántica. La definición más consolidada es la que la describe como "una especificación explícita y formal sobre una conceptualización compartida" [12], [13]. Es decir, las ontologías definen conceptos y relaciones de algún dominio, de forma compartida y consensuada; y esta conceptualización debe ser representada de una manera formal, legible y utilizable por las computadoras. Si bien las ontologías no son fáciles de escribir y deben de ser provistas por un experto en el dominio a ser tratado, la extracción de la información basada en este tipo de técnicas puede ser totalmente automatizada ya que no depende de ninguna manera en estructuras de la fuente ni del significado semántico del lenguaje (como es el caso de la técnica basada en "Procesamiento de Lenguaje Natural"). Se puede afirmar que es la técnica que mejor se adapta a los cambios que puedan sufrir las fuentes. Como corolario a esto se obtiene un wrapper que sirve tanto para fuentes HTML como no HTML. Un ejemplo de este tipo de wrappers es el trabajo realizado por Embley y Campbell [14].

2.2 Características deseables en los Wrappers

Se presenta en esta sección las características de los wrappers que se identifican como más relevantes [5], y que sería deseable que el wrapper propuesto tuviera en cuenta.

Automatización. Esta característica se refiere a la cantidad de trabajo que debe de realizar el usuario durante la generación del wrapper. Toda herramienta de extracción de información debe de ser lo mas automática posible, liberando al usuario de tediosas tareas.

Manejo de objetos con estructura compleja. En el caso particular de páginas HTML se puede decir que su estructura es compleja y semiestructurada, además es común observar documentos HTML que no están bien formados, páginas HTML que presentan varios niveles de anidación, por ejemplo el hecho de escribir una tabla dentro de otra, dentro de un párrafo escribir un enlace, etc. En este caso es necesario que el wrapper generado pueda manejar objetos con estructura compleja ya que la extracción de la información se realizará sobre fuentes escritas en su mayoría en HTML.

Soporte para fuentes con datos semiestructurados y con texto semiestructurado. Por dato semiestructurado se refiere a aquella fuente de información que representa la información en forma de grilla, como pueden ser las páginas que presentan productos para la venta. Por texto semiestructurado se refiere a aquella fuente de información que representa la información en un texto libre y por lo tanto los diferentes conceptos deben de ser inferidos a partir de este texto. Las páginas HTML pueden ser escritas utilizando tanto datos semiestructurados como textos semiestructurados, por lo tanto el wrapper a ser generado debe de tener soporte para ambos tipos de fuentes.

Salida en Formato XML. XML es sin lugar a duda el formato de representación de datos e intercambio de los mismos más usado actualmente. Existen una inmensa variedad de técnicas y herramientas capaces de procesar documentos XML. Por lo tanto se hace necesario que el wrapper genere su salida en formato XML.

Adaptación a los Cambios. Por lo general las fuentes de información están sufriendo cambios constantemente y en particular en la Web, donde la información se actualiza constantemente y en donde la presentación y contenido de las páginas HTML cambia minuto a minuto. Para que un wrapper funcione adecuadamente extrayendo información de la Web es fundamental que pueda acompañar los cambios constantes producidos en la fuente (página HTML) y que además continúe funcionando adecuadamente. Para que esto sea posible el wrapper no debe de basarse únicamente en la estructura del documento sino que debe focalizarse en la semántica de la fuente a ser tratada. Dado que HTML es un standard aceptado mundialmente, cuando la fuente a consultar es una página HTML, el wrapper puede utilizar el significado de los diferentes ítems (tags) del lenguaje de forma de incrementar la inteligencia de la extracción.

Disponibilidad de Interfase Gráfica. Con el avance que existe hoy en día en los entornos gráficos de los sistemas operativos, se hace necesario que el wrapper posea una interfaz gráfica que facilite al usuario la construcción del mismo utilizando asistentes que guíen al usuario realizándole preguntas.

Soporte para fuentes No HTML. Mucha de la información semiestructurada que es almacenada electrónicamente no está presente en formato HTML, sino en archivos de texto tales como emails, códigos de programas, documentación, logs, etc. Si bien se puede decir que esta es una característica deseable en un wrapper de propósito general, se puede afirmar que en este caso particular de estudio, las fuentes a ser consultadas son páginas HTML, por lo tanto esta característica no será tomada en cuenta en este trabajo.

2.3 Conclusiones

La Web es un gran repositorio de información que cambia constantemente, en cuanto a la información brindada ya que la misma es actualizada constantemente. Por lo tanto un wrapper que tome como fuente de información la Web, debe de adaptarse a los constantes cambios que esta sufre, ya sean, cambios de contenido, así como cambios estructurales. De nada sirve tener un wrapper que al menor cambio sufrido en la página HTML haya que adaptarlo en forma manual.

Tal como se presenta en [15] uno de los principales problemas al momento de extraer información de un repositorio de datos, como lo son las páginas HTML, es la utilización de palabras ambiguas para definir y referirse a los conceptos de un dominio.

La técnica que resuelve el problema de la ambigüedad de los conceptos de un dominio, es la técnica basada en ontologías. Esta técnica, además, se adapta con facilidad a los cambios que las páginas HTML puedan sufrir, dado que no utiliza la estructura de la fuente para extraer información.

De las técnicas para el desarrollo de wrappers expuestas en la sección 2.1, se puede observar que, a excepción de las técnicas de "Procesamiento de Lenguaje Natural" y "Basadas en Ontologías", el resto de las técnicas de alguna forma u otra utilizan la estructura del documento y/o la estructura de presentación de la información en el documento para realizar la extracción de información. Este tipo de técnicas no se adaptan con facilidad a los cambios que la fuente pueda sufrir, ya sea de contenido, como de estructura. Si bien la técnica de "Procesamiento de Lenguaje Natural", no utiliza estructura alguna, presenta el problema de la ambigüedad de las palabras utilizadas para definir y referirse a los conceptos de un dominio.

Los wrappers basados en ontologías presentan la dificultad al momento de escribir la ontología en si misma, la cual debe de ser provista por un experto en el dominio a ser tratado. El hecho de escribir ontologías escapa al alcance de este trabajo y se asume que ya se dispone de la ontología para el dominio sobre el cual extraer información.

3 Wrappers Basados en Ontologías

Como resultado del análisis del Capítulo 2, el wrapper propuesto en este trabajo toma como base la técnica "Basada en Ontologías", focalizando su extracción en fuentes escritas en lenguaje HTML. Dado que HTML es un lenguaje standard que mantiene cierta estructura, el wrapper se extiende de manera que utiliza el significado de los diferentes ítems (tags) del lenguaje HTML al momento de realizar la extracción de modo de incrementar la inteligencia del mismo.

A continuación se presenta el proceso para la construcción de wrappers basados en ontologías. El proceso consta de varias etapas desde que el usuario ingresa la información a procesar hasta que se obtiene una salida. La siguiente figura presenta un diagrama del proceso propuesto:

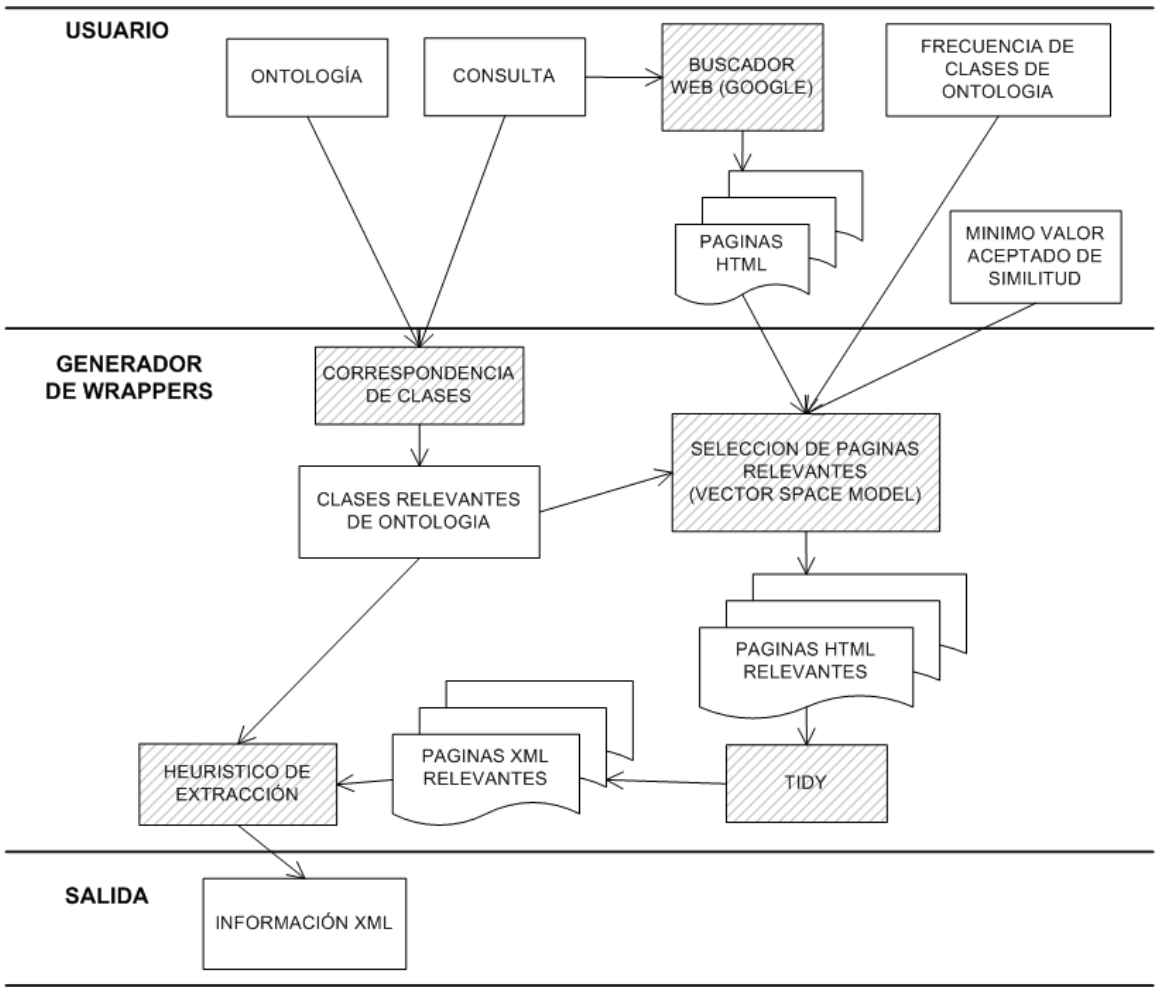

Figura 2 – Proceso de Construcción de Wrappers Basados en Ontologías

3.1 Usuario

Como entrada para el wrapper el usuario debe de especificar una consulta sobre la cual trabajar. En base a esta consulta, debe de suministrar páginas HTML que pueden ser el resultado de una búsqueda utilizando algún motor de búsqueda como por ejemplo Google (http://www.google.com). El usuario debe, además, suministrar la definición de la ontología, la frecuencia con que deben de aparecer las clases de la ontología en los documentos según la información que se desee extraer, así como un valor porcentual mínimo para el cual considera aceptable que la ontología y la página HTML son "similares".

El hecho de definir una ontología y la frecuencia de aparición de sus clases es una tarea compleja que debe de ser realizada por un experto en el dominio sobre el cual se desea trabajar. Ambos puntos exceden al alcance de este trabajo.

3.2 Generador de Wrappers

Esta etapa es la encargada de realizar todos los pasos necesarios para poder construir un wrapper basado en ontologías.

Esta etapa presenta varias dificultades. Algunas de estas dificultades pueden ser expresadas de la siguiente manera:

a. ¿De toda la ontología, cuales son las clases y relaciones que interesan a los efectos de resolver el problema planteado?
b. ¿De las páginas suministradas por el usuario, cuales contienen información para extraer y cuales son irrelevantes?
c. Cuándo se analiza una página suministrada por el usuario que contiene información relevante, ¿que porción de la misma se extrae?, ¿se extrae la página completa o solo la sección (el contexto) de la página donde se encontró la información?
d. En caso de querer extraer solo el contexto, ¿cómo se define esta contexto?, ¿es la oración dónde aparece la información?, ¿es el párrafo?, ¿qué significa "oración" en HTML?, ¿y "párrafo"?
e. Los desarrolladores de páginas HTML por lo general no escriben HTML de acuerdo a la especificación propuesta por W3C [1], como por ejemplo, no cierran los tags HTML. Esto ocasiona un problema ya que al no tener un documento bien estructurado, se dificulta la escritura de algoritmos que trabajen sobre el documento. ¿Cómo se resuelve este problema de forma de automatizar el wrapper?
f. ¿Se puede independizar al wrapper del idioma de las páginas suministradas por el usuario?, ¿y de la consulta?

En este capítulo se abordarán todas estas interrogantes, junto con algunas otras que vayan surgiendo a medida que se profundiza en el análisis de la solución. Esta etapa se divide en una serie de pasos bien definidos, los cuales se enumeran a continuación.

3.2.1 Identificación de Conceptos Relevantes de la Ontología

La primer tarea que el wrapper debe de realizar es una correspondencia entre la consulta y la ontología. Es decir, a partir de la consulta se seleccionan los conceptos y relaciones de interés de la ontología que son relevantes para la consulta planteada.

En una primera instancia se propuso eliminar de la consulta que el usuario ingresa las palabras que son no connotativos, es decir, que no aportan valor para la extracción, estas palabras se clasifican en los siguientes grupos:

a. *Pronombres:* son palabras que señalan a una persona, animal o cosa, reemplazando al sustantivo en el sujeto. Ejemplos de este grupo son: él, ellos, nosotros, conmigo, les, contigo, etc.
b. *Modificador Directo:* son artículos y adjetivos que acompañan al sustantivo núcleo del sujeto y concuerdan con él en género y número. Ejemplos de este grupo son: la, el, los, etc.
c. *Adverbios:* son palabras que modifican a un verbo, un adjetivo o a otro adverbio. En la oración funcionan como circunstanciales o formando parte de modificadores. Ejemplos de este grupo son: lejos, arriba, quizás, muy, también, etc.

Si bien parece razonable quitar los grupos antes mencionados de la consulta que el usuario ingresó, existe el problema de que estos grupos dependen del idioma con el que se esté trabajando, inclusive en otros idiomas diferentes al español pueden no existir estas definiciones o pueden existir nuevas definiciones.

Para poder afrontar el problema de que el wrapper funcione correctamente independientemente del idioma utilizado, se decidió no quitar ninguna palabra de la consulta ingresada por el usuario, sino que se intenta realizar una correspondencia entre todas las palabras de la consulta y la ontología. Las palabras que no tengan un correspondiente dentro de la ontología se descartan y no son tenidas en cuenta durante el proceso.

Por lo tanto, la correspondencia entre la consulta y las clases de la ontología se realiza de la siguiente manera:

a. Se buscan todas las palabras de la consulta que aparezcan directamente en la ontología, ya sea como nombre de clase, etiqueta de clase o valor de alguna propiedad de la clase o nombre de relación entre clases de la ontología.
b. De cada palabra de la consulta se obtiene una lista de sinónimos y para cada sinónimo se procede como se indicó en el punto anterior.
c. Para cada clase de la ontología que se seleccionó en el punto a, se consideran como relevantes todas las subclases que dependan de esta, es decir, se considera todo el árbol cuya raíz es la clase en cuestión. Las relaciones que puedan existir en este subárbol no son tomadas en cuenta.
d. De las relaciones surgidas en el punto a, se consideran como relevantes las clases que son extremo de la relación, es decir, las clases que definen la relación. Para cada una de estas dos clases, se aplica lo dicho en el punto c.

El concepto de Diccionario de Sinónimos debe ser utilizado siempre, ya que asegura un mayor grado de certeza al momento de seleccionar las Clases, Sub Clases y sus Relaciones dentro de la ontología. Como ejemplos de Diccionario de Sinónimos se pueden citar los siguientes:

a. Idioma Español: http://tradu.scig.uniovi.es/sinon.cgi
b. Idioma Inglés: http://www.cogsci.princeton.edu/~wn/

Como resultado de la aplicación de los puntos anteriores se obtiene una nueva ontología, únicamente con las clases y relaciones relevantes.

A continuación se presenta un ejemplo sobre este punto. Se asume que se tiene la siguiente ontología:

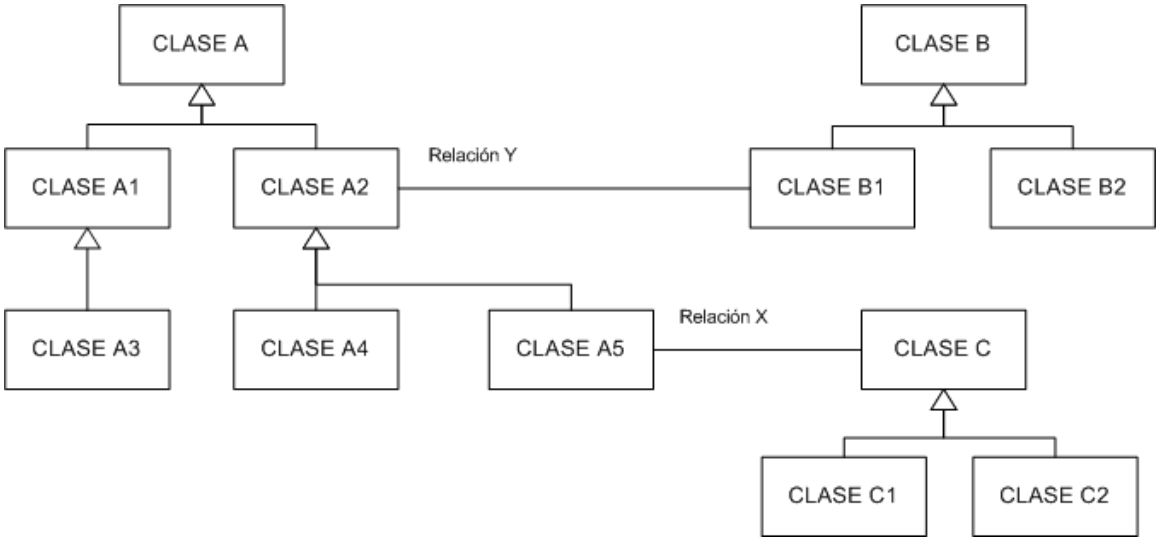

Figura 3 – Ejemplo Genérico de Ontología

La consulta sobre la cual se quiere extraer información es la siguiente:

- Los A2 relacionados mediante X con Z.

Siguiendo los pasos antes presentados para realizar la correspondencia de clases se obtiene lo siguiente:

a. Palabras que aparecen directamente en la ontología: A2, por lo tanto la clase A2 de la ontología se corresponde con el término A2.
b. También aparece el término X que es una relación, por lo tanto se consideran las clases que están en los extremos de la relación, las cuales son A5 y C.
c. En el diccionario de sinónimos aparece que el término Z es sinónimo de B2, por lo tanto la clase B2 de la ontología se corresponde con el término Z.
d. De cada clase seleccionada se consideran las subclases derivadas, sin tener en cuenta las relaciones existentes en las subclases. Se seleccionan las subclases A4, C1 y C2.

El resultado de la correspondencia de clases queda de la siguiente manera:

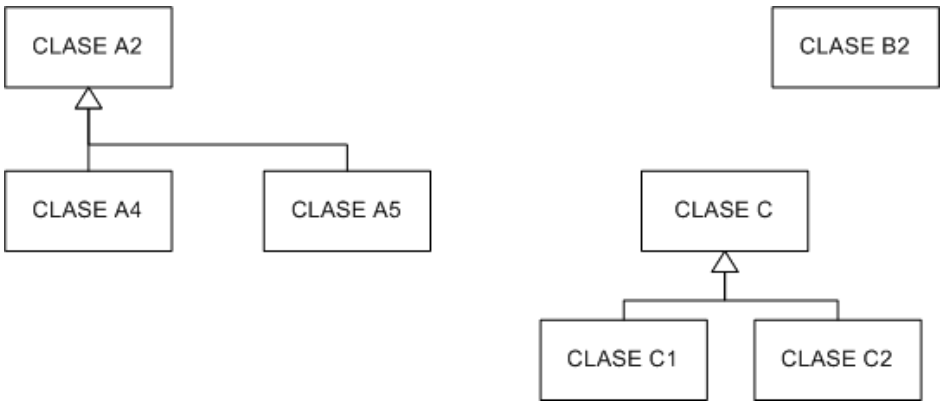

Figura 4 – Ejemplo Genérico de Extracción de Clases Relevantes

La relación X no se considera como parte de las clases relevantes ya que no es un concepto (clase) propiamente dicho, en cambio, la relación aporta el hecho de que se considere a la Clase C y todas sus subclases, así como a la Clase A5.

Puede darse el caso de que no exista ninguna correspondencia entre la consulta y la ontología, es decir, que luego de aplicar los puntos anteriores no se obtiene ninguna clase de la ontología que se corresponda con ninguna palabra o sinónimo de la consulta. En este caso se puede afirmar que o bien la consulta no está bien formulada o bien la ontología no es la adecuada para el dominio considerado. En este caso se debiera de reformular la consulta en términos de la ontología o utilizar otra ontología mas adecuada para el dominio sobre el cual se desea extraer información.

3.2.2 Selección de Páginas HTML Relevantes

En base a la ontología obtenida, las páginas HTML suministradas por el usuario y la frecuencia de aparición de las clases en los documentos, también suministradas por el usuario, el wrapper debe de seleccionar aquellas páginas HTML relevantes, es decir, se eliminan aquellas páginas HTML que no tratan sobre el tema de interés. Para ello, se utiliza un modelo que ayuda a determinar cuan similares son dos documentos (la ontología y la página HTML en este caso). El modelo tiene el nombre de Vector Space Model (VSM) y fue propuesto por Salton [16]. Como salida se obtienen las páginas HTML que son relevantes para el tema tratado, las que no son relevantes se descartan.

El modelo propone armar dos vectores. El primero de ellos, llamado OV, indica para cada clase relevante de la ontología cual es la frecuencia esperada con que debe de aparecer la clase en el documento HTML.

El valor de la frecuencia con que se espera que una clase aparezca en un documento, forma parte de los valores de entrada que son suministrados por el usuario y escapa al alcance de este documento.

El vector OV se define de la siguiente manera:

OV = (<ClaseC1 : FrecEsperada1>, <ClaseC2 : FrecEsperada2>, …)

El segundo vector que propone el modelo, llamado DV, contiene para las clases relevantes, la cantidad de veces que aparecen efectivamente en el documento HTML. Como ocurrencia del concepto se debe de considerar no solo el nombre de la clase sino la etiqueta de la clase, los sinónimos de dicha clase (para ello se utiliza un Diccionario de Sinónimos tal cual se explicó en la sección 3.2.1), así como también los posibles valores que las propiedades de la clase puedan tomar. Para saber que valores puede tomar una propiedad, en la definición de la ontología, se debe de asociar a cada propiedad, o bien una lista con los posibles valores, o bien una expresión regular que ayude a la identificación de las mismas dentro del documento.

El vector DV se define de la siguiente forma:

DV = (<ClaseC1 : FrecReal1>, <ClaseC2 : FrecReal2>, …)

Como se mencionó antes, se está midiendo la similitud que existe entre el Vector de la Ontología (OV) y el Vector del Documento (DV), para ello se mide el coseno del ángulo entre estos dos vectores. Para ello el modelo Vector Space Model utiliza la función Similarity Cosine Function (SIM), la cual calcula el ángulo agudo entre ambos vectores:

$$SIM(D, O) = \cos \varnothing = P / N$$

Figura 5 – Función Similitud

Donde D es el documento fuente, O es la ontología considerada, P es el producto escalar de ambos vectores y N es la normalización de los vectores. Dicho de otra manera la anterior fórmula queda:

$$\text{SIM}(D, O) = \cos \varnothing = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2 \sum_j y_j^2}}$$

Figura 6 – Función Similitud Expandida

Siendo x_i los valores con se espera que aparezcan las clases en la ontología y y_i la cantidad de veces que aparece la clase en la fuente consultada.

Se considera que la página HTML es relevante cuando el resultado de SIM es superior a un valor mínimo aceptable. Este valor debe de ser suministrado por el usuario.

3.2.3 Transformación de HTML a XML

Al buscar información dentro de páginas HTML, uno de los primeros problemas a los que se enfrenta el usuario es que en la mayoría de los casos el código HTML no está bien escrito, es decir, los tags no tienen su homólogo que los cierra. Esto ocasiona un problema ya que al no tener un documento bien estructurado, se dificulta la escritura de algoritmos que trabajen sobre el documento.

Para solucionar este problema, la página HTML se pasa por Tidy [17] para obtener una página bien escrita (desde el punto de vista de estructura de HTML), es decir, todos los tags tienen su homólogo que los cierra. De esta forma el documento queda bien estructurado y es manejable desde el punto de vista de algoritmos.

Actualmente XML está muy difundido y existen una variedad amplia de API's que facilitan el recorrido de las estructuras de XML en busca de información, por lo que sería deseable que la página HTML se pudiera representar como un documento XML. Pasar de HTML a XML es una tarea trivial con la ayuda de Tidy. Por lo tanto esta etapa del wrapper consiste en obtener, de las páginas HTML relevantes, un código HTML bien formado (desde el punto de vista de la estructura) y representado con estructura de XML, es decir, se obtienen las páginas XML relevantes.

A continuación se presentan algunos ejemplos de las transformaciones que hace Tidy desde el código HTML mal escrito un código XML:

HTML (mal escrito)	XML (Tidy)
`<h1>título`	`<h1>título</h1>`
`<h2>sub título</h3>`	`<h2>sub título</h2>`
`<h1><i>cursiva título</h1>`	`<h1><i>cursiva título</i></h1>`
`<p>Nuevo párrafo <b>negrita`	`<p>nuevo párrafo <b>negrita</b>`
`texto suelto`	`<p>texto suelto</p>`
`<a href="#refs">referencia<a>`	`<a href="#refs">referencia</a>`

`<li>item 1`
`<li>item 2`
	`<ul>`
	`  <li>item 1</li>`
	`  <li>item 2</li>`
	`</ul>`

Tabla 1 – Ejemplo de Tidy

Es importante recalcar que de existir texto que no está contenido en ningún tag, Tidy lo pone dentro del tag "P", de esta forma Tidy asegura que todos los textos del documento HTML están contenidos dentro de un tag (con su homólogo que los cierra) y que por lo tanto el nuevo documento es un documento XML bien formado.

Al momento de ejecutar Tidy hay que ingresarle los siguientes valores a las variables que se indican en la tabla:

Variable de Entorno	Valor	Explicación
char-encoding	latin1	Permite leer caracteres ASCII hasta 255.
output-xml	yes	La salida es en XML.
add-xml-decl	yes	Agrega la declaración necesaria al archivo para que sea compatible con la especificación de XML.
Enclose-text	yes	El texto que esté suelto lo encierra con el tag "P".
hide-comments	yes	Quita los comentarios HTML.
tidy-mark	no	No pone marca de que el texto fue procesado con Tidy.

Tabla 2 – Parámetros de Tidy

3.2.4 Heurístico de Extracción

Finalmente con las páginas XML relevantes y junto con las clases de la ontología relevantes, se procede a aplicar un heurístico que extrae la información deseada. A continuación se presenta como sería el heurístico de extracción propuesto.

El heurístico de extracción propuesto a continuación forma parte del aporte realizado por este trabajo a la comunidad informática. El mismo surge a raíz de analizar a fondo la especificación propuesta por W3C sobre HTML [1]. Si bien la especificación sugiere no utilizar tablas para armar la presentación de la página, las mismas son ampliamente utilizadas por los desarrolladores de páginas HTML. Uno de los problemas que ataca el heurístico propuesto es el hecho de eliminar de la página considerada las tablas que son utilizadas para armar la presentación de la página HTML.

La extracción se realizará sobre documentos HTML, por ende contiene información de estructura de documento y presentación que sería deseable mantener una vez culminado el proceso de extracción.

Uno de los primeros problemas que se presentan en la extracción es saber que porción del texto extraer luego de que se detecta una ocurrencia de la clase en la página HTML relevante. Por ejemplo, se podría extraer toda la oración donde aparece la ocurrencia de la clase, se podría decidir extraer solo la palabra anterior y posterior en la oración que aparece la ocurrencia de la clase, etc. En esta propuesta se maneja el concepto de nodo "contenedor", es decir, se extrae todo el contenido del nodo donde aparece la ocurrencia de la clase. Con los nodos contenedores se pretende extraer el contexto en el cual aparece la ocurrencia de la clase. Por nodo contenedor, se definen los tags HTML:

- TITLE
- TABLE
- P
- DL
- OL
- UL
- H1, H2, H3, H4, H5

Como ocurrencia de la clase se debe de considerar no solo el nombre de la clase sino la etiqueta de la clase, los sinónimos de dicha clase (para ello se utiliza un Diccionario de Sinónimos tal cual se explicó en la sección 3.2.1), así como también los posibles valores que las propiedades de la clase puedan tomar. Para saber que valores puede tomar una propiedad, en la definición de la ontología, se debe de asociar a cada propiedad, o bien una lista con los posibles valores, o bien una expresión regular que ayude a la identificación de las mismas dentro del documento.

El concepto de nodo contenedor significa que si se detecta una clase de la ontología dentro del código HTML se busca dentro de cual de los tags anteriores está contenida la clase y se extrae el contenido de todo el tag. Se pretende extraer el contexto en el cual aparece la clase. El resto de los tags HTML, que no son contenedores, son tratados como parte del texto a ser extraído. Por ejemplo, si una clase relevante de la ontología aparece dentro de una celda de una tabla (dentro de un tag "TD"), entonces de acuerdo al concepto de nodo contenedor, se extrae toda la tabla. Se supone que si una celda tiene información relevante, entonces las otras celdas de una forma u otra tratarán acerca del tema de interés. Lo mismo sucede con el resto de los nodos contenedores. Por ejemplo con respecto al nodo "P" lo que se pretende es obtener todo el párrafo que trata sobre el tema de interés. De esta forma se asegura que se extrae el contexto en el cual aparece la clase de interés.

Muchas fuentes HTML utilizan tablas para armar la presentación de la misma, con lo que el heurístico de extracción debería de reconocer cuando una tabla es utilizada para presentación y cuando es utilizada para mostrar datos en forma tabular ya que en el primer caso no interesa extraer el contenido del nodo "TABLE", mientras que en el segundo caso si interesa. Cuando la tabla es utilizada para armar el layout de la página, el concepto de nodo contenedor "TABLE" deja de ser válido. En caso de determinar que la tabla se utiliza para armar el layout de la página, la tabla, como estructura, es descartada por el heurístico de extracción. Cabe aclarar, que cuando se decide descartar una tabla, lo que se hace es descartar la estructura de tabla, y quedarse con el contenido de cada celda y analizarlo para saber si se debe de extraer información o no. El heurístico propone que cuando se está frente a una tabla de este tipo, se descarte la estructura de tabla y cada celda (tag "TD") sea considerada como un párrafo (tag "P").

Por lo tanto, un punto importante a resolver durante la extracción es determinar cuando una tabla se utiliza para presentar datos en forma de grilla y cuando la tabla se usa para armar el layout de la página.

A continuación se presenta una propuesta para determinar cuando se está frente a una tabla para desplegar datos o frente a una tabla utilizada para armar el layout de la página.

Paso 1 – Presentación de la Tabla

Como primer punto a ser analizado en una tabla, es el hecho de que se utilicen tags HTML para darle presentación a la tabla. Es decir, se debe de investigar la utilización o no del tag "TH".

Quienes escriben páginas HTML pueden o no conocer de HTML y los diferentes tags que el lenguaje provee, por lo tanto si un usuario está escribiendo una tabla y utiliza el tag "TH" se puede decir que el usuario conoce de HTML y sabe que ese tag se debe de utilizar en los cabezales de tablas que presentan información en forma de grilla. Quien usa tablas para layout nunca utiliza el tag "TH".

Por lo tanto como primer punto del heurístico, se explora la existencia o no del tag "TH" en la tabla. Si existe, se afirma que la tabla es para presentar datos en forma de grilla y no es descartada.

Si no se utiliza el tag "TH", a priori no se puede inferir nada acerca de la tabla. Se debe de continuar explorando la tabla.

Paso 2 – Contenido de la Celda

Este paso toma en cuenta el largo del contenido de la celda. Se basa en el supuesto de que si la tabla se utiliza para presentar datos en forma de grilla, entonces el contenido de la celda es por lo general un valor o una pequeña frase.

Este paso propone contar la cantidad de caracteres que hay dentro de cada celda (tag "TD") y sumarlos. Al resultado se lo divide entre la cantidad de celdas, de esta forma se obtiene el largo promedio de cada celda. Si dicho número es inferior a cierto valor (relativamente pequeño, por ejemplo 200 caracteres o inferior), entonces se puede afirmar que se está frente a una tabla que se utiliza para presentar datos en forma de grilla y no frente a una tabla utilizada para armar la presentación de la página. La tabla no se descarta.

Paso 3 – Descarte de la Estructura de Tabla

En caso de no caer dentro de ninguno de los pasos anteriores, no se puede afirmar nada acerca de la tabla y se opta por suponer que la tabla es utilizada para armar la presentación, por lo tanto se descarta la estructura de la misma.

Para descartar la tabla lo que se hace es que sustituir cada celda (tag "TD") por un párrafo (tag "P") y se eliminan los tags "TABLE" y "TR".

A continuación se presenta un ejemplo en donde la tabla es descartada:

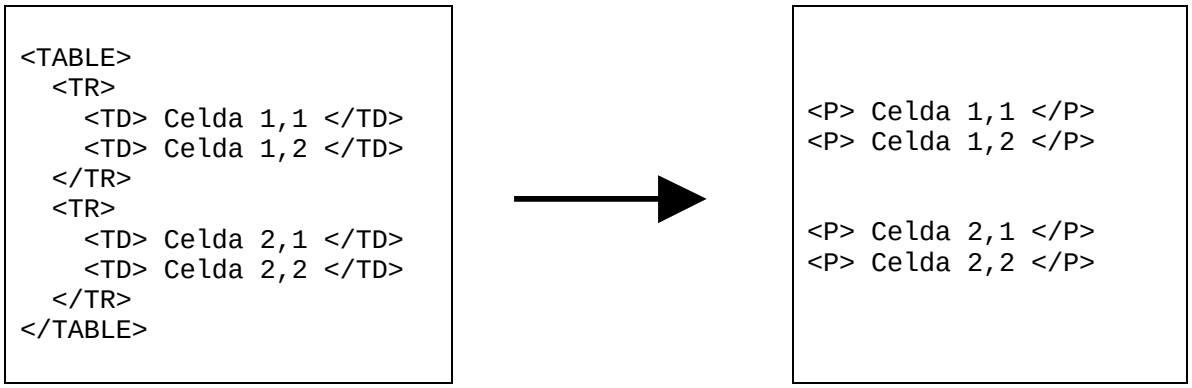

Figura 7 – Ejemplo de Extracción de Tabla

Propuesta de Heurístico

A continuación se presentan los pasos a seguir por el heurístico de extracción propuesto.

Paso 1

Para cada tabla (tag "TABLE") de la página HTML, se determina si es una tabla para desplegar datos o es una tabla utilizada para armar la presentación de la página. En este último caso la tabla es descartada.

El hecho de que existan tablas anidadas, es decir, que una celda contenga a otra tabla, no es relevante a los efectos de este paso.

Paso 2

Para cada nodo contenedor se considera la distancia que existe entre dicho nodo y el nodo raíz del documento XML, es decir, la cantidad de nodos que hay en el recorrido del árbol desde el nodo contenedor hasta el nodo raíz.

Se consideran todos los nodos contenedores cuya distancia a la raíz es la menor. Para cada uno de estos nodos, el heurístico investiga si existe alguna ocurrencia de alguna clase relevante. A los efectos de ver la ocurrencia de cada clase dentro del nodo, no se considera el contenido de los nodos contenedores que están anidados dentro del nodo contenedor considerado.

En caso de que se detecte una ocurrencia de alguna clase, se extrae todo el contenido del nodo contenedor, incluyendo los nodos contenedores anidados, en caso de existir.

En caso de que no se detecte ninguna ocurrencia de ninguna clase dentro del nodo contenedor, se vuelve a ejecutar lo expuesto en este paso, pero esta vez para todos los nodos contenedores anidados que están dentro del nodo considerado y cuya distancia al nodo raíz es la menor.

Cabe destacar que de la forma en que está propuesta la recorrida y extracción de los nodos contenedores no puede darse el caso de que una misma porción del documento HTML sea extraída más de una vez.

Puede suceder el caso de que en la página HTML considerada no exista ningún nodo contenedor. Como se sabe que la página es relevante, se opta por extraer el contenido de la misma en forma íntegra.

Ejemplo de Propuesta de Heurístico

A continuación se presenta un ejemplo de cómo se aplicarían los pasos propuestos en el heurístico de extracción. Se considera la siguiente página HTML:

```
<HTML>
  <BODY>
    <P>
      <A HREF=...>...</A>
      <H1>...</H1>
      <P>...</P>
    </P>
    <DL>...</DL>
  </BODY>
</HTML>
```

Figura 8 – Ejemplo de Página HTML

Se consideran los nodos contenedores cuya distancia a la raíz (tag "HTML") es la menor, es decir, primero se procesa el nodo contenedor:

```
    <P>
      <A HREF=...>...</A>
      <H1>...</H1>
      <P>...</P>
    </P>
```

Figura 9 – Primer Nodo Contenedor de Menor Distancia

Una vez finalizado el procesamiento del nodo contenedor anterior, se procesa el otro nodo contenedor cuya distancia a la raíz es la menor:

```
    <DL>...</DL>
```

Figura 10 – Segundo Nodo Contenedor de Menor Distancia

Para el primer nodo contenedor a procesar, se considera únicamente el tag "A" para ver si hay ocurrencias de las clases relevantes. Por lo dicho en la propuesta del heurístico, los nodos contenedores anidados ("H1" y "P") no son considerados para ver si existen ocurrencias de las clases. Se supone que en el tag "A" no hay ocurrencias de ninguna clase relevante, si lo hubiera se extra el contenido íntegro del nodo contenedor "P" incluyendo los nodos contenedores anidados ("H1" y "P"). Según lo expuesto en el heurístico, se deben de considerar los nodos contenedores anidados cuya distancia a la raíz es la menor.

Por lo tanto se consideran los nodos contenedores:

```
<H1>...</H1>
<P>...</P>
```

Figura 11 – Nodos Contenedores de Figura 9

Para los efectos de este ejemplo se supone que en ambos contenedores hay ocurrencias de las clases relevantes, por lo tanto ambos nodos contenedores son extraídos íntegramente, de lo contrario se vuelve a iterar sobre cada uno de los nodos contenedores.

Ahora es el turno de procesar el nodo contenedor de la figura 10:

```
<DL>...</DL>
```

Figura 12 – Segundo Nodo Contenedor de Menor Distancia (Figura 10)

Para los efectos de este ejemplo se supone que en el nodo contenedor hay ocurrencias de las clases relevantes, por lo tanto el nodo contenedor es extraído íntegramente.

El heurístico de extracción obtiene como resultado la siguiente extracción:

```
<H1>...</H1>
<P>...</P>
<DL>...</DL>
```

Figura 13 – Resultado de Extracción

3.3 Salida

A medida que el heurístico de extracción va extrayendo los nodos contenedores de interés, los mismos se van concatenando con los anteriores nodos contenedores. De esta manera se asegura que en el documento resultante se respeta el orden en que los nodos contenedores aparecen en el documento fuente.

Como el nodo contenedor se extrae de un documento con estructura de XML, el contenido del nodo contenedor es un documento XML y por lo tanto al ir concatenando todos lo nodos contenedores que se van extrayendo, se obtiene un documento de salida en formato XML. Este documento de salida contiene a todos los nodos contenedores que se fueron extrayendo con el heurístico visto en el punto 3.2.4.

Para respetar la estructura de los documentos XML, es necesario que el nodo raíz del documento XML sea un nodo único que no contenga nodos hermanos, para ello todo lo extraído en la sección anterior se anida dentro del tag “EXTRACCION”.

Continuando con el ejemplo visto en la sección 3.2.4, el resultado de la extracción sería el documento XML que se presenta a continuación:

```
<EXTRACCION>
    <H1>...</H1>
    <P>...</P>
    <DL>...</DL>
</EXTRACCION>
```

Figura 14 – Archivo de Salida

Sobre el documento de salida se puede realizar consultas XML de todo tipo utilizando el lenguaje XPath [18], como por ejemplo, obtener solamente los párrafos del documento extraído, información que se obtiene recorriendo los nodos XML denominados “P”. Otra consulta que se podría hacer sobre el documento extraído es, obtener todos los enlaces, información que se obtiene recorriendo los nodos XML denominados “A”.

4 Ejemplo

En esta sección se presenta un ejemplo de como funciona el wrapper según lo visto en la sección anterior.

4.1 Usuario

El ejemplo se basa en una ontología que está declarada en base al funcionamiento de un Departamento de Computación. La misma está accesible en la Web en la siguiente URL: http://www.cs.umd.edu/projects/plus/DAML/onts/cs1.0.daml. A esta ontología se le agregó la clase "Locate" para hacer más interesante el ejemplo presentado. Esta clase tiene un atributo llamado "City" que toma el valor "New York".

A continuación se presenta un esquema gráfico de la ontología. En el documento "Documentación de Tesis - Expansión de Ejemplo.doc" se presenta la ontología escrita en RDF [19].

Cabe aclarar que la clase "Locate" es subclase de la clase inicial "Root", sin embargo, en el dibujo no se representa para no agregar más flechas y que por ende quede poco entendible.

Esquema de la Ontología:

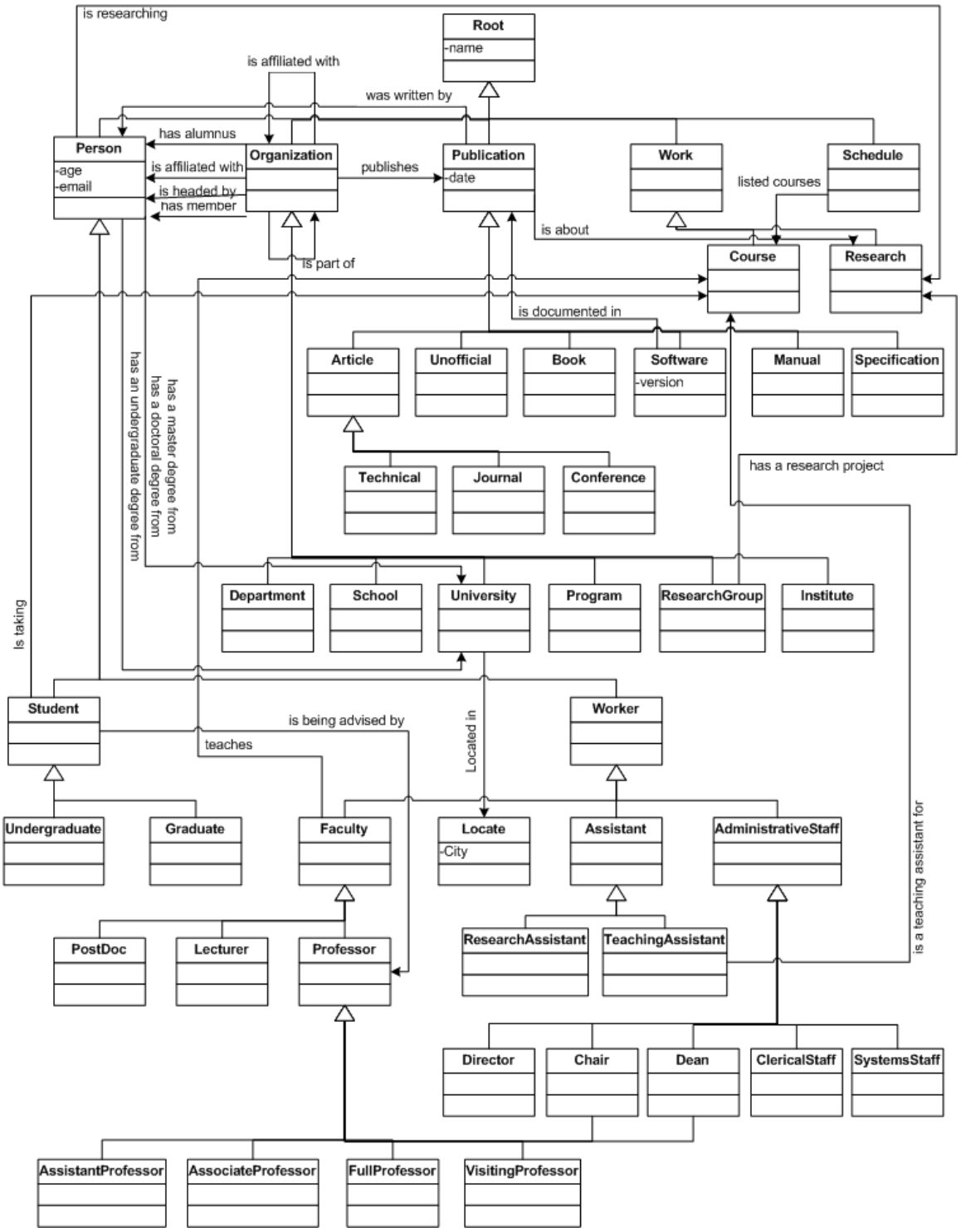

Figura 15 – Ontología

A los efectos de este ejemplo, la consulta que se formula es la siguiente:

- *Professors of the Computer Science Department from a University located in New York*

En base a la consulta anterior, se realiza una búsqueda en Internet utilizando motores de búsqueda habituales. Debido a la alta popularidad de Google (http://www.google.com), se recomienda utilizar dicho buscador como punto de partida. La cadena de búsqueda que se introdujo fue la siguiente:

- Professors of the Computer Science Department from a University located in New York

Cabe destacar que no se utilizó ningún carácter especial de Google. El resultado de búsqueda que presentó Google fue el siguiente:

- http://www.cs.buffalo.edu/info-people.shtml
- http://www.cs.nyu.edu/web/People/faculty.html
- http://www.cs.buffalo.edu/
- http://www.cs.jhu.edu/
- http://www.cs.jhu.edu/news_.html
- http://www.cs.columbia.edu/
- http://www.cs.columbia.edu/faculty_traub.html
- http://www.cs.fredonia.edu/
- http://www-cs.engr.ccny.cuny.edu/
- http://www.wordiq.com/definition/University_of_Pennsylvania

Las anteriores direcciones que presentó Google, corresponden a la primera página del resultado. Google presentó más de 10 páginas de resultado.

4.2 Generador de Wrappers

Esta etapa es la encargada de realizar todos los pasos necesarios para poder generar el wrapper y poder extraer la información que el usuario necesita.

El ejemplo se presenta siguiendo el mismo orden de la sección 3.

4.2.1 Identificación de Conceptos Relevantes de la Ontología

La primer tarea que el wrapper debe de realizar es una correspondencia entre la consulta y la ontología. Es decir, a partir de la consulta se seleccionan las clases y relaciones de interés de la ontología que son relevantes para la extracción.

Según lo propuesto en la sección 3, los pasos a seguir para determinar la correspondencia de las clases son los siguientes:

Se buscan todas las palabras de la consulta que aparezcan directamente en la ontología, ya sea como nombre de clase, etiqueta de clase o valor de alguna propiedad de la clase o nombre de relación entre clases de la ontología.

De la consulta se deduce que las palabras que aparecen literalmente en la ontología son:

Clases:
- University
- Department

Relaciones:
- Located in

Valores:
- Locate (el atributo "City" tiene el valor "New York")

De cada palabra de la consulta se obtiene una lista de sinónimos y para cada sinónimo se procede como se indicó en el punto anterior.

En este paso se utiliza el diccionario de sinónimos de inglés propuesto en la sección 3 (http://www.cogsci.princeton.edu/~wn/). Cada palabra de la oración se introduce en el diccionario y se obtienen los sinónimos.

Palabra	Sinónimo
Professors	Professor
of	-
the	-
Computer	computing machine, computing device, data processor, electronic computer, information processing system
Science	scientific discipline
Department	section
From	-
A	-
University	-
Located	locate, place, site
in	-
New	modern, fresh, young
York	-

Tabla 3 – Tabla de Sinónimos

Los sinónimos que aparecen en la ontología son:

- Professor
- Locate

Para cada clase de la ontología que se seleccionó en el punto a, se consideran como relevantes todas las subclases que dependan de esta, es decir, se considera todo el árbol cuya raíz es la clase en cuestión. Las relaciones que puedan existir en este subárbol no son tomadas en cuenta.

De la ontología se deducen las siguientes subclases de las clases seleccionadas en los dos puntos anteriores.

- University
- Locate
- Department
- Professor
  - AssistantProfessor
  - AssociateProfessor
  - FullProfessor
  - VisitingProfessor
  - Chair
  - Dean

De las relaciones surgidas en el punto a, se consideran como relevantes las clases que son extremo de la relación, es decir, las clases que definen la relación. Para cada una de estas dos clases, se aplica lo dicho en el punto c.

La relación surgida anteriormente es “located in”. Las clases que son extremo de esta relación son:

- University
- Locate

Por lo tanto consideramos estas dos clases y todas las subclases que se derivan a partir de estas dos. En este caso particular ninguna de estas dos clases tiene subclases.

Resumiendo este paso, se obtuvo que las clases que se corresponden con la consulta ingresada son:

- University
- Locate
- Department
- Professor
- AssistantProfessor
- AssociateProfessor
- FullProfessor
- VisitingProfessor
- Chair
- Dean

A continuación se extrae del diagrama gráfico de la ontología las clases anteriormente mencionadas. Las clases que se considerarán de ahora en más son:

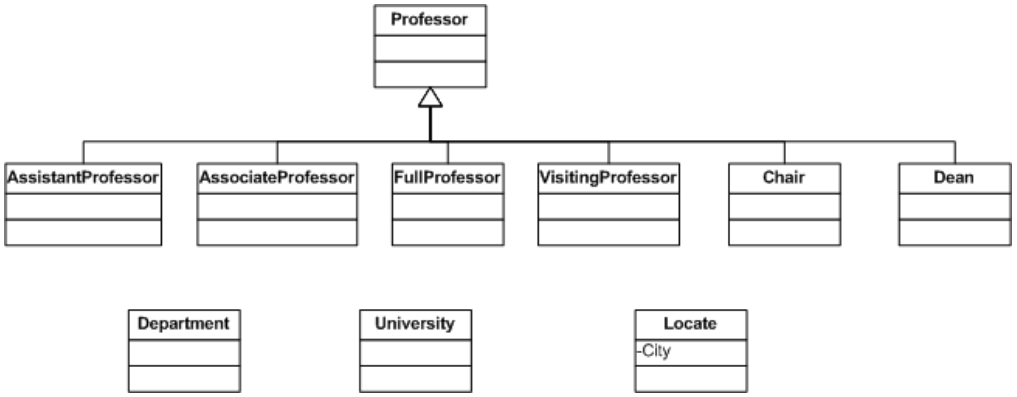

Figura 16 – Clases Relevantes

4.2.2 Selección de Páginas HTML Relevantes

En base a las clases relevantes de la ontología, obtenidas en el punto anterior, las páginas HTML suministradas por el usuario y la frecuencia de aparición de las clases en los documentos, también suministradas por el usuario, el wrapper debe de seleccionar aquellas páginas HTML relevantes, es decir, se eliminan aquellas páginas HTML que no tratan sobre el tema de interés. Para ello, se utiliza un modelo que ayuda a determinar cuan similares son dos documentos (la ontología y la página HTML en este caso). El modelo tiene el nombre de Vector Space Model (VSM). Como salida se obtienen las páginas HTML que son relevantes para el tema tratado, las que no son relevantes se descartan.

Para simplificar la tarea del usuario, éste puede ingresar las frecuencias esperadas con que debe ocurrir cada clase en el documento HTML, solo de las clases obtenidas en el punto 4.2.1, es decir, solo de las clases que son relevantes según la consulta ingresada.

A continuación se presenta una tabla con las frecuencias esperadas de ocurrencia de cada clase, las cuales son ingresadas por el usuario.

Clase	Frecuencia Esperada
University	0.9
Locate	0.3
Department	0.05
Professor	0.99
AssistantProfessor	0.7
AssociateProfessor	0.8
FullProfessor	0
VisitingProfessor	0.01
Chair	0.02
Dean	0.1

Tabla 4 – Frecuencia Esperada de las Clases

Siguiendo los pasos indicados en la sección 3.2.2

El modelo propone armar dos vectores. El primero de ellos, llamado OV, indica para cada clase relevante de la ontología cual es la frecuencia esperada con que debe de aparecer la clase en el documento HTML.

OV = (<University : FrecEsperada1>, <Locate : FrecEsperada2>, <Department : …>, <Professor : …. >, <AssistantProfessor : ….. >, <AssociateProfessor : ….. >, <FullProfessor: …. >, <VisitingProfessor : ….. >, <Chair : …. >, <Dean : …. >)

El segundo vector que propone el modelo, llamado DV, contiene para las clases relevantes, la cantidad de veces que aparecen efectivamente en el documento HTML. Como ocurrencia del concepto se debe de considerar no solo el nombre de la clase sino la etiqueta de la clase, los sinónimos de dicha clase (para ello se utiliza un Diccionario de Sinónimos tal cual se explicó en la sección 3.2.1), así como también los posibles valores que las propiedades de la clase puedan tomar. Para saber que valores puede tomar una propiedad, en la definición de la ontología, se debe de asociar a cada propiedad, o bien una lista con los posibles valores, o bien una expresión regular que ayude a la identificación de las mismas dentro del documento.

A continuación se presenta para cada clase, la etiqueta correspondiente, los sinónimos (obtenidos de http://www.cogsci.princeton.edu/~wn/) y los valores de las propiedades. Es decir, que encontrando cualquier de estas palabras, se puede afirmar que se está ante una ocurrencia de la clase.

Clase	Etiqueta	Sinónimo	Valores Propiedades
University	University	-	-
Locate	Locate	place, site	New York
Department	Department	Section	-
Professor	Professor	-	-
AssistantProfessor	Assistant Professor	-	-
AssociateProfessor	Associate Professor	-	-
FullProfessor	Full Professor	-	-
VisitingProfessor	Visiting Professor	-	-
Chair	Chair	-	-
Dean	Dean	-	-

Tabla 5 – Sinónimos y Valores de las Clases

A los efectos de este ejemplo se considera la primera página del resultado que se obtuvo con Google. Utilizando esta página como ejemplo, se le aplicará el VSM para determinar si es una página HTML relevante o no.

La página considerada es la siguiente: http://www.cs.buffalo.edu/info-people.shtml. En el documento "Documentación de Tesis - Expansión de Ejemplo.doc" se presenta el código de la página HTML con estructura de XML.

Tomando en cuenta la tabla anterior, para cada clase se cuenta efectivamente cuantas veces ocurre en la página HTML, es decir, para cada clase se busca dentro de la página HTML cuantas veces aparece cada Clase, Etiqueta, Sinónimo y Valores de Propiedades.

Clase	Ocurrencias
University	47
Locate	3
Department	1
Professor	70
AssistantProfessor	23
AssociateProfessor	22
FullProfessor	0
VisitingProfessor	0
Chair	4
Dean	1

Tabla 6 – Ocurrencias de las Clases

Es importante destacar que las palabras "assistant professor" y "associate professor" encontradas en el documento HTML, cuentan como ocurrencia para las clases "AssistantProfessor" y "AssociateProfessor" respectivamente, así como para la clase "Professor".

De la tabla anterior se obtiene el vector DV que queda de la siguiente forma:

DV = (<University : 47>, <Locate : 3>, <Department : 1>, <Professor : 70>, <AssistantProfessor : 23>, <AssociateProfessor : 22>, <FullProfessor: 0>, <VisitingProfessor : 0>, <Chair : 4>, <Dean : 1>)

Se calcula la función SIM(D,O) de la siguiente manera:

$$SIM(D, O) = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2 \sum_j y_j^2}}$$

Figura 17 – Función Similitud Desarrollada

Siendo x_i los valores con se espera que aparezcan las clases en la ontología y y_i la cantidad de veces que aparece la clase en la fuente consultada.

X = (0.9 , 0.3 , 0.05 , 0.99 , 0.7 , 0.8 , 0 , 0.01 , 0.02 , 0.1)
Y = (47 , 3 , 1 , 70 , 23 , 22 , 0 , 0 , 4 , 1)

Figura 18 – Valores de los Vectores de Función Similitud

Calculando la función SIM para el ejemplo se tiene que:

SIM (D, O) = 0.93293556

Figura 19 – Resultado de Función Similitud

Es decir, el documento anterior tiene más de un 93 por ciento de certeza que trata el mismo tema que la ontología.

El usuario debe de ingresar un porcentaje mínimo para el cual acepta que la página HTML es relevante según la función SIM propuesta en el algoritmo VSM (Vector Space Model).

A los efectos de este ejemplo se asume que si la función SIM es superior a 0.7, es decir 70 por ciento, entonces la página HTML es considerada como relevante y no es descartada.

En el ejemplo anterior la página HTML es considerada relevante ya que la función SIM dio un valor de 0.93334259.

Puede darse el caso en que todas las páginas HTML sean descartadas ya que la función SIM no alcanza el valor mínimo. En este caso se sugiere utilizar otro buscado diferente a Google, como por ejemplo Yahoo (http://www.yahoo.com) o Altavista (http://www.altavista.com).

4.2.3 Transformación de HTML a XML

Para solucionar este problema, la página HTML se pasa por Tidy tal cual se indica en el punto 3.2.3, para obtener una página bien escrita (desde el punto de vista de estructura de HTML), es decir, todos los tags tienen su homólogo que los cierra. De esta forma el documento queda bien estructurado y es manejable desde el punto de vista de algoritmos.

En el documento "Documentación de Tesis - Expansión de Ejemplo.doc" se presenta se presenta el código de la página HTML (http://www.cs.buffalo.edu/info-people.shtml) con estructura de XML, luego de haber sido pasado por Tidy.

4.2.4 Heurístico de Extracción

Finalmente con las páginas XML relevantes y junto con las clases de la ontología relevantes, se procede a aplicar un heurístico que extrae la información deseada.

A continuación se presenta como sería el heurístico de extracción propuesto.

Paso 1

Lo primero a realizar en el heurístico es descartar todas las tablas que son utilizadas para armar la presentación de la página.

Se analizan únicamente algunas tablas de la página HTML. En el documento "Documentación de Tesis - Expansión de Ejemplo.doc" se presenta la página HTML completa sin las tablas que son utilizadas para presentación.

Tabla 1

El hecho de que esta tabla tenga dos tablas anidadas, no es relevante ya que cada tabla se investiga por separado.

```
<table width="100%" border="0" cellpadding="0" cellspacing="0"
bgcolor="#003366">
      <tr>
            <td>
                  <a href="http://www.buffalo.edu/"><img
src="/images/ub-logo.gif" width="434" height="56" border="0"
alt="University at Buffalo, The State University of New York"></a>
            </td>
            <td align="right">
                  <table cellpadding="8" cellspacing="0" border="0">
                        <tr>
                              <td>
                                    <form
action="http://www.cse.buffalo.edu/cgi-bin/htsearch" method="GET">
                                    <table cellpadding="0"
cellspacing="4" border="0">
                                          <tr><td><img
src="/images/search-cse.gif" width="63" height="5" alt="Search
CSE"></td></tr>
                                          <tr><td><input type="text"
style="width: 100px" size="15" name="words"></td></tr>
                                          <tr><td align="right"><input
type="image" src="/images/go-button.gif" width="22" height="5" alt="Go"
name="go" border="0"></td></tr>
                                    </table>
                                    </form>
                              </td>
                        </tr>
                  </table>
            </td>
      </tr>
</table>
```

Figura 20 – Extracto de Página HTML

Se observa que la tabla no contiene el tag “TH” por lo que no se puede afirmar nada de la tabla en este paso.

La tabla posee dos celdas. La primera celda contiene 171 caracteres y la segunda celda contiene 588 caracteres. En promedio cada celda contiene 379.5 caracteres. Si se supone que un número razonable de largo promedio para una tabla de tipo de presentación deberían de ser pocos caracteres. A los efectos de este ejemplo se toma como largo promedio mayor el valor 300.

Como el largo promedio de cada celda es mayor que 300 caracteres, se supone que se está frente a una tabla de presentación, por lo tanto se descarta la tabla.

Para descartar la tabla lo que se hace es sustituir cada celda (tag “TD”) por un párrafo (tag “P”) y se eliminan los tags “TABLE” y “TR”.

Por lo tanto el código HTML queda de la siguiente manera:

```
            <p>
                  <a href="http://www.buffalo.edu/"><img
src="/images/ub-logo.gif" width="434" height="56" border="0"
alt="University at Buffalo, The State University of New York"></a>
            </p>
            <p>
                  <table cellpadding="8" cellspacing="0" border="0">
                        <tr>
                              <td>
                                    <form
action="http://www.cse.buffalo.edu/cgi-bin/htsearch" method="GET">
                                    <table cellpadding="0"
cellspacing="4" border="0">
                                          <tr><td><img
src="/images/search-cse.gif" width="63" height="5" alt="Search
CSE"></td></tr>
                                          <tr><td><input type="text"
style="width: 100px" size="15" name="words"></td></tr>
                                          <tr><td align="right"><input
type="image" src="/images/go-button.gif" width="22" height="5" alt="Go"
name="go" border="0"></td></tr>
                                    </table>
                                    </form>
                              </td>
                        </tr>
                  </table>
            </p>
```

Figura 21 – Extracción de Tabla

Ahora se considera la tabla anidada de la instancia anterior. Notar que esta tabla, a su vez contiene otra tabla anidada.

```
                  <table cellpadding="8" cellspacing="0" border="0">
                        <tr>
                              <td>
                                    <form
action="http://www.cse.buffalo.edu/cgi-bin/htsearch" method="GET">
                                    <table cellpadding="0"
cellspacing="4" border="0">
                                          <tr><td><img
src="/images/search-cse.gif" width="63" height="5" alt="Search
CSE"></td></tr>
                                          <tr><td><input type="text"
style="width: 100px" size="15" name="words"></td></tr>
                                          <tr><td align="right"><input
type="image" src="/images/go-button.gif" width="22" height="5" alt="Go"
name="go" border="0"></td></tr>
                                    </table>
                                    </form>
                              </td>
                        </tr>
                  </table>
```

Figura 22 – Tabla Anidada

Se observa que la tabla no contiene el tag “TH” por lo que no se puede afirmar nada de la tabla en este paso.

La tabla posee una celda. La celda contiene 491 caracteres. Tal cual se supuso anteriormente, las tablas que no son utilizadas para presentación deben de tener en promedio 300 caracteres o menos.

En este caso el largo promedio es 491 caracteres que son mayores que 300 caracteres. Se supone que se está frente a una tabla de presentación, por lo tanto se descarta la tabla.

Para descartar la tabla lo que se hace es que sustituir cada celda (tag “TD”) por un párrafo (tag “P”) y se eliminan los tags “TABLE” y “TR”.

Por lo tanto el código HTML queda de la siguiente manera:

```
                                <p>
                                      <form
action="http://www.cse.buffalo.edu/cgi-bin/htsearch" method="GET">
                                      <table cellpadding="0"
cellspacing="4" border="0">
                                            <tr><td><img
src="/images/search-cse.gif" width="63" height="5" alt="Search
CSE"></td></tr>
                                            <tr><td><input type="text"
style="width: 100px" size="15" name="words"></td></tr>
                                            <tr><td align="right"><input
type="image" src="/images/go-button.gif" width="22" height="5" alt="Go"
name="go" border="0"></td></tr>
                                      </table>
                                      </form>
                                </p>
```

Figura 23 – Extracción de Tabla

Ahora se considera la tabla que se encuentra dentro del tag “FORM”. Notar que esta tabla, no contiene tablas anidadas.

```
                                      <table cellpadding="0"
cellspacing="4" border="0">
                                            <tr><td><img
src="/images/search-cse.gif" width="63" height="5" alt="Search
CSE"></td></tr>
                                            <tr><td><input type="text"
style="width: 100px" size="15" name="words"></td></tr>
                                            <tr><td align="right"><input
type="image" src="/images/go-button.gif" width="22" height="5" alt="Go"
name="go" border="0"></td></tr>
                                      </table>
```

Figura 24 – Tabla Anidada

Se observa que la tabla no contiene el tag “TH” por lo que no se puede afirmar nada de la tabla en este paso.

La tabla posee tres celdas. La primera celda contiene 95 caracteres, la segunda celda contiene 63 caracteres y la tercera celda contiene 100 caracteres. En promedio cada celda tiene 86 caracteres. Tal cual se supuso anteriormente, las tablas que no son utilizadas para presentación deben de tener en promedio 300 caracteres o menos. Por lo tanto esta tabla se asume que no es utilizada para presentación y por lo tanto no se descarta.

Por lo tanto el código HTML queda de la siguiente manera:

```
                                    <table cellpadding="0"
cellspacing="4" border="0">
                                          <tr><td><img
src="/images/search-cse.gif" width="63" height="5" alt="Search
CSE"></td></tr>
                                          <tr><td><input type="text"
style="width: 100px" size="15" name="words"></td></tr>
                                          <tr><td align="right"><input
type="image" src="/images/go-button.gif" width="22" height="5" alt="Go"
name="go" border="0"></td></tr>
                                    </table>
```

Figura 25 – Extracción de Tabla

Resumiendo, de la tabla presentada anteriormente y de los descartes de tablas de presentación, el código HTML queda de la siguiente manera:

```
            <p>
                  <a href="http://www.buffalo.edu/"><img
src="/images/ub-logo.gif" width="434" height="56" border="0"
alt="University at Buffalo, The State University of New York"></a>
            </p>
            <p>
                              <p>
                                    <form
action="http://www.cse.buffalo.edu/cgi-bin/htsearch" method="GET">
                                    <table cellpadding="0"
cellspacing="4" border="0">
                                          <tr><td><img
src="/images/search-cse.gif" width="63" height="5" alt="Search
CSE"></td></tr>
                                          <tr><td><input type="text"
style="width: 100px" size="15" name="words"></td></tr>
                                          <tr><td align="right"><input
type="image" src="/images/go-button.gif" width="22" height="5" alt="Go"
name="go" border="0"></td></tr>
                                    </table>
                                    </form>
                              </p>
            </p>
```

Figura 26 – Resultado de Extracción de Tablas de Figura 20

Tabla 2

En este caso también existen tablas anidadas, pero para lo efectos de determinar si cada una es utilizada para presentación o no, es irrelevante.

```
<table width="100%" border="0" cellpadding="0" cellspacing="0"><tr>
      <td class="page_header2" bgcolor="#CCCC66"><p>
            <a class="csehomelink" href="index.shtml">Computer Science
and Engineering</a>
      </p></td>
      <td align="right" class="page_header2_bg" bgcolor="#CCCC66">
            <table cellpadding="0" cellspacing="0" border="0"><tr>
                  <td><img src="/images/dots-red.gif" width="10"
height="15" alt=""></td>
                        <td class="category_title_small"><p><a
href="/apply.shtml">Apply</a></p></td>
                  <td><img src="/images/blank.gif" width="6"
height="15" alt=""></td>
                  <td><img src="/images/dots-blue.gif" width="10"
height="15" alt=""></td>
                        <td class="category_title_small"><p><a
href="/explore-prospective.shtml">
                              Prospective Students
                        </a></p></td>
                  <td><img src="/images/blank.gif" width="6"
height="15" alt=""></td>
                  <td><img src="/images/dots-blue.gif" width="10"
height="15" alt=""></td>
                        <td class="category_title_small"><p><a
href="/research.shtml">
                              Research
                        </a></p></td>
                  <td><img src="/images/blank.gif" width="6"
height="15" alt=""></td>
                  <td><img src="/images/dots-blue.gif" width="10"
height="15" alt=""></td>
                        <td class="category_title_small"><p><a
href="/academics-general.shtml">
                              Academics
                        </a></p></td>
                  <td><img src="/images/blank.gif" width="6"
height="15" alt=""></td>
                  <td><img src="/images/dots-blue.gif" width="10"
height="15" alt=""></td>
                        <td class="category_title_small"><p><a
href="/">
                              Home
                        </a></p></td>
                  <td><img src="/images/blank.gif" width="6"
height="15" alt=""></td>
            </tr></table>
      </td>
</tr></table>
```

Figura 27 – Extracto de Página HTML

Se observa que la tabla no contiene el tag “TH” por lo que no se puede afirmar nada de la tabla en este paso.

La tabla posee dos celdas. La primera celda contiene 90 caracteres y la segunda celda contiene 1290 caracteres. En promedio cada celda contiene 690 caracteres. En promedio cada celda es mayor a 300 caracteres por lo tanto la tabla es considerada como de presentación y se descarta.

Para descartar la tabla lo que se hace es que sustituir cada celda (tag “TD”) por un párrafo (tag “P”) y se eliminan los tags “TABLE” y “TR”.

Por lo tanto el código HTML queda de la siguiente manera:

```
      <p><p>
            <a class="csehomelink" href="index.shtml">Computer Science
and Engineering</a>
      </p></p>
      <p>
            <table cellpadding="0" cellspacing="0" border="0"><tr>
                  <td><img src="/images/dots-red.gif" width="10"
height="15" alt=""></td>
                        <td class="category_title_small"><p><a
href="/apply.shtml">
                              Apply
                        </a></p></td>
                  <td><img src="/images/blank.gif" width="6"
height="15" alt=""></td>
                  <td><img src="/images/dots-blue.gif" width="10"
height="15" alt=""></td>
                        <td class="category_title_small"><p><a
href="/explore-prospective.shtml">
                              Prospective Students
                        </a></p></td>
                  <td><img src="/images/blank.gif" width="6"
height="15" alt=""></td>
                  <td><img src="/images/dots-blue.gif" width="10"
height="15" alt=""></td>
                        <td class="category_title_small"><p><a
href="/research.shtml">
                              Research
                        </a></p></td>
                  <td><img src="/images/blank.gif" width="6"
height="15" alt=""></td>
                  <td><img src="/images/dots-blue.gif" width="10"
height="15" alt=""></td>
                        <td class="category_title_small"><p><a
href="/academics-general.shtml">
                              Academics
                        </a></p></td>
                  <td><img src="/images/blank.gif" width="6"
height="15" alt=""></td>
                  <td><img src="/images/dots-blue.gif" width="10"
height="15" alt=""></td>
                        <td class="category_title_small"><p><a
href="/">
                              Home
                        </a></p></td>
                  <td><img src="/images/blank.gif" width="6"
height="15" alt=""></td>
            </tr></table>
      </p>
```

Figura 28 – Extracción de Tabla

Ahora se considera la tabla anidada de la instancia anterior. Notar que esta tabla, no contiene tablas anidadas.

```
            <table cellpadding="0" cellspacing="0" border="0"><tr>
                  <td><img src="/images/dots-red.gif" width="10"
height="15" alt=""></td>
                        <td class="category_title_small"><p><a
href="/apply.shtml">
                              Apply
                        </a></p></td>
                  <td><img src="/images/blank.gif" width="6"
height="15" alt=""></td>
                  <td><img src="/images/dots-blue.gif" width="10"
height="15" alt=""></td>
                        <td class="category_title_small"><p><a
href="/explore-prospective.shtml">
                              Prospective Students
                        </a></p></td>
                  <td><img src="/images/blank.gif" width="6"
height="15" alt=""></td>
                  <td><img src="/images/dots-blue.gif" width="10"
height="15" alt=""></td>
                        <td class="category_title_small"><p><a
href="/research.shtml">
                              Research
                        </a></p></td>
                  <td><img src="/images/blank.gif" width="6"
height="15" alt=""></td>
                  <td><img src="/images/dots-blue.gif" width="10"
height="15" alt=""></td>
                        <td class="category_title_small"><p><a
href="/academics-general.shtml">
                              Academics
                        </a></p></td>
                  <td><img src="/images/blank.gif" width="6"
height="15" alt=""></td>
                  <td><img src="/images/dots-blue.gif" width="10"
height="15" alt=""></td>
                        <td class="category_title_small"><p><a
href="/">
                              Home
                        </a></p></td>
                  <td><img src="/images/blank.gif" width="6"
height="15" alt=""></td>
            </tr></table>
```

Figura 29 – Tabla Anidada

Se observa que la tabla no contiene el tag “TH” por lo que no se puede afirmar nada de la tabla en este paso.

La tabla posee quince celdas. En promedio cada celda tiene 60 caracteres. Tal cual se supuso anteriormente, las tablas que no son utilizadas para presentación deben de tener en promedio 300 caracteres o menos. Por lo tanto esta tabla se asume que no es utilizada para presentación y por lo tanto no se descarta.

Resumiendo, de la tabla presentada anteriormente y de los descartes de tablas de presentación, el código HTML queda de la siguiente manera:

```
      <p><p>
            <a class="csehomelink" href="index.shtml">Computer Science
and Engineering</a>
      </p></p>
      <p>
            <table cellpadding="0" cellspacing="0" border="0"><tr>
                  <td><img src="/images/dots-red.gif" width="10"
height="15" alt=""></td>
                        <td class="category_title_small"><p><a
href="/apply.shtml">
                              Apply
                        </a></p></td>
                  <td><img src="/images/blank.gif" width="6"
height="15" alt=""></td>
                  <td><img src="/images/dots-blue.gif" width="10"
height="15" alt=""></td>
                        <td class="category_title_small"><p><a
href="/explore-prospective.shtml">
                              Prospective Students
                        </a></p></td>
                  <td><img src="/images/blank.gif" width="6"
height="15" alt=""></td>
                  <td><img src="/images/dots-blue.gif" width="10"
height="15" alt=""></td>
                        <td class="category_title_small"><p><a
href="/research.shtml">
                              Research
                        </a></p></td>
                  <td><img src="/images/blank.gif" width="6"
height="15" alt=""></td>
                  <td><img src="/images/dots-blue.gif" width="10"
height="15" alt=""></td>
                        <td class="category_title_small"><p><a
href="/academics-general.shtml">
                              Academics
                        </a></p></td>
                  <td><img src="/images/blank.gif" width="6"
height="15" alt=""></td>
                  <td><img src="/images/dots-blue.gif" width="10"
height="15" alt=""></td>
                        <td class="category_title_small"><p><a
href="/">
                              Home
                        </a></p></td>
                  <td><img src="/images/blank.gif" width="6"
height="15" alt=""></td>
            </tr></table>
      </p>
```

Figura 30 – Extracción de Tabla

El resto de las tablas se procesan de forma similar a las vistas hasta ahora. En el documento "Documentación de Tesis - Expansión de Ejemplo.doc" se presenta el código HTML de la página, habiendo descartado todas las tablas que son utilizadas para presentación.

Paso 2

Según lo expuesto en el heurístico:

Se consideran todos los nodos contenedores cuya distancia a la raíz es la menor. Para cada uno de estos nodos, el heurístico investiga si existe alguna ocurrencia de alguna clase relevante. A los efectos de ver la ocurrencia de cada clase dentro del nodo, no se considera el contenido de los nodos contenedores que están anidados dentro del nodo contenedor considerado.

A continuación se presenta el código HTML de la página, pero solo mostrando los nodos contenedores que están a la menor distancia de la raíz (tag "HTML"):

```
<HTML>
<HEAD>
  <TITLE>University at Buffalo: Computer Science andEngineering</TITLE>
  <META http-equiv="Content-Type" content="text/html; charset=iso-8859-
1" />
  <LINK rel="STYLESHEET" type="text/css" href="/styles/default.css" />
</HEAD>
<BODY  marginwidth="0"  marginheight="0"  topmargin="0"  leftmargin="0"
bgcolor="#EEEEEE">
  <P>...</P>
</BODY>
</HTML>
```

Figura 31 – Nodos Contenedores de Menor Distancia

Los nodos contenedores más cercanos a la raíz son: el nodo "TITLE" y el nodo "P". Según lo expuesto en el heurístico, primero se procesa el nodo contenedor "TITLE" y luego se procesa el nodo contenedor "P".

Nodo Contenedor 1.

```
...
<TITLE>University at Buffalo: Computer Science andEngineering</TITLE>
...
```

Figura 32 –Nodo Contenedor

Para este nodo se buscan ocurrencias de las clases. Las ocurrencias que cada clase puede tomar fueron presentadas en la sección 4.2.2.

En este caso se detecta que la clase "University" ocurre dentro del nodo contenedor, por lo tanto se extrae todo el contenido del nodo (se marca en negrita la ocurrencia de la clase):

```
<TITLE>University at Buffalo: Computer Science and
Engineering</TITLE>
```

Figura 33 – Ocurrencia de Clases

Nodo Contenedor 2

Se procesa el siguiente nodo contenedor de menor distancia a la raíz.

```
  <P>...</P>
```

Figura 34 – Nodo Contenedor

Expandiendo el código HTML de este nodo contenedor se obtiene lo siguiente:

```
<P>
  <IMG src="/images/770px_Spacer.gif" width="770" height="1" alt="" />
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <TABLE cellpadding="0" cellspacing="0" border="0" width="100%"
bgcolor="#FFFFFF">...</TABLE>
  <TABLE width="100%" border="0" cellpadding="0"
cellspacing="0">...</TABLE>
  <TABLE width="100%" border="0" cellpadding="0"
cellspacing="0">...</TABLE>
</P>
```

Figura 35 – Expansión de Nodos Contenedores

Este nodo contenedor posee otros nodos contenedores anidados. Según lo expuesto en el heurístico se buscan ocurrencias de las clases dentro del nodo contenedor pero sin considerar los nodos contenedores anidados. Por lo tanto se debe de buscar ocurrencias de las clases solo en el código:

```
  <IMG src="/images/770px_Spacer.gif" width="770" height="1" alt="" />
```

Figura 36 – Nodo HTML

En este código HTML no existe ninguna ocurrencia de ninguna clase. En caso de que existiese una ocurrencia entonces se extrae todo el nodo contenedor, incluyendo a los nodos contenedores anidados y se terminaría de procesar el documento HTML.

Como no existe ninguna ocurrencia de ninguna clase en el nodo contenedor (sin considerar los nodos contenedores anidados), se procesan por separado los nodos contenedores anidados que están a menor distancia de la raíz. Por lo tanto los nuevos nodos contenedores a procesar son:

```
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <P>...</P>
  <TABLE cellpadding="0" cellspacing="0" border="0" width="100%"
bgcolor="#FFFFFF">...</TABLE>
  <TABLE width="100%" border="0" cellpadding="0"
cellspacing="0">...</TABLE>
  <TABLE width="100%" border="0" cellpadding="0"
cellspacing="0">...</TABLE>
```

Figura 37 – Nodos Contenedores

A continuación se desarrollan cada uno de estos nodos contenedores (“Nodo Contenedor 2.1” corresponde al primero de ellos y “Nodo Contenedor 2.2” corresponde al segundo y así sucesivamente).

Nodo Contenedor 2.1

Expandiendo el primer nodo contenedor anteriormente presentado se obtiene el siguiente código HTML:

```
<P>
  <A href="http://www.buffalo.edu/">
    <IMG  src="/images/ub-logo.gif"  width="434"  height="56"  border="0"
alt="University at Buffalo, The State University of New York" />
  </A>
</P>
```

Figura 38 – Nodo Contenedor

Para este nodo se buscan ocurrencias de las clases relevantes obtenidas en la sección 4.2.1. Las ocurrencias que cada clase puede tomar fueron presentadas en la sección 4.2.2.

En este caso se detecta que la clase “University” ocurre dos veces dentro del nodo contenedor y la clase “Locate” ocurre una vez (el texto “New York” corresponde a un valor que toma la propiedad “City” de la clase “Locate” por lo tanto es una ocurrencia de la clase “Locate”). Al existir ocurrencias de clases, se extrae el nodo contenedor completo. Se marca en negrita la ocurrencia de la clase.

```
<P>
  <A href="http://www.buffalo.edu/">
    <IMG  src="/images/ub-logo.gif"  width="434"  height="56"  border="0"
alt="University at Buffalo, The State University of New York" />
  </A>
</P>
```

Figura 39 – Ocurrencia de Clase

Nodo Contenedor 2.2

Expandiendo el segundo nodo contenedor anteriormente presentado se obtiene el siguiente código HTML:

```
<P>
  <P>...</P>
</P>
```

Figura 40 – Nodo Contenedor

Este nodo contenedor está compuesto únicamente de nodos contenedores. Según el heurístico propuesto, hay que buscar ocurrencias de las clases dentro del nodo contenedor sin considerar los nodos nodos contenedores anidados. Por lo tanto, no existen ocurrencias de ninguna clase dentro de este nodo contenedor. Este nodo contenedor es descartado y se debe de analizar cada uno de los nodos contenedores anidados, que están a menor distancia de la raíz, por separado para saber si se deben de extraer o no.

Para seguir cierto orden en la numeración de los nodos contenedores a procesar, se enumera de la siguiente manera: "Nodo Contenedor 2.2.1" al nodo contenedor anidado.

Nodo Contenedor 2.2.1

El código HTML de este nodo contenedor es el siguiente:

```
<P>
  <FORM action="http://www.cse.buffalo.edu/cgi-bin/htsearch"
method="GET">
    <TABLE cellpadding="0" cellspacing="4" border="0">...</TABLE>
  </FORM>
</P>
```

Figura 41 – Nodo Contenedor

Este nodo contenedor está compuesto únicamente del nodo "FORM" que no es un nodo contenedor. El nodo "FORM" es tratado como cualquier otro nodo que no es contenedor, en el sentido de que se buscan ocurrencias dentro del nodo sin considerar los nodos contenedores que estén contenidos. Por lo tanto, no existen ocurrencias de ninguna clase dentro de este nodo. Este nodo es descartado y se debe de analizar cada uno de los nodos contenedores anidados, que están a menor distancia de la raíz, por separado para saber si se deben de extraer o no. Este nodo contenedor es denominado "Nodo Contenedor 2.2.1.1".

Nodo Contenedor 2.2.1.1

El código HTML de este nodo contenedor es el siguiente:

```
<TABLE cellpadding="0" cellspacing="4" border="0">
  <TR>
    <TD>
      <IMG src="/images/search-cse.gif" width="63" height="5"
alt="Search CSE" />
    </TD>
  </TR>
  <TR>
    <TD>
      <INPUT type="text" style="width: 100px" size="15" name="words" />
    </TD>
  </TR>
  <TR>
    <TD align="right">
      <INPUT type="image" src="/images/go-button.gif" width="22"
height="5" alt="Go" name="go" border="0" />
    </TD>
  </TR>
</TABLE>
```

Figura 42 – Nodo Contenedor

En el código HTML no existen ocurrencias de las clases, por lo tanto el nodo contenedor es descartado.

Cabe destacar que en este punto se ha terminado de procesar todo el Nodo Contenedor 2.2, el cual como se vio anteriormente no contiene ocurrencias de las clases relevantes vistas en la sección 4.2.1, por lo tanto el Nodo Contenedor 2.2 es descartado.

Continuando con el procesamiento de los restantes Nodos Contenedores:

Nodo Contenedor 2.3

El código HTML de este nodo contenedor es el siguiente:

```
<P>
  <P>...</P>
</P>
```

Figura 43 – Nodo Contenedor

Este nodo contenedor está compuesto únicamente de nodos contenedores. Según el heurístico propuesto, hay que buscar ocurrencias de las clases dentro del nodo contenedor sin considerar los nodos nodos contenedores anidados. Por lo tanto, no existen ocurrencias de ninguna clase dentro de este nodo contenedor. Este nodo contenedor es descartado y se debe de analizar cada uno de los nodos contenedores anidados, que están a menor distancia de la raíz, por separado para saber si se deben de extraer o no. Este nodo contenedor es denominado "Nodo Contenedor 2.3.1".

Nodo Contenedor 2.3.1

El código HTML de este nodo contenedor es el siguiente:

```
<P>
  <A class="csehomelink" href="index.shtml">Computer Science and
Engineering</A>
</P>
```

Figura 44 – Código HTML de Nodo Contenedor

En el código HTML no existen ocurrencias de las clases, por lo tanto el nodo contenedor es descartado.

Cabe destacar que en este punto se ha terminado de procesar todo el Nodo Contenedor 2.3, el cual como se vio anteriormente no contiene ocurrencias de las clases relevantes vistas en la sección 4.2.1, por lo tanto el Nodo Contenedor 2.3 es descartado.

Se continúa con el procesamiento del Nodo Contenedor 2.2.20. El resto de los Nodos Contenedores, se procesan de manera muy similar a los vistos hasta ahora. De todas, formas en el documento "Documentación de Tesis - Expansión de Ejemplo.doc" se presenta el documento completo resultante de la extracción.

Nodo Contenedor 2.21

El código HTML de este nodo contenedor es el siguiente:

```
<P>
  <P class="header">...</P>
  <UL class="body">...</UL>
  <P class="header1">...</P>
  <P class="header2">...</P>
  <DL class="body">...</DL>
  <P class="header1">...</P>
  <DL class="body">...</DL>
  <P class="header1">...</P>
  <DL class="body">...</DL>
  <P class="header1">...</P>
  <DL class="body">...</DL>
  <P class="header1">...</P>
  <DL class="body">...</DL>
  <P class="header1">...</P>
  <P class="header1">...</P>
  <DL class="body">...</DL>
  <P class="header1">...</P>
  <DL class="body">...</DL>
  <P class="header1">...</P>
  <DL class="body">...</DL>
  <P class="header1">...</P>
  <P class="header1">...</P>
  <DL class="body">...</DL>
  <P class="header1">...</P>
  <DL class="body">...</DL>
  <P class="header1">...</P>
  <P class="header1">...</P>
  <DL class="body">...</DL>
  <P>...</P>
  <DL class="body">...</DL>
  <P class="header1">...</P>
  <DL class="body">...</DL>
  <P class="header1">...</P>
  <DL class="body">...</DL>
  <P class="header1">...</P>
  <DL class="body">...</DL>
</P>
```

Figura 45 – Nodo Contenedor

Este nodo contenedor está compuesto únicamente de nodos contenedores. Según el heurístico propuesto, hay que buscar ocurrencias de las clases dentro del nodo contenedor sin considerar los nodos nodos contenedores anidados. Por lo tanto, no existen ocurrencias de ninguna clase dentro de este nodo contenedor. Este nodo contenedor es descartado y se debe de analizar cada uno de los nodos contenedores anidados, que están a menor distancia de la raíz, por separado para saber si se deben de extraer o no.

Para seguir cierto orden en la numeración de los nodos contenedores a procesar, cada uno de ellos se enumera de la siguiente manera: "Nodo Contenedor 2.21.1", "Nodo Contenedor 2.21.2", "Nodo Contenedor 2.21.3", etc.

A continuación se presenta el desarrollo de algunos nodos contenedores.

Nodo Contenedor 2.21.10

El código HTML de este nodo contenedor es el siguiente:

```
<P class="header1">Associate Professors</P>
```

Figura 46 – Nodo Contenedor

En este caso se detecta que la clase "AssociateProfessor" ocurre dentro del nodo contenedor (es la etiqueta de la clase la que está presente en el nodo contenedor), por lo tanto se extrae todo el contenido del nodo (se marca en negrita la ocurrencia de la clase):

```
<P class="header1">Associate Professors</P>
```

Figura 47 – Ocurrencia de Clases

Nodo Contenedor 2.21.11

El código HTML de este nodo contenedor es el siguiente:

```
<DL class="body">
  <DT>
    <A href="faculty/chomicki/">
      <B>Jan Chomicki</B>
    </A>
  </DT>
  <DD>Associate Professor</DD>
  <DD>(Ph.D., Rutgers University)</DD>
  <DD>Databases - integrity and interoperability; policy management,
electronic commerce and agent-based systems; datawarehousing</DD>
  <DT>
    <A href="faculty/df33/">
      <B>Daniel Fischer</B>
    </A>
  </DT>
  <DD>Associate Professor</DD>
  <DD>(Ph.D., TelAviv University, Israel)</DD>
  <DD>Bioinformatics, Computational Structural Molecular Biology,
Protein Fold Recognition, Protein Structure Comparison</DD>
  <DT>
    <A href="faculty/qiao/">
      <B>Chunming Qiao</B>
    </A>
  </DT>
  <DD>Associate Professor</DD>
```

```
  <DD>Adjunct Associate Professor of Electrical Engineering</DD>
  <DD>(Ph.D.,University of Pittsburgh)</DD>
  <DD>Computer Communication Networks, Parallel and Distributed
Processing, Optical Communications</DD>
  <DT>
    <A href="faculty/rapaport/">
      <B>William J.Rapaport</B>
    </A>
  </DT>
  <DD>Associate Professor</DD>
  <DD>Adjunct Professor of Philosophy</DD>
  <DD>(Ph.D., Indiana University)</DD>
  <DD>
    <A href="/sneps/">Artificial Intelligence</A>
    ,Computational Linguistics,
    <A href="/cogsci/">Cognitive Science</A>
    ,Philosophical Issuesin Computer Science
  </DD>
  <DT>
    <A href="faculty/regan/">
      <B>Kenneth W.Regan</B>
    </A>
  </DT>
  <DD>Associate Professor</DD>
  <DD>Director of Graduate Studies</DD>
  <DD>(Ph.D., Oxford University)</DD>
  <DD>
    <A href="research-algorithms.shtml">Theoretical Computer
Science</A>
  </DD>
  <DT>
    <A href="faculty/peter/">
      <B>Peter D.Scott</B>
    </A>
  </DT>
  <DD>Associate Professor</DD>
  <DD>Adjunct Associate Professor of Electrical Engineering</DD>
  <DD>Adjunct Associate Professor of Physiology and Biophysics</DD>
  <DD>(Ph.D., Cornell University)</DD>
  <DD>Controls, Signals and Systems</DD>
  <DT>
    <A href="/%7Ersridhar/">
      <B>Ramalingam Sridhar</B>
    </A>
  </DT>
  <DD>Associate Professor</DD>
  <DD>Adjunct Associate Professor of Electrical Engineering</DD>
  <DD>(Ph.D., Washington State University, Pullman)</DD>
  <DD>Computer Architecture. VLSI systems</DD>
  <DT>
    <A href="http://www.cedar.buffalo.edu/%7Erohini/">
      <B>Rohini K.Srihari</B>
    </A>
  </DT>
  <DD>Associate Professor</DD>
  <DD>Adjunct Associate Professor of Electrical Engineering</DD>
  <DD>Director of Undergraduate Studies</DD>
```

```
  <DD>(Ph.D., University at Buffalo)</DD>
  <DD>Multimedia Information Retrieval, Multimodal Interfaces,
Computacional Linguistics, Context-basedVision</DD>
  <DT>
    <A href="/%7Eshambhu/">
      <B>Shambhu Upadhyaya</B>
    </A>
  </DT>
  <DD>Associate Professor</DD>
  <DD>Adjunct Associate Professor of Electrical Engineering</DD>
  <DD>(Ph.D., Univ. of Newcastle, NSW, Australia)</DD>
  <DD>Fault-Tolerant Computing (Hardware/Software), VLSI Testing,
Distributed Systems, Computer Security</DD>
</DL>
```

Figura 48 – Nodo Contenedor

El nodo contenedor presentado (“DL”) no contiene nodos contenedores anidados.

Para este nodo se buscan ocurrencias de las clases relevantes obtenidas en la sección 4.2.1. Las ocurrencias que cada clase puede tomar fueron presentadas en la sección 4.2.2.

En este caso hay varias clases que ocurren en el nodo contenedor. Por ejemplo, se detecta que la clase “University” ocurre varias veces dentro del nodo contenedor, la clase “AssociateProfessor” también ocurre varias veces (en este caso se encuentra como ocurrencia la etiqueta de la clase que es el texto “Associate Professor”) y la clase “Professor” también ocurre varias veces dentro del nodo contenedor.

Como dentro del nodo contenedor se encontraron ocurrencias de algunas clases, se extrae todo el nodo contenedor.

El resto de los nodos contenedores se procesan de forma similar a los presentados hasta ahora. De todas, formas en el documento “Documentación de Tesis - Expansión de Ejemplo.doc” se presenta el documento completo resultante de la extracción.

4.3 Salida

A medida que el heurístico de extracción va extrayendo los nodos contenedores de interés, los mismos se van concatenando con los anteriores nodos contenedores. De esta manera se asegura que en el documento resultante se respeta el orden en que los nodos contenedores aparecen en el documento fuente.

Como el nodo contenedor se extrae de un documento con estructura de XML, el contenido del nodo contenedor es un documento XML y por lo tanto al ir concatenando todos lo nodos contenedores que se van extrayendo, se obtiene un documento de salida en formato XML. Este documento de salida contiene a todos los nodos contenedores que se fueron extrayendo en la sección 4.2.

Para respetar la estructura de los documentos XML, es necesario que el nodo raíz del documento XML sea un nodo único que no contenga nodos hermanos, para ello todo lo extraído en la sección anterior se anida dentro del tag "EXTRACCION".

Continuando con el ejemplo, el resultado de la extracción sería el documento XML que se presenta a continuación. En el documento "Documentación de Tesis - Expansión de Ejemplo.doc" se presenta el documento completo resultante de la extracción.

```
<EXTRACCION>
  <TITLE>University at Buffalo: Computer Science and
Engineering</TITLE>
  <P>
    <A href="http://www.buffalo.edu/">
      <IMG src="/images/ub-logo.gif" width="434" height="56" border="0"
alt="University at Buffalo, The State University of New York" />
    </A>
  </P>
  ...
  <P class="header1">Associate Professors</P>
  <DL class="body">
    <DT>
      <A href="faculty/chomicki/">
        <B>Jan Chomicki</B>
      </A>
    </DT>
    <DD>Associate Professor</DD>
    <DD>(Ph.D., Rutgers University)</DD>
    <DD>Databases - integrity and interoperability; policy management,
electronic commerce and agent-based systems; datawarehousing</DD>
    <DT>
      <A href="faculty/df33/">
        <B>Daniel Fischer</B>
      </A>
    </DT>
    <DD>Associate Professor</DD>
    <DD>(Ph.D., TelAviv University, Israel)</DD>
    <DD>Bioinformatics, Computational Structural Molecular Biology,
Protein Fold Recognition, Protein Structure Comparison</DD>
    <DT>
```

```
      <A href="faculty/qiao/">
        <B>Chunming Qiao</B>
      </A>
    </DT>
    <DD>Associate Professor</DD>
    <DD>Adjunct Associate Professor of Electrical Engineering</DD>
    <DD>(Ph.D.,University of Pittsburgh)</DD>
    <DD>Computer Communication Networks, Parallel and Distributed
Processing, Optical Communications</DD>
    <DT>
      <A href="faculty/rapaport/">
        <B>William J.Rapaport</B>
      </A>
    </DT>
    <DD>Associate Professor</DD>
    <DD>Adjunct Professor of Philosophy</DD>
    <DD>(Ph.D., Indiana University)</DD>
    <DD>
      <A href="/sneps/">Artificial Intelligence</A> ,Computational
Linguistics,
      <A href="/cogsci/">Cognitive Science</A> ,Philosophical Issuesin
Computer Science
    </DD>
    <DT>
      <A href="faculty/regan/">
        <B>Kenneth W.Regan</B>
      </A>
    </DT>
    <DD>Associate Professor</DD>
    <DD>Director of Graduate Studies</DD>
    <DD>(Ph.D., Oxford University)</DD>
    <DD>
      <A href="research-algorithms.shtml">Theoretical Computer
Science</A>
    </DD>
    <DT>
      <A href="faculty/peter/">
        <B>Peter D.Scott</B>
      </A>
    </DT>
    <DD>Associate Professor</DD>
    <DD>Adjunct Associate Professor of Electrical Engineering</DD>
    <DD>Adjunct Associate Professor of Physiology and Biophysics</DD>
    <DD>(Ph.D., Cornell University)</DD>
    <DD>Controls, Signals and Systems</DD>
    <DT>
      <A href="/%7Ersridhar/">
        <B>Ramalingam Sridhar</B>
      </A>
    </DT>
    <DD>Associate Professor</DD>
    <DD>Adjunct Associate Professor of Electrical Engineering</DD>
    <DD>(Ph.D., Washington State University, Pullman)</DD>
    <DD>Computer Architecture. VLSI systems</DD>
    <DT>
      <A href="http://www.cedar.buffalo.edu/%7Erohini/">
        <B>Rohini K.Srihari</B>
```

```
      </A>
    </DT>
    <DD>Associate Professor</DD>
    <DD>Adjunct Associate Professor of Electrical Engineering</DD>
    <DD>Director of Undergraduate Studies</DD>
    <DD>(Ph.D., University at Buffalo)</DD>
    <DD>Multimedia Information Retrieval, Multimodal Interfaces,
Computacional Linguistics, Context-basedVision</DD>
    <DT>
      <A href="/%7Eshambhu/">
        <B>Shambhu Upadhyaya</B>
      </A>
    </DT>
    <DD>Associate Professor</DD>
    <DD>Adjunct Associate Professor of Electrical Engineering</DD>
    <DD>(Ph.D., Univ. of Newcastle, NSW, Australia)</DD>
    <DD>Fault-Tolerant Computing (Hardware/Software), VLSI Testing,
Distributed Systems, Computer Security</DD>
  </DL>
  ...
</EXTRACCION>
```

Figura 49 – Archivo de Salida

5 Resultados Experimentales

En este capítulo se evalúa la propuesta realizada contra la extracción manual. Para seguir en la misma línea de análisis se presentan los resultados que se obtienen utilizando la propuesta, tomando como entrada la ontología y consulta utilizada como ejemplo en el capítulo 4. En esta ocasión se consideran los 14 primeros resultados que se obtienen utilizando Google.

5.1 Selección de Páginas HTML Relevantes

En esta sección se evalúan los resultados obtenidos por el algoritmo encargado de seleccionar las páginas HTML relevantes, es decir, el algoritmo de Vector Space Model (VSM).

En la columna "Relevante (manual)", se indica el resultado de realizar la selección de páginas HTML relevantes de manera manual, es decir, lo que uno esperaría como resultado si realiza el proceso simplemente observando las páginas HTML. En la columna "Relevante (propuesta)", se indica el resultado que se obtuvo mediante la propuesta realizada en este trabajo. A los efectos de estas pruebas se considera que la página HTML es relevante si su función de similitud supera el valor 0.7. En la columna "Función Similitud" se expresa el valor devuelto luego de aplicar el algoritmo de Vector Space Model sobre el documento HTML.

	URL	Relevante (manual)	Relevante (propuesta)	Función Similitud
1	http://www.cs.buffalo.edu/info-people.shtml	SI	SI	0.93
2	http://www.cs.nyu.edu/web/People/faculty.html	SI	NO	0.61
3	http://www.cs.buffalo.edu/	SI	SI	0.72
4	http://www.cs.jhu.edu/	NO	NO	0.50
5	http://www.cs.jhu.edu/news_.html	SI	SI	0.71
6	http://www.cs.columbia.edu/	NO	NO	0.25
7	http://www.cs.columbia.edu/faculty_traub.html	SI	NO	0.55
8	http://www.cs.fredonia.edu/	NO	NO	0.29
9	http://www-cs.engr.ccny.cuny.edu/	NO	NO	0.49
10	http://www.wordiq.com/definition/University_of_Pennsylvania	NO	NO	0.68
11	http://web.grinnell.edu/careerdevelopment/jobs/computer&math.html	NO	NO	0.66
12	http://chronicle.com/jobs/profiles/2488.htm?pg=i	NO	NO	0.51
13	http://www.panix.com/clay/nyc/schools.shtml	NO	NO	0.51
14	http://web.gc.cuny.edu/dept/psych/subprogs/excog_1.html	NO	NO	0.50

Tabla 7 – Resultados Experimentales de Páginas HTML Relevantes

De los resultados obtenidos se puede observar que las páginas HTML seleccionadas por Vector Space Model caen dentro de las páginas que uno esperaría que el algoritmo seleccione. De todas formas, existen dos páginas que uno esperaría que el algoritmo seleccione y éste no las seleccionó debido a que su función de similitud (SIM) no superó el valor 0.7 que era el mínimo aceptable para considerar a la página como relevante.

Estos resultados son muy dependientes de los valores ingresados por el usuario, referentes a la frecuencia con que debe de aparecer cada clase de la ontología, así como al valor aceptado por la función de similitud (SIM) del algoritmo Vector Space Model. También es importante explotar en forma correcta el diccionario de sinónimos, así como una lista de posibles valores que pueda tomar cada clase dentro del dominio estudiado.

Cabe destacar que el módulo de "Selección de Páginas HTML Relevantes" ha sido utilizado por [20] como una caja negra, es decir, que ha sido utilizado sin realizarle modificaciones a lo que se planteó en este trabajo. Los resultados obtenidos han sido clasificados como aceptables. En el documento "Proceso de Clasificación" del mencionado trabajo, se explican los resultados obtenidos y su grado de aceptación.

5.2 Heurístico de Extracción

Cabe destacar que en esta extracción se incluyeron como relevantes las URLs: 1, 2, 3, 5 y 7.

La columna "Extracción Aceptable" se refiere al hecho de si se extrajeron de la página HTML todos los nombres de profesores que había, sin importar si en la extracción se trajo información que no era relevante. El concepto "aceptable" refiere a que el heurístico no haya dejado de extraer información relevante.

	URL	Extracción Aceptable
1	http://www.cs.buffalo.edu/info-people.shtml	SI
2	http://www.cs.nyu.edu/web/People/faculty.html	NO
3	http://www.cs.buffalo.edu/	SI
5	http://www.cs.jhu.edu/news_.html	SI
7	http://www.cs.columbia.edu/faculty_traub.html	SI

Tabla 8 – Resultados Experimentales del Heurístico de Extracción

En el caso de la URL número 2, el heurístico falló debido a que en dicha página se utiliza el concepto "faculty" para referenciar a los profesores. Incluyendo como sinónimo de la clase "Professor" a la palabra "faculty", el problema queda resuelto.

Cabe destacar que para ejecutar estas pruebas, no se ha explotado en toda su potencia los diccionarios de sinónimos ni tampoco se utilizó en forma exhaustiva una lista de posibles valores que pueda tomar cada clase dentro del dominio estudiado.

6 Conclusiones

La técnica de construcción de wrappers basados en ontologías resuelve el problema de la ambigüedad de las palabras utilizadas para definir y referirse a los conceptos de un dominio. Esta técnica, además, se adapta con facilidad a los cambios que las páginas HTML puedan sufrir.

Dado que el trabajo se focaliza en la extracción de información sobre páginas HTML, el wrapper propuesto toma como base la técnica basada en ontologías y la extiende utilizando el significado de los diferentes ítems (tags) del lenguaje HTML de modo de incrementar la inteligencia al momento de realizar la extracción.

Para la construcción del módulo de extracción se presentó un heurístico de extracción que mantiene la estructura original del documento en formato XML, por lo tanto se obtiene el beneficio de que sobre el resultado final se pueden realizar consultas de tipo XML utilizando XPath.

Quienes escriben las páginas HTML pocas veces tienen en cuenta el vocabulario común definido en la ontología para el dominio tratado y por lo tanto utilizan palabras ambiguas para definir y referirse a los conceptos desarrollados. Por esta razón es importante hacer un buen manejo de un diccionario de sinónimos, tema que es desarrollado en [15].

El aporte más interesante que realiza este trabajo fue presentar un proceso de extracción para construir un wrapper guiado por la consulta del usuario y basado en ontologías, así como una propuesta concreta de heurístico de extracción de información a ser utilizada por el wrapper.

En el heurístico se propone una metodología para extraer la información de interés de la página HTML basado en el concepto de "nodo contenedor", el cual permite extraer el contexto donde aparece la información relevante y además, se mantiene la estructura y orden de aparición de dicha información dentro de la página HTML fuente. En dicho heurístico también se presenta una propuesta que ayuda determinar cuando una tabla se utiliza como parte de la presentación de la página o cuando es utilizada para presentar información en forma de grilla.

En el trabajo se han propuesto y desarrollado diferentes aspectos que hacen al proceso de extracción de información:

a. Se propuso y desarrolló una metodología para que dada una ontología se seleccionen solo las clases y relaciones que interesan a los efectos de resolver el problema planteado.
b. Se implementó el algoritmo Vector Space Model (propuesto por Salton) que permite decidir, de las páginas suministradas por el usuario, cuales contienen información para extraer y cuales son irrelevantes.

c. Se propuso y desarrolló una metodología para poder quitar de la página HTML las tablas que son utilizadas para armar la presentación de la página.
d. Se propuso y desarrolló un heurístico de extracción de información el cual se basa en el concepto de "Nodo Contenedor". Este concepto fue introducido en este trabajo y permite decidir que porción (contexto) de información de la página relevante se debe de extraer.
e. Se utilizó Tidy para corregir los tags HTML y por lo tanto obtener una página bien escrita desde el punto de vista del lenguaje HTML.

Sin embargo, los wrappers basados en ontologías presentan la dificultad al momento de escribir la ontología en si misma, la cual debe de ser provista por un experto en el dominio a ser tratado. El hecho de escribir ontologías escapa al alcance de este trabajo.

Se implementó un prototipo para la generación de wrappers presentada, el cual puede ser accedido desde http://www.decorar.com/sitios/alvaro/wrapper/wrapper.zip.

6.1 Trabajos a futuro.

Como trabajo a futuro se podría mejorar las heurísticas para determinar mediante algoritmos más precisos cuando una tabla es utilizada para armar la presentación de la página y cuando la tabla es utilizada para presentar información en formato de grilla. De esta forma el heurístico sería mas preciso al momento de decidir para que es utilizada la tabla que ese está analizando. En tal sentido se recomienda el trabajo realizado por Yalin Wang y Jianying Hu [21].

Otro trabajo a futuro interesante, sería el hecho de extender el wrapper para que pueda recibir como entrada otros tipos de fuentes diferentes al HTML, como por ejemplo, archivos PDF, archivos de correo electrónico, planillas de cálculo, etc.

7 Referencias

[1] **HTML.** W3C. http://www.w3.org/TR/html4/. Ultima visita a la página Agosto de 2004.

[2] **XML.** eXtensible Markup Languaje. http://www.w3.org/XML/. Ultima visita a la página Agosto de 2004.

[3] Ricardo Baeza Yates y Cuauhtémoc Rivera Loaiza. **Ubicuidad y Usabilidad en la Web.** Centro de Investigación de la Web Departamento de Ciencias de la Computación, Universidad de Chile. Diciembre 2002.

[4] M. Chantal Pérez Hernández. **Estudios de Lingüística Española**. Universidad de Málaga, España. 2002.

[5] **A Brief Survey of Web Data Extraction Tools.** A. Laender and B. Ribeiro-Neto and A. Silva and J. Teixeira. SIGMOD Record (31) 2, June,2002.

[6] Gerald Huck, Peter Fankhauser, Karl Aberer, Erich Neuhold. **Jedi: Extracting and Synthesizing Information from the Web.** German National Research Center for Information Technology. Noviembre, 1998.

[7] **Document Object Model.** W3C. http://www.w3.org/DOM. Ultima visita a la página Agosto de 2004.

[8] Arnaud Sahuguet y Fabien Azavant. **Web Wrapper Factory (W4F).** http://db.cis.upenn.edu/DL/WWW8/. Ultima visita a la página Agosto de 2004.

[9] Mary Elaine Califf and Raymond J. Mooney. **Bottom-Up Relational Learning of Pattern Matching Rules for Information Extraction.** 2003.

[10] Nicholas Kushmerick. **Wrapper Induction: Efficiency and Expresiveness.** 2000.

[11] Brad Adelberg. **NoDoSE - A Tool for Semi Automatically Extracting Structured and Semistructured Data from Text Documents.** 1998.

[12] Gruber T. **Toward Principles for the Design of Ontologies Used for Knowledge Sharing**. Technical Report KSL-93-04, Knowledge Systems Laboratory, Stanford University, CA, 1993.

[13] Studer S, Benjamins R., Fensel D. **Knowledge Engineering: Principles and Methods**. Data and Knowledge Engineering, vol. 25, pp. 161-197, 1998.

[14] Embley y Campbell. **A Conceptual-Modeling Approach to Extracting Data from the Web.** 1999.

[15] Claudia Deco. **Propuesta de un Refinador Semántico para Recuperación de Información desde la Web**. Tésis de Maestría 2004. InCo, Facultad de Ingeniería, Universidad de la República. Uruguay.

[16] Salton, G. and McGill, J. M. (Eds.): **Introduction to Modern Information Retrieval**, McGraw-Hill, 1986.

[17] **Tidy:** http://www.w3.org/People/Raggett/tidy. Ultima visita a la página Agosto de 2004.

[18] **XPath.** XML Path Language. http://www.w3.org/TR/xpath. Ultima visita a la página Agosto de 2004.

[19] **RDF.** Resource Description Framework. http://www.w3.org/RDF/. Ultima visita a la página Agosto de 2004.

[20] Verónica Giaudrone, Marcelo Guerra y Marcelo Vaccaro. **Transformación e integración de información en una arquitectura de Web Warehouse**. Tésis de

Grado 2004. InCo, Facultad de Ingeniería, Universidad de la República. Uruguay. http://www.fing.edu.uy/~pgwebadw. Ultima visita a la página Noviembre de 2004.

[21] Yalin Wang y Jianying Hu. **A Machine Learning Based Approach for Table Detection on the Web**. Mayo 11, 2002, Honolulu, Hawaii, USA.