UNIVERSIDAD DE BELGRANO

Las tesinas de Belgrano

Facultad de Ingeniería y Tecnología Informática

Web mining: implementando técnicas de data mining en un servidor web

Nº 16 Gabriel H. Dandretta

Tutor: Alejandro A. Vaisman

Departamento de Investigación
Junio 2002

Abstract

El presente trabajo forma parte del proyecto de investigación de cátedra *«Utilización de Técnicas de Data Mining para Optimizar el Acceso a Páginas Web en un Servidor de Internet»*, proyecto que propone la utilización de técnicas de descubrimiento de información oculta en grandes Bases de Datos (Data Mining) a efectos de disminuir los tiempos de espera necesarios para cargar una determinada página en el momento de ser solicitada. Para ello se incorporó a la arquitectura tradicional, compuesta por una Intranet y un Servidor Proxy que brinda salida a Internet, un nuevo Servidor de Data Mining encargado de anticipar dinámicamente los requerimientos de páginas por parte de los usuarios en función de las páginas accedidas precedentemente, de modo de éstas se encuentren disponibles más rápidamente en el momento en que efectivamente se las requiera. Para permitir este funcionamiento conjunto presentamos aquí una interfase de comunicación entre los Servidores mencionados anteriormente. Se presentan diferentes alternativas de diseño y los resultados obtenidos en la implementación realizada.

Contenidos

1. Introducción

El presente trabajo forma parte del proyecto de investigación de cátedra «Utilización de Técnicas de Data Mining para Optimizar el Acceso a Páginas Web en un Servidor de Internet» orientado a la aplicación de técnicas de Data Mining en el contexto de los servidores de páginas Web, cuyo propósito es el de desarrollar una solución que brinde una mayor velocidad de accesos a las páginas publicadas. El contenido de este trabajo en particular es el de presentar el estado del arte en el tema y desarrollar e implementar una interfase que permita comunicar al servidor de Data Mining con el servidor Proxy.

La arquitectura del proyecto antes mencionado consta de tres componentes fundamentales:

* **Un servidor de Data Mining** que se ocupa de predecir las posibles próximas páginas a ser requeridas por los usuarios. Su construcción será realizada por algunos miembros del equipo de trabajo.
* **Un servidor Proxy** que encamina los pedidos de páginas Web desde la Intranet hacia Internet y se ocupa de administrar un espacio de memoria caché interna. Se utilizará un servidor comercial que cumpla estas funciones.
* **Una interfase** que permita interconectar los módulos antes mencionados, dando soporte a las necesidades de ambos y, al mismo tiempo, buscando la mayor eficiencia posible para contribuir al resultado total del sistema.

El contenido de este trabajo se refiere al desarrollo de una solución que implemente la interfase nombrada anteriormente y a la correcta puesta a punto del Servidor Proxy de modo de poder llevarse a cabo una correcta comunicación entre los elementos de la aplicación. Para ello, primero se hará referencia a conceptos tanto de Data Mining como a los Servidores Proxy, necesarios para comprender la naturaleza y características de la solución a implementar.

La limitación de los alcances de este Trabajo Final de Carrera responde a cuestiones de tiempo y de extensión del trabajo, ya que el presente es una labor individual y no grupal como lo es el proyecto completo. No obstante, en la conclusión se nombran futuras ampliaciones posibles, y caminos alternativos que podrán utilizarse como base para otras investigaciones.

Todos los desarrollos y cuestiones prácticas se realizan sobre entorno Win32: Microsoft Windows NT 4.0, Microsoft SQL Server 7.0 y Microsoft Proxy Server 2.0.

1.1 Objetivo

El propósito de este Trabajo Final de Carrera es conceptualizar, desarrollar e implementar la conexión entre el Servidor Proxy y el Servidor de Data Mining a utilizarse en el ya mencionado trabajo de investigación de cátedra. Además se pretende dar conceptos necesarios sobre las técnicas y tecnologías necesarias para llevar a cabo dicho objetivo.

1.2 Descripción del Sistema

El Sistema de Web Mining que se desarrolla tiene, como se dijo anteriormente, el objetivo de anticipar las próximas páginas web que serán accedidas por los usuarios, a partir de la información que se tiene registrada sobre los anteriores accesos.

La estructura convencional de salida a Internet de una Intranet puede verse como lo indica la Figura 1.1, donde todas las estaciones de trabajo se encuentran conectadas a Internet través de un Servidor Proxy. El funcionamiento de este servidor será estudiado en el Capítulo 3.

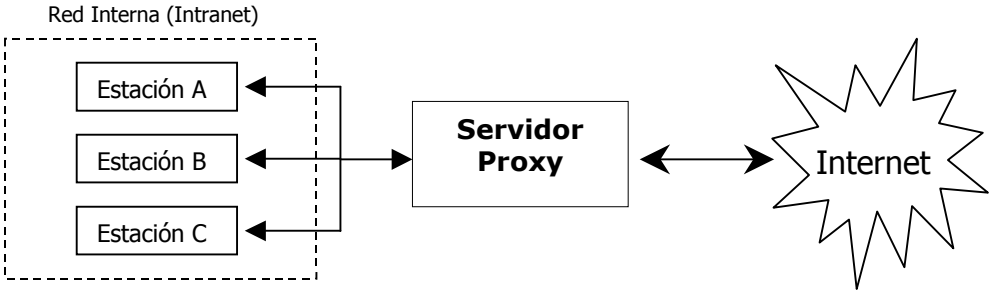


Figura 1.1: Estructura Tradicional

Dado que el sistema a desarrollar necesita tener permanente acceso al flujo de pedidos que se realizan hacia el exterior y, debido a que el único repositorio que contiene esta información es el registro (log) del servidor proxy, el sistema deberá ubicarse lógicamente entre las terminales de usuario y la conexión a Internet, uniéndose a la arquitectura proxy convencional en la forma indicada en la Figura 1.2.

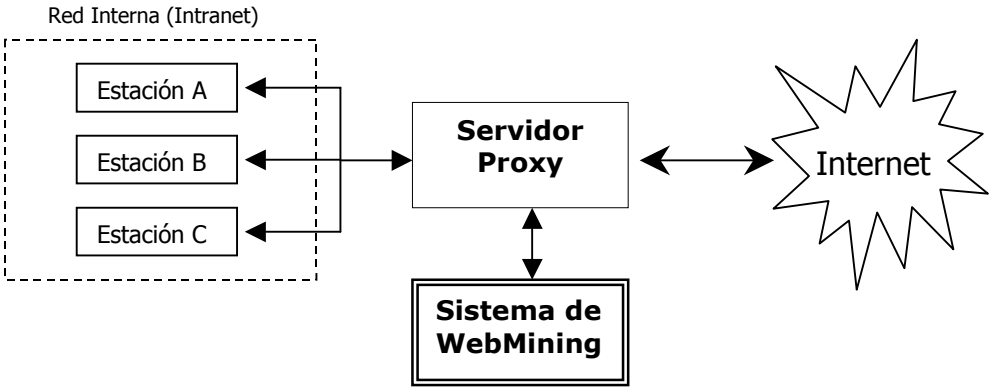


Figura 1.2: Ubicación del Sistema de WebMining

Otra alternativa sería construir una aplicación cliente que capture los patrones de navegación directamente desde las terminales de los usuarios (como realizan algunos programas denominados SpyWare[1]). Esta alternativa fue desechada porque requiere de la instalación de dicha aplicación en cada computadora conectada a la Intranet.

En la arquitectura mostrada en la Figura 1.2, se observa que el sistema de WebMining solamente se relaciona con el «mundo exterior» (las terminales e Internet) a través del Servidor Proxy. Esto es así por dos motivos:

1. Porque toma los datos básicos para trabajar a partir de él, sin recurrir directamente a los clientes. De esta forma la implementación es *independiente de la cantidad de terminales* que se estén utilizando y

1. **SpyWare**: Programas que se instalan sin el consentimiento del usuario y que envían información sensible a través de Internet sin que éste sea informado.

permite una escalabilidad natural de la arquitectura. Por el contrario, trabajar con cada cliente en particular requeriría desarrollar una aplicación que se instale en cada terminal y que acceda remotamente a otra que actúe como servidora, lo que aumentaría el tráfico en la red.

2. Los accesos a Internet son canalizados y controlados en forma centralizada por la configuración del Servidor Proxy, de forma que solamente sea necesario mantener actualizado un componente del sistema respecto de las rutas, gateways, servidores DNS y direcciones IP. De no ser así, sería necesario mantener simultáneamente dos configuraciones idénticas en diferentes componentes, lo cual puede ser fuente de errores por desactualización o diferencias.

Desde un punto de vista más específico, el sistema completo a desarrollar se grafica en la Figura 1.3, dónde pueden verse todos sus componentes, de los cuales este trabajo se ocupa de los remarcados en negrita.

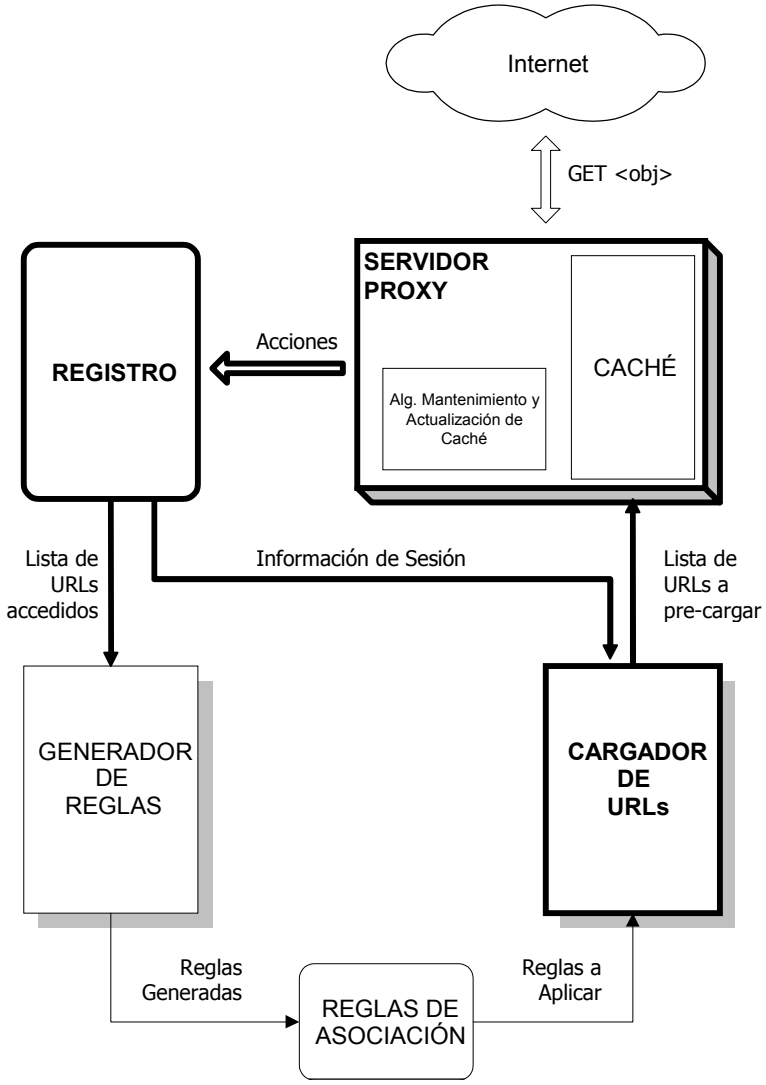


Figura 1.3: Estructura del Sistema de Web Mining

Según el esquema anterior, se describen ahora los elementos correspondientes y su interrelación dentro del sistema:

El *Servidor Proxy* es, como se dijo, uno de los componentes fundamentales. Su función incluye el acceso a Internet, la multiplexación del ancho de banda, el mantenimiento de una caché de objetos, la registración de pedidos y resultados obtenidos, etc. El Registro, o LOG, es el almacenamiento donde el Servidor Proxy registra todas sus actividades, tanto los pedidos de los clientes, como el acceso a la caché y las transferencias desde Internet. El Generador de Reglas es el módulo que aplica las técnicas de Data Mining necesarias a la información contenida en el Registro de Actividades y genera un archivo de Reglas de Asociación.

El *Cargador de URLs* toma las reglas necesarias de este archivo intermedio y lleva a cabo las operaciones necesarias para almacenar en la caché del proxy los objetos necesarios.

1.3 Organización del Trabajo

El presente trabajo se organiza de la siguiente manera:

En el Capítulo 2 se analiza el concepto de Data Mining orientado siempre a los conocimientos necesarios para cumplir con el objetivo del proyecto y dar fundamento a la solución que se implementa. El Capítulo 3 reúne información descriptiva y técnica respecto de los Servidores Proxy, su estructura, funcionamiento y registros de información. Este capítulo contiene información esencial a partir de la cual se desarrolla la solución que se implementará. En el Capítulo 4 se detallan el concepto, el funcionamiento y la implementación del sistema que cumpla con los objetivos de este trabajo. El Capítulo 5 contiene pruebas de evaluación sobre el funcionamiento del sistema y estadísticas para determinar correctos valores de configuración. Finalmente, en el Capítulo 6 se resumen las contribuciones de esta tesina y se discuten posibles trabajos futuros.

A modo de anexo se incluye una breve sección que define una posible implementación en java de una solución que implemente un servicio de caché propio y se incluye un listado de referencias y fuentes de información.

2. Data mining

Este capítulo brinda conceptos introductorios sobre Data Mining, sus usos y posibilidades.

2.1 Introducción a Data Mining

Informalmente se define Data Mining (literalmente «Minería de Datos») como la extracción de conocimiento a partir de grandes volúmenes de datos. Formalizando un poco más, podemos definirlo como la búsqueda de relaciones y patrones globales que existen en grandes bases de datos pero que permanecen ocultos ante los métodos convencionales de análisis. También se utilizan otros términos con igual o similar significado, tales como Knowledge Mining from Databases (de «Minería de Conocimiento de Bases de Datos»), Knowledge Extraction (de «Extracción de Conocimiento»), Data Archeology (de «Arqueología de Datos») y Data Dredging (de «Excavación de Datos»).

Algunos autores también se refieren a esta actividad bajo el nombre de KDD: Knowledge Discovery in Databases (de «Descubrimiento de Conocimiento en Bases de Datos»), no obstante, otros disienten y afirman que Data Mining es solamente un paso en el proceso total de KDD.

A continuación se detallan las etapas del proceso de KDD:

1. Limpieza de los Datos: Este paso intenta remover tanto ruido como inconsistencias que pueda encontrarse en los datos sobre los que se trabajará. Como se verá más adelante, no siempre es deseable que este paso se lleve a cabo.
2. Integración de los Datos: Se combinan las diferentes fuentes de datos en un único repositorio. Este repositorio global se denomina Data Warehouse[2].
3. Selección de Datos: A partir de los datos integrados en el repositorio, se toman para trabajar solamente aquellos que puedan resultar útiles para el análisis que se va a realizar.
4. Transformación de Datos: Es la agrupación de los datos útiles de forma apropiada para utilizar los algoritmos de Data Mining.
5. Data Mining: Es la aplicación de métodos inteligentes para obtener patrones útiles de datos.
6. Evaluación de Patrones: De todos los patrones obtenidos, hay que diferenciar aquellos que son verdaderamente útiles, es decir, los que representan algun tipo de conocimiento o medición interesante.
7. Presentación del Conocimiento: Es la aplicación de técnicas especiales utilizadas en la representación a los usuarios de conocimiento extraido.

Para evitar confusiones en este trabajo se considerará que Data Mining es un sinónimo de KDD.

Las herramientas de Data Mining predicen tendencias y comportamientos, posibilitando tomar decisiones proactivas y conducidas por un conocimiento acabado de la información («knowledge-driven decisions»). Las herramientas de Data Mining pueden responder a preguntas de negocios que tradicionalmente consumen demasiado tiempo de procesamiento, que los usuarios de esta información no están dispuestos a aceptar. Estas herramientas exploran las bases de datos en busca de patrones ocultos.

2. **Data Warehouse**: Conjunto de datos orientados a un sujeto, integrado, variable en el tiempo y no volátil, utilizado como soporte para el proceso de decisión gerencial. (Según W. H. Imnon [INM96])

Con el pasar de los años, junto con el avance de la tecnología, el costo de almacenar y mantener los datos ha disminuído drásticamente. Debido a esto, las empresas y organismos llevan registros de todas sus operaciones de manera realmente precisa y completa. La acumulación constante de estos datos genera gigantescos repositorios que se administran con herramientas que no han sido diseñadas para manejar estos cambios.

El auge que ha alcanzado actualmente el Data Mining es debido a que en el presente nos encontramos ante enormes cantidades de datos y con la urgente necesidad de transformarlos en información útil y conocimiento. Se dice que sin Data Mining somos «ricos en datos» pero «pobres en información». Autores como Han y Kamber [HAN01] consideran que Data Mining es la evolución natural de la tecnología de información.

Muchas compañías (como las de telecomunicaciones, ventas online y supermercados) recogen y refinan cantidades masivas de datos. Las técnicas de Data Mining pueden ser implantadas rápidamente en plataformas existentes de software y hardware para aumentar el valor y la utilidad de las fuentes de información existentes.

El nombre de Data Mining deriva de la similitud entre buscar en grandes bases de datos información valiosa sobre negocios, y minar una montaña para encontrar metales preciosos. Ambos procesos requieren examinar una inmensa cantidad de material, o investigar inteligentemente hasta encontrar exactamente donde residen los valores. No obstante, la terminología utilizada no es exacta, ya que en realidad debería llamarse «Information Mining», ya que lo que se busca extraer es información y no datos. Pero ese nombre no ha tenido tanta aceptación como los mencionados al principio de esta sección.

2.2 Patrones de Información

Como se dijo anteriormente, el objetivo de Data Mining es obtener patrones de datos. Pero, ¿qué tipos de patrones pueden ser encontrados utilizando las técnicas de esta tecnología?

Para comenzar a responder esa cuestión, primero hay que hacer una división importante. Las tareas de Data Mining pueden ser descriptivas o predictivas. Las descriptivas caracterizan las propiedades generales de los datos en una base de datos. Por el contrario, las predictivas realizan inferencias en los datos para poder realizar predicciones. En todos los casos en los que se desee aplicar técnicas de Data Mining, es muy importante tener en claro de qué tipo es el más conveniente para utilizar. Esto depende de los resultados que se quieran obtener. En el ejemplo puntual de este trabajo, desarrollar un módulo de precarga de objetos es necesariamente predictivo.

A continuación se listan las funcionalidades y tipos de patrones que pueden ser descubiertos utilizando Data Mining:

a. Descripción de Clases: La idea de trabajar con este tipo de patrones es poder asociar datos en clases o conceptos. Hay tres formas de encarar este enfoque. La primera se denomina Caracterización de Datos (Data Caracterization) y se basa en realizar una sumarización de las características generales de una clase en particular de datos. Los resultados de este tipo de análisis suele presentarse en la forma de reglas de caracterización.
   La segunda es la Discriminación de Datos (Data Discrimination), que es una comparación entre las características generales de los objetos de una clase respecto a las de otro conjunto contrastante. En este caso se habla de relgas de discriminación.
   Finalmente, también se puede aplicar una combinación de ambas, es decir, utilizar conjuntamente la caracterización con la discriminación.
b. Analisis de Asociación: Es el descubrimiento de reglas de asociación que muestran condiciones del tipo atributo-valor que ocurren con frecuencia dentro de un conjunto de datos. Por ser la clase de patrones elegidas para el proyecto, este tema se retomará más detalladamente en la Sección 2.3 de este trabajo.
c. Clasificación y Predicción: Es el proceso por el cual se busca un conjunto de modelos o funciones que describan y distingan clases de datos o conceptos, con el objeto de utilizar dichos modelos para predecir de que clase son ciertos objetos. El modelo derivado se basa en el análisis de un conjunto de datos de entrenamiento (es decir, datos de los cuales sí se conoce su clase).
d. Análisis de Clusters: En el caso anterior se hacía referencia a analizar clases conocidas, pero el clustering analiza objetos sin consultar clases conocidas. Por este motivo, esta técnica se clasifica como aprendisaje no supervisado. En general, las clases no se presentan en los datos de entrenamiento simplemente porque no se conocen. El analisis de clusters se utiliza justamente para identificar las clases. El proceso trabaja agrupando objetos según el principio de «maximizar la similitud dentro de una clase y minimizar la similitud entre clases».
e. Análisis de Infrecuentes: Una base de datos puede contener objetos que no obedecen al comportamiento general o al modelo de los datos. Esos objetos son los infrecuentes (Outliers). La mayoría de los

métodos de Data Mining descartan estos objetos como si se trataran de ruido o excepciones. Pero hay que tener en cuenta que, en algunos casos, este tipo de información puede ser más útil que la los sucesos regulares. Este tipo de análisis suele llamarse Outliers Mining.

f. Análisis Evolutivo: Describe y modela la monotonía o la tendencia que poseen determinados objetos que tienen un comportamiento variable en el tiempo. Un ejemplo clásico de aplicación de estos patrones es el de desarrollar un sistema que identifique las regularidades en un conjunto de acciones de la bolsa, para poder determinar así los ciclos de alzas y bajas.

No obstante, debe tenerse en cuenta que los sistemas de Data Mining son propensos a generar muchos más patrones o reglas de los que se esperan al inicio del proceso. Esto no necesariamente quiere decir que hay tanto conocimiento oculto en el conjunto original de datos, ya que hay que tener en cuenta que no todos los patrones generados son útiles. Es necesario entonces poder discriminar qué patrones son útiles y cuáles no lo son.

Un patrón es **interesante** si cumple con las siguientes condiciones:

* Es fácilmente comprensible para las personas.
* Es válido, con cierto grado de certeza, para otro conjunto de datos, ya sea nuevo o de prueba.
* Tiene una utilidad potencial.
* Expresa un conocimiento novedoso y no trivial.

Un patrón también será útil si sirve para validar una hipótesis que el usuario pretende confirmar.

No es difícil notar que las condiciones observadas anteriormente son, en su mayoría, subjetivas ya que dependen, en gran medida, de las creencias que el usuario tiene de los datos. Para establecer un criterio más uniforme se crearon varias medidas objetivas para determinar si un patrón es, o no, interesante. Principalmente se basan en la estructura de los patrones descubiertos y en las estadísticas que los apoyan.

En el caso de las ***reglas de asociación***, las medidas objetivas que se utilizan son el *soporte* y la *confianza* (que se verán en la Sección 2.3).

Una pregunta surge de inmediato: ¿Pueden las técnicas de Data Mining encontrar todos los patrones interesantes de un conjunto de datos?. Esta cuestión hace referencia a la completitud de un determinado algoritmo. A menudo es poco realista e incluso sería ineficiente pretender que un sistema de este tipo genere absolutamente todos los patrones. No obstante se suelen utilizar restricciones y medidas de interés definidas por el usuario, que permiten focalizar la búsqueda sobre la información realmente interesante y obtener así todas las reglas de interés.

Derivado de eso, también tiene sentido preguntarse: ¿Puede un sistema de Data Mining generar solamente patrones interesantes?. Para ello es necesario optimizar al máximo el algoritmo a utilizar. Sería deseable que los sistemas de Data Mining generen únicamente los patrones interesantes, ya que esto aumenta la eficiencia tanto del usuario como del sistema, porque ninguno de los dos tiene la necesidad de buscar entre todos los patrones generados para encontrar los realmente interesantes. De todas formas, el problema de la optimización a este nivel en la actualidad sigue siendo un reto.

2.3 Reglas de Asociación

La búsqueda de reglas de asociación se utiliza para encontrar asociaciones interesantes o relaciones de correlación entre un conjunto grande de datos. A continuación se ven conceptos básicos de este enfoque.

2.3.1 Introducción

La forma clásica de ver la extracción de reglas de asociación es a través del análisis del caso del carrito de compras, es decir, intentar obtener información sobre los hábitos de compras de los clientes a partir de la información de las transacciones diarias que se realizan.

Analicemos el clásico ejemplo de un supermercado. A través del registro de transacciones, donde se indican qué items se vendieron, se podría determinar qué productos son comprados simultáneamente por una cantidad importante de clientes. De esta forma podría descubrirse, por ejemplo, que los días Sábado es muy posible que la gente que compra cerveza también compre pañales. Este tipo de información resulta sumamente útil y se extrajo de los datos que la empresa ya poseía pero de la cual no podía extraer «conocimiento» de este tipo utilizando herramientas comunes.

Si pensamos en el universo como el conjunto de todos los artículos disponibles en una tienda, entonces cada uno posee una variable lógica que representa su presencia o ausencia dentro de un grupo de elementos. Cada carrito de compras puede, entonces, ser representado por un vector lógico de valores asignados a esas variables. Es posible analizar estos vectores de forma conjunta para determinar los patrones de compras y reflejar aquellos items que se encuentran frecuentemente asociados o comprados juntos, sin olvidar que lo que realmente interesa es la información «no trivial» que pueda extraerse. Estos patrones pueden representarse en la forma de reglas de asociación.

Por ejemplo, la información que indica que los clientes que compran cerveza también suelen adquirir pañales al mismo tiempo se representa por la siguiente regla de asociación:

$$\text{cerveza} \Rightarrow \text{pañales}$$

[soporte=4%, confianza=70%]

El soporte y la confianza son dos medidas del interés de la regla. Estos parámetros indican qué tan útil puede resultar la regla extraída. Un **soporte** del 4% indica que el 4% de todas las transacciones analizadas mostraron que la cerveza y los pañales fueron comprados juntos. Una **confianza** del 70% indica que el 70% de los clientes que compraron cerveza también compraron pañales. Generalmente, una regla de asociación se considera útil si satisface un mínimo de soporte y un mínimo de confianza preestablecidos. Ambos umbrales (threshold) son fijados por el usuario o por un experto en el tema.

Definición 1 - [Soporte y Confianza] *Una regla A ⇒ B en el conjunto de transacciones D tiene soporte s, cuando s es el porcentaje de las transacciones de D que contienen A U B (tanto a A como a B).* Esto puede expresarse como la probabilidad P(A U B):

$$\text{Soporte} (A \Rightarrow B) = P (A \cup B)$$

Por otro lado, la regla A ⇒ B en el conjunto de transacciones D tiene una confianza c, cuando c es el porcentaje de transacciones de D que, conteniendo a A, también contienen a B. Esto se expresa como la probabilidad condicional P(A|B):

$$\text{Confianza} (A \Rightarrow B) = P (A \mid B)$$

Definición 2 - [Conjunto de Items Frecuentes] *Dado un soporte S, un conjunto de items frecuentes es el conjunto de items que aparece al menos S veces dentro del conjunto de transacciones.*

Un conjunto de items frecuente es un conjunto de items que están en la misma transacción y que se repiten al menos el mínimo de soporte en todas las transacciones consideradas.

El proceso para extraer las reglas de asociación de una base de datos consta de dos partes:

a. Encontrar todos los conjuntos de items frecuentes.
b. Generar reglas de asociación fuertes entre los items frecuentes.

Las reglas de asociación permiten dada una confianza y un soporte predefinidos, determinar si existe una relación entre dos conjuntos de items. Por ejemplo, si tenemos una lista de transacciones de un supermercado, donde cada transacción es una compra realizada por un cliente cualquiera, mediante las reglas de asociación, se puede descubrir que cuando un cliente lleva el producto X, también lleva el producto Y, con un soporte y una confianza determinados.

Una vez obtenidos los conjuntos de items frecuentes, para obtener las reglas de asociación se puede expresar la probabilidad condicional en términos de la cuenta soporte de un conjunto de items:

$$\text{confianza} (A \Rightarrow B) = P (B \mid A) = \text{cuenta\_soporte} (A \cup B) / \text{cuenta\_soporte} (A)$$

Donde cuenta_soporte(A U B) es el numero de transacciones que contienen los conjuntos de items A y B, o sea A U B. Y cuenta_soporte(A) es el numero de transacciones que contienen al conjunto de items A.

A partir de esta ecuación, las reglas de asociación pueden ser generadas según el Algoritmo 1.

Algoritmo 1 - [Generación de Reglas de Asociación]

INPUT: Conjunto de items frecuentes

OUTPUT: Conjunto de reglas de asociación

MÉTODO:

1. Para cada conjunto de items frecuentes L, generar todos los subconjuntos NO vacíos de L.
2. Para cada subconjunto S NO vacío de L, generar la regla «s —> (L-s)» si (cuenta_soporte(L) / cuenta_soporte(s)) >= confianza_mínima, si confianza_mínima es el umbral mínimo de la confianza.

Park y otros [PAR95] presentan un algoritmo de hashing[3] para generar itemsets grandes, lo cual consideran la parte principal de todo el proceso de mining. Por otro lado, Hidber [HID99] introduce un nuevo algoritmo para calcular itemsets voluminosos de manera online, utilizando dos pasadas por la secuencia de transacciones.

3 **Hashing**: Es la transformación de un a cadena de caracteres en un valor (o clave) más corto y de longitud fija que lo representa.

2.3.2 Tipos de Reglas de Asociación

El análisis del carrito de compras es un ejemplo de solamente una de las formas que pueden tener las reglas de asociación. En realidad, hay varias clases posibles de reglas. Estas se clasifican siguiendo el criterio explicado a continuación:

* **Según el tipo de valores que se manejan en la regla:**

Si una regla está basada en la presencia o ausencia de items, se dice que es una regla de asociación **Booleana**. Ejemplo:

$$\text{cerveza} \Rightarrow \text{pañales}$$

Si una regla describe una asociación entre atributos o items cuantitativos, entonces se dice que es una regla de asociación **Cuantitativa**. Para este tipo de reglas, los valores se dan particionados en intervalos (indicados por dos puntos '..'). El siguiente es un ejemplo de este tipo de reglas, dónde X es una variable que representa a un cliente:

$$\text{edad(X,»20..25")} \wedge \text{ingresos(X,»20000..35000")} \Rightarrow \text{compra(X,minicomponente)}$$

* **Según las dimensiones de los datos que intervienen en la regla:**

Si los items o atributos de una regla solamente referencian una dimensión (donde una dimensión se representa mediente un predicado), entonces es una regla de **Dimensión Simple**. Ejemplo:

$$\text{compra(X,cerveza)} \Rightarrow \text{compra(X,pañales)}$$

En cambio, cuando una regla referencia dos o más dimensiones, se dice que la regla es **Multi-Dimensional**. En el siguiente ejemplo se utilizan dos dimensiones, *compra y edad*:

$$\text{compra(X,cerveza)} \wedge \text{edad (X,»20..45")} \Rightarrow \text{compra(X,pañales)}$$

* **Según los niveles de abstracción empleados en el conjunto de reglas:**

Algunos métodos de extracción de reglas permiten encontrar conocimiento en diferentes niveles de abstracción. Por ejemplo:

$\text{edad(X,»19..25")} \Rightarrow \text{consulta(X,»www.onlineub.com»)}$
$\text{edad(X,»19..25")} \Rightarrow \text{consulta(X,»www.onlineub.com/default.htm»)}$

En el primer ejemplo, la regla hace referencia al sitio consultado, mientras que la segunda hace referencia concreta a que página del sitio se accede. Ambas reglas se diferencian por el nivel de abstracción. De esta forma pueden diferenciarse entre reglas de asociación Multinivel y de Nivel Único.

* **Según extensiones a la extracción de asociaciones:**

La extracción de asociaciones puede ser extendida al análisis de correlaciones, dónde la ausencia o presencia de items correlativos puede ser identificada. También puede extenderse para extraer *maxpatterns*[4] y elementos frecuentemente cercanos. Estas extensiones pueden ser utilizadas para disminuir la cantidad de itemsets frecuentes generados en el mining.

2.3.3 Algoritmo Apriori

El algoritmo implementado para el proyecto esta basado en el algoritmo APRIORI [AGR93]. Este es un algoritmo que permite detectar los conjuntos de items más frecuentes en distintas transacciones, a través de la generación de candidatos y de reglas de asociación booleanas.

El nombre de este algoritmo hace referencia a que, en cada paso, se utiliza el conocimiento de la propiedad llamada Apriori de los itemsets frecuentes. Esta propiedad indica que para que un itemset sea frecuente todos sus subconjuntos no vacíos también deberán ser frecuentes.

El algoritmo tiene dos pasos: el paso JOIN (junta) y el paso PRUNE (optimización). El paso JOIN consiste en calcular los conjuntos de K items frecuentes, y el paso PRUNE se encarga de depurar aquellos conjuntos de K items que incluyen algún K-I conjunto de items no frecuente dentro, reduciendo la cantidad de K items frecuentes, optimizando así el proceso.

4 **Maxpattern** (patrón máximo): es un patrón frecuente tal que cualquier superpatrón propio no es frecuente.

Es importante aclarar que el algoritmo Apriori va calculando los conjuntos de K items frecuentes, a partir del conjunto K-1 de items frecuentes, por lo tanto, si asegurándose que K-1 tiene solo conjuntos de items frecuentes (gracias al paso PRUNE), se logra reducir el procesamiento en el paso JOIN.

Algoritmo 2 - [Algoritmo Apriori]

INPUT: Conjunto de transacciones (D), Soporte mínimo (min_sup)
OUTPUT: Itemsets frecuentes en D (L1)
ESTRUCTURAS DE DATOS: Conjunto de n itemsets frecuentes (Ln), Candidatos a itemsets frecuentes (Cn)

```
L1 = {Lista de itemsets en D}
PARA (k=2 ; Lk-1<>0 ; k++)
    Ck = APRIORI_GEN(Lk-1,min_sup)
    PARA (todas las transacciones t ( D)
            Ct = SUBCONJ(Ck ,t)     // Toma todos los candidatos que contienen a t
            PARA (todos los candidatos c ( Ct)
                    c.count++
            FIN PARA
    FIN PARA
    Lk = { c ( Ck / c.count ( min_sup }
FIN PARA
DEVOLVER L=Uk Lk

PROCEDIMIENTO APRIORI_GEN (Lk-1,min_sup)
    PARA (todos los itemsets l1 ( Lk-1)
            PARA (todos los itemsets l2 ( Lk-1)
                    SI (l1[1]=l2[1])^ (l1[2]=l2[2])^...^ (l1[k-1]=l2[k-1]) ENTONCES
                            c = l1 JOIN l2                  //Paso JOIN
                            SI (TIENE_SUBCONJ_FRECUENTES (c,Lk-1))
                                    BORRAR c                //Paso PRUNE
                            SI NO
                                    AGREGAR c A Ck
                            FIN SI
                    FIN SI
            FIN PARA
    FIN PARA
    DEVOLVER Ck
FIN PROCEDIMIENTO

FUNCION TIENE_SUBCONJ_FRECUENTES (c,Lk-1)
    PARA (cada subconjunto (k-1) s ( C)
            SI ( S ( Lk) ENTONCES
                    DEVOLVER VERDADERO
            SI NO
                    DEVOLVER FALSO
            FIN SI
    FIN PARA
FIN FUNCION
```

A continuación se brinda un ejemplo del funcionamiento del algoritmo Apriori:

Ejemplo 1:

Sea un conjunto de Transacciones T={T1,T2,...,T9} donde cada Ti está constituído por un conjunto de ítems Ii. El soporte exigido es del 22% (2/9).

T1 : I1, I2, I5
T2 : I2, I4
T3 : I2, I3

T4 : I1, I2, I4
T5 : I1, I3
T6 : I2, I3
T7 : I1, I3
T8 : I1, I2, I3, I5
T9 : I1, I2, I3

En la primera pasada del algoritmo cada ítem es miembro del conjunto de candidatos y se busca el número de ocurrencias de cada uno en todas las transacciones (Tabla 2.1).

Itemset	Soporte
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Tabla 2.1: Itemsets candidatos de 1 elemento

Itemset
{I1}
{I2}
{I3}
{I4}
{I5}

Tabla 2.2: Itemsets frecuentes de 1 elemento

La cantidad mínima de apariciones indicada por el soporte es 2, es decir que en el primer paso no se borrarán itemsets ya que todos los candidatos son frecuentes (Tabla 2.2). A continuación se procede a generar la lista de cantidatos para dos ítems y se calcula el soporte para cada uno de ellos (Tabla 2.3). Los marcados en negrita son aquellos que no cumplen con el soporte necesario y se eliminarán (Tabla 2.4).

Itemset	Soporte
{I1,I2}	4
{I1,I3}	4
{I1,I4}	**1**
{I1,I5}	2
{I2,I3}	4
{I2,I4}	2
{I3,I4}	**0**
{I3,I5}	**1**
{I4,I5}	**0**

Tabla 2.3: Itemsets candidatos de 2 elementos

Itemset
{I1,I2}
{I1,I3}
{I1,I5}
{I2,I3}
{I2,I4}
{I2,I5}

Tabla 2.4: Itemsets frecuentes de 2 elementos

A partir de los itemsets frecuentes se generan los candidatos de tres ítems (Tabla 2.5). Pero al realizarse la junta (paso JOIN del algoritmo) debe tenerse en cuenta la propiedad Apiori (todos los subconjuntos de un conjunto frecuente también deben ser frecuentes) porque gracias a ella se pueden eliminar combinaciones que no tienen posibilidades de ser frecuentes (paso PRUNE del algoritmo), ahorrando tiempo de proceso.

Itemset	Soporte
{I1,I2,I3}	2
{I1,I2,I5}	2
{I1,I3,I5}	No puede ser frecuente por {I3,I5}
{I2,I3,I4}	No puede ser frecuente por {I3,I4}
{I2,I3,I5}	No puede ser frecuente por {I3,I5}
{I2,I4,I5}	No puede ser frecuente por {I4, I5}

Tabla 2.5: Itemsets candidatos de 3 elementos

Posteriormente se comprueba el soporte de los itemsets candidatos (sólo de aquellos que no fueron eliminados por el prune) y se determinan cuales son los itemsets frecuentes (Tabla 2.6).

Itemset
{I1,I2,I3}
{I1,I2,I5}

Tabla 2.6: Itemsets frecuentes de 3 elementos

El proceso continúa en forma análoga hasta que el conjunto de items candidatos tenga cero elementos y ya no puedan determinarse nuevos itemsets frecuentes.

Para obtener una descripción más completa y formal del algoritmo Apriori ver Agrawal [AGR93] y Han [HAN01].

Se han propuesto las siguientes optimizacione a efectos de mejorar la eficiencia del algoritmo. Algunas de ellas son:

* **Hashing:** Utilizar una técnica de hashing para reducir el tamaño de los itemsets candidatos. Ver: Park, Chen y Yu [PAR95].
* **Reducción de Transacciones:** Se basa en reducir el número de transacciones tenidas en cuenta en las futuras iteraciones del algoritmo. Ver: Agrawal y Srikant [AGR94], Han y Fu [HAN95] y Park, Chen y Yu [PCY95]
* **Particiones:** Funciona particionando los datos en dos grupos. En un primer paso se determinan los itemsets frecuentes locales a cada partición. Posteriormente se determina cuales de esos son itemsets frecuentes globales. Ver: Savarese, Omiecinski y Navathe [SAV95]
* **Muestreo:** Trabaja con el concepto de determinar los itemsets frecuentes de un subconjunto (muestra) del total de las transacciones. Se pierde precisión a cambio de eficiencia. Ver: Toivonen [TOI96]
* **Conteo Dinámico de Itemsets:** Es una modificación del Apriori que consume menos pasadas por las transacciones, ya que los nuevos itemsets candidatos pueden ser agregados en cualquier momento durante el proceso. Ver: Brin, Motwani, Ullman y Tsur [BRI97]

2.4 Web Mining

La aplicación de técnicas de Data Mining sobre los datos contenidos en la World Wide Web (WWW) recibe el nombre de Web Mining. Si bien esta definición es bastante sencilla, abarca tres enfoques bien diferenciados:

El primer enfoque apunta a realizar mining en los servidores. Esto es, sobre los datos que se encuentran almacenados en los Servidores de Web. De esta forma, analizando los logs del servidor de una empresa, puede determinarse cómo los usuarios ven el sitio y utilizar esta información para mejorar su estructura o evaluar su índice de conversión[5]. En lo sucesivo de este trabajo llamaremos a este enfoque «Server Web Mining» o SWM.

El segundo enfoque es trabajar con técnicas de Data Mining en los documentos obtenidos de la red. El objetivo principal de este tipo de webmining es el de mejorar los motores de búsqueda. En lo sucesivo de este trabajo llamaremos a este enfoque «Document Web Mining» o DWM.

Finalmente, el tercer enfoque se encuentra ubicado entre los anteriores. Se trabaja con la información que los usuarios de una red local bajan de los servidores remotos, para obtener así patrones de navegación (similares al SWM) pero localmente, según los objetos que son descargados desde Internet (como en el DWM). La implementación de Data Mining de este trabajo está abarcada por este tercer enfoque, al que en lo sucesivo llamaremos «Usage Web Mining» o UWM.

2.4.1 Server Web Mining

En la actualidad, los servidores de páginas Web generan numerosos datos provenientes del registro (log) de los pedidos que reciben de parte de las aplicaciones clientes de los usuarios. Estos requerimientos son registrados de manera constante e inmediata durante todo el tiempo en el que el servicio está activo.

Todo ese volumen de datos contiene información que no puede detectarse utilizando técnicas convencionales y que permite conocer de que forma los usuarios navegan dentro del sitio, que página los atrae más, cuales casi no son visitadas, etc. Esta información se transforma en vital si pensamos en una empresa que proporciona el servicio de venta online a sus clientes.

No obstante, las herramientas más comunes de estadísticas de utilización de sitios Web no utilizan técnicas de Data Mining. Sin embargo, a causa de la creciente demanda de eficiencia por parte de los sitios de comercio electrónico deberá recurrirse a una herramienta más avanzada que utilice estas herramientas a la hora de descubrir el conocimiento que se almacena día a día y que, en la actualidad, casi nadie puede utilizar correctamente.

Sobre este tema hay numerosos trabajos publicados. Por ejemplo, Zaiane y otros [ZXH98] hacen referencia a la utilización de técnicas de Data Mining sobre los logs de un servidor de web para mejorar la performance del sitio y aumentar la calidad de sus contenidos. Chen y otros [CPY96] se refieren a la extracción de patrones transversales en un ambiente de información distribuida, tal como la WWW. Borges y Lavene [BOR00] y también Büchnet, Baumgarten y otros [BUC99] discuten el descubrimiento de patrones de navegacion de los usuarios dentro de un sitio. Por otro lado, Lan y Bressan [LAN99] aplican técnicas de Data Mining con el objetivos de hacer más competitivos los servidores de Web.

Mendelzon y Rafiei [MEN00] introducen un proceso se búsqueda para determinar los temas sobre los que un sitio web posee una «alta reputación» a partir de los páginas que tienen vínculos directos con ella, mientras que Kleinberg [KLE97] formula un algoritmo para determinar relaciones entre la estructura de vínculos de un sitio y encontrar así páginas que actúen como «autoridades» sobre determinados temas, basado en la topología determinada por los hipervínculos.

2.4.2 Document Web Mining

Su finalidad es la de organizar perfiles de usuarios para mejorar la eficacia de las búsquedas que ellos realizan a través de Internet. Se aplica también en la búsqueda de documentos muy similares, con el objeto de filtrar búsquedas más generales realizadas de forma convencional.

Este tipo de aplicación de técnicas de Data Mining se ha concretado a través de la implementación de agentes inteligentes que intentan extraer las características semánticas de las palabras o la estructura de los documentos HTML. Una vez hecho eso, se utilizan dichas características para organizar los documentos en diferentes clasificaciones. La principal metodología utilizada es la de clustering (ver Sección 2.2). Dado que estas aplicaciones tienen poca relación con el objetivo del presente trabajo, no nos extenderemos en su análisis.

---

5 **Indice de Conversión**: Relación entre la cantidad de visitas que tiene un sitio y la cantidad de transacciones que se concretan en él. Es un indicador utilizado para evaluar sitios de e-commerce ya que mide el grado en que los visitantes se transforman en clientes.

Para profundizar estos conocimientos puede recurrirse a Broder y otros [BRO97], un paper que hace referencia a un método (basado en clustering) para determinar la similaridad sintáctica entre archivos y que fue aplicado a páginas de la Web. Una posible aplicación de esto es identificar violaciones a los derechos de propiedad intelectual de un autor. Moore y otros [MHB98] se refieren a técnicas para evaluar el contenido de páginas web, llevando a cabo experimentos con datos reales para comparar heurísticas y métodos de clustering con las técnicas convencionales basadas en distancia.

2.4.3 Usage Web Mining

Esta rama del Web Mining trabaja con la información que se registra, en una red local, acerca de los accesos que los usuarios internos han hecho a servidores remotos de páginas web, para obtener así patrones de navegación. La diferencia principal con el SWM, es que éste tiene por objetivo analizar un sitio para optimizarlo, en cambio el UWM buscan optimizar la conexión en general desde el punto de vista de los usuarios, independientemente del sitio al que ingresen.

Sobre este tópico no tenemos conocimiento acerca de publicaciones al momento de escribirse esta tesina.

Respecto a la fuente de datos utilizada para extraer las reglas de asociación, hay dos opciones posibles:

a) Utilizar la información que se registra en cada cliente (navegador) a través de la memoria caché local para determinar las páginas accedidas últimamente.
b) Utilizar el registro del Servidor Proxy, que contiene todos los objetos accedidos por los clientes de la Intranet.

La opción (a) tiene como ventaja que, en entorno Windows, se dispone de funciones propias de la API[6] del sistema operativo para controlar los objetos almacenados. La desventaja es que debe instalarse una aplicación cliente en cada terminar para administrar dicha información.

Por otro lado, la opción (b) es más centralizada (ya que solamente debe instalarse una aplicación) y contiene información histórica de un período más largo de tiempo. Los inconvenientes son: que se necesita tener instalado un Servidor Proxy para que los clientes accedan a Internet y que no hay funciones de la API que den soporte a la administración de los objetos que se encuentren en la caché del Servidor.

Para el caso de este proyecto se ha decidido por trabajar bajo la alternativa (b) debido a los siguientes motivos:

* Inicialmente el sistema completo puede implementarse y probarse utilizando una sola computadora. No obstante, pueden agregarse terminales clientes al conjunto sin necesidad de un esfuerzo adicional
* La información histórica contenida en el registro del Servidor Proxy es de un período más amplio y más variada (ya que se registra a partir de los accesos de todos los usuarios que tenga el sistema). Esto permite que se generen reglas específicas reconocidas a través de un patrón de accesos de un usuario que posteriormente puedan aplicarse a otro usuario que accede a través de otra terminal. En el caso de la primera alternativa esto sería imposible.

No obstante al no disponerse de una API incorporada, como ya se mencionó, deberá desarrollarse una interfase que permita regular el comportamiento del Servidor Proxy y coordinar su funcionamiento con el algoritmo de Data Mining. Esto es, precisamente, el núcleo fundamental d este trabajo.

2.4.4 Beneficios e Inconvenientes del Web Mining

La utilización de técnicas de Data Mining que trabajen sobre el log de un servidor (tanto en el SWM como en el UWM) promete beneficios, aunque existen algunos inconvenientes para llevarla a cabo.

A continuación enumeramos los beneficios que pueden obtenerse al utilizar esta tecnología:

* Permite mejorar le performance en el servidor.
* Permite una reestructuración del sitio para mejorar su navegabilidad.
* Permite descubrir potenciales clientes de comercio electrónico.
* Identifica las horas pico de acceso y los lugares preferidos por los usuarios, para permitir colocar publicidades estratégicas.

Pero también deben tenerse en cuenta los siguientes inconvenientes, que pueden quitar exactitud a los resultados obtenidos:

* No hay forma exacta de determinar el inicio y fin de la sesión de un usuario.
* No se tiene información sobre el acceso a páginas almacenadas en la caché local de los clientes de navegación.
* La información registrada puede ser ambigua si hay cambio de nombres de servidores o reubicación de páginas.

6 **API**: Application Programmable Interface, Interfase Programable de Usuario. Conjunto de funciones que brindan un servicio a una capa de mayor nivel de abstracción.

2.5 Pre-Procesamiento de los Datos

Las bases de datos reales, a diferencia de las teóricas o ideales, suelen contener problemas tales como ruido y datos faltantes o inconsistentes. Esto se debe, principalmente, al enorme tamaño que poseen y a que son utilizados para registrar eventos constantemente. El pre-procesamiento de los datos busca mejorar la calidad de los datos y, de esta forma, la de los resultados del mining.

2.5.1 Introducción

Existen numerosas técnicas para el pre-procesamiento de los datos. La Limpieza de Datos (Data Cleaning) se utiliza para eliminar el ruido y corregir las inconsistencias. La Integración de Datos (Data Integration) unifica los datos de múltiples fuentes en un repositorio de datos coherentes, como un Data Warehouse o un Cubo de Datos[7]. Las Transformaciones de Datos (Data Transformations), tales como la normalización, se utilizan para mejorar la eficiencia y precisión de los algoritmos de mining que involucran cálculos de distancias. La Reducción de Datos (Data Reduction) permite disminuir la cantidad de datos ya sea aplicando funciones de agregación, eliminando datos redundantes, aplicando clustering u otros métodos.

Cualquiera de las técnicas nombradas anteriormente permiten reducir el tiempo de procesamiento necesario para un algoritmo de Data Mining y, al mismo tiempo, mejora la calidad general de los patrones extraídos.

Para el caso de nuestro proyecto, la técnica de pre-procesamiento que se utilizará sobre los datos de entrada es la de Limpieza de Datos. La siguiente sección expande los conceptos ya explicados.

2.5.2 Limpieza de Datos

Las rutinas de limpieza de datos funcionan llenando los valores faltantes, alisando el ruido, identificando y removiendo infrecuentes y resolviendo inconsistencias. Si los datos no se limpian pueden producir confusiones en el proceso de mining, dando como resultado una salida poco confiable.

A continuación se verán los métodos básicos que se utilizan para la Limpieza de Datos:

a. Para valores faltantes:
   a. Ignorar la tupla: Se utiliza principalmente cuando el valor que falta es el que caracteriza a la tupla.
   b. Completar el valor manualmente: Se utiliza solamente en algunos casos puntuales porque requiere demasiado tiempo.
   c. Usar una constante global para completar el valor: Es un método bastante simple que consiste en reemplazar los valores faltantes por una constante predefinida que esté fuera del rango de los datos (N/A, 0, -1, etc). El inconveniente es que las rutinas de mining pueden encontrar relación entre las tuplas que contienen esta constante.
   d. Usar el promedio del atributo para completar el valor.
   e. Usar el promedio del atributo de los elementos de la misma clase para completar el valor: Es similar al anterior pero utilizando algún criterio para definir diferentes clases de tuplas.
   f. Usar el valor más probable para completar el valor: Se utilizan métodos varios (regresión, formalismos Bayesianos, árbol de inducción, etc) para determinar el valor más probable.
b. Ruido: El ruido es un error aleatorio que se produce en una variable medida.
   a. Vecinos: Se utiliza el valor de las tuplas vecinas para realizar un promedio de los valores de forma local. Es similar a las técnicas utilizadas para retocar fotografías.
   b. Clustering: Se utiliza esta técnica para separar los datos en conjuntos. Aquellos datos que no caen en ningún conjunto son llamados infrecuentes y pueden (según el caso) tenerse como ruido.
   c. Inspección automática y manual: Se buscan los infrecuentes con algoritmos combinados con revisión humana (valores no esperados).
   d. Regresión: Consiste en alisar los datos haciendo que se ajusten con una función.
   c. Inconsistencias: Las inconsistencias pueden detectarse y corregirse manualmente utilizando referencias externas, pero también se pueden emplear herramientas de ingeniería del conocimiento para detectar la violación de alguna restricción sabida de los datos.

2.6 Resumen

En este capítulo hemos introducido el concepto y las técnicas de Data Mining, exponiendo de forma simple aquellos conocimientos necesarios para abordar la consecución del objetivo de este trabajo. Los pilares fundamentales son, definitivamente, el algoritmo Apriori y la clasificación de los distintos tipos de Web Mining.

7 **Cubo de Datos**: Modelización de datos en múltiples dimensiones. Ver [GRA96].

3. Servidores Proxy

Este capítulo trata sobre características, propiedades y cuestiones técnicas de los Servidores Proxy en general, y del Microsoft Proxy Server[8] en particular. Éste producto ha sido elegido para ser utilizado en el proyecto para mantener compatibilidad con el resto de la plataforma Microsoft instalada en el laboratorio de desarrollo pre-existente.

3.1 Conceptos de Servidores Proxy

Las redes de área local (LAN) funcionan, en general, según el conjunto de protocolos TCP/IP[9]. Las características de este set de protocolos son las siguientes: arquitectura cliente-servidor, organización por capas (layers) y conexión orientada a puertos (cada terminal de la red posee 64000 puertos virtuales que pueden conectarse a un puerto de otra terminal).

Los bloques de datos que son transmitidos a través de las LANS son fraccionados en pequeños segmentos llamados paquetes. Esto permite una mejor distribución del tiempo y los recursos necesarios para procesar un pedido. Cada paquete se compone de dos partes: La primera contiene información sobre sus características, su dirección de origen y su dirección de destino. A esta sección se le conoce como header (cabecera). Luego de este encabezado se haya una porción llamada cuerpo que contiene parte de los datos originales que se desean transmitir.

Cuando varias LANs se interconectan y se desea que dos computadoras conectadas a diferentes redes se comuniquen, es necesario disponer de una entidad que pueda administrar los caminos (o rutas) que los paquetes deben tomar para poder pasar de una red A a otra C a través de una red intermedia B. Estas entidades son los routes (o encaminadores).

Un *servidor proxy* puede verse como un traductor que actúa como intermediario entre los clientes de una red local e Internet [HEI99]. Superficialmente funciona como un router, pero con la diferencia de que en éste los paquetes se transfieren casi intactos entre una red y la otra, mientras que el servidor proxy modifica las cabeceras de los mismos.

Más formalmente, la RFC[10] 1945 [RFC1945] define el término proxy como *«un programa intermediario que actúa tanto como servidor y como cliente con el propósito de realizar pedidos en representación de otro cliente. Las peticiones se procesan internamente o se pasan, con una posible modificación, hacia otros servidores. Un servidor proxy debe interpretar y, de ser necesario, reescribir un mensaje de pedido antes de reenviarlo [...]».*

Desde el punto de vista del hardware es una computadora que está conectada a dos redes diferentes. Esto, además, le permite controlar el flujo de información que pasa de una red a otra. Su función principal es la de proveer acceso a Internet a varias estaciones de trabajo en forma confiable y a bajo costo, utilizando una sola conexión con el exterior. Esto resulta extremadamente útil debido a que solo se necesita una dirección IP válida de Internet para conectar cualquier número de terminales a la red (con IPs de red interna). Además, puede integrarse con funciones de firewall para proveer de seguridad a la red.

Una característica importante es la capacidad de almacenamiento de memoria caché, que aumenta la performance de las conexiones, disminuyendo los costos de conexión. Según la RFC 1945 [RFC1945], el término caché se refiere a *«un almacenamiento local de un programa, de mensajes de respuesta y el subsistema que controla el almacenaje, recuperación y borrado de los mismos [...]».* El tema de la caché se analiza en la Sección 3.2.2

De forma simple un servidor proxy funciona de la siguiente manera: primero escucha[11] a las computadoras de la red interna. Cuando una aplicación cliente hace un pedido, responde modificando las direcciones fuente y destino y pasándolo a Internet. Cuando una computadora del exterior responde, el servidor pasa esa respuesta nuevamente a la aplicación cliente de la computadora que hizo el pedido.

Las ventajas de utilizar servidores proxy son varias:

* El servidor proxy es la única computadora visible desde Internet. La red privada permanece invisible y pueden utilizarse características de firewall para impedir el ingreso de usuarios no autorizados.

8 Microsoft Proxy Server es una Marca Registrada de Microsoft Corp.

9 **TCP/IP**: Transfer Control Protocol / Internet Protocol, por Protocolo de Control de Transferencia y Protocolo de Internet. TCP está definido en [RFC793] e IP en [RFC791].

10 **RFC**: Request For Comments (Pedido de Comentarios). Es un documento formal emitido por la IETF (Internet Engineering Task Force, Fuerza de Tareas Ingenieriles de Internet). Algunos son meramente informativos mientras que otros tienen carácter de estándar. La RFC 1945 define el estándar HTTP/1.0

11 El termino escuchar (listen) hace referencia a abrir un puerto TCP y mantenerlo en ese estado esperando que un cliente se conecte solicitando un servicio.

* Si se desea permitir que sólo algunos individuos accedan a Internet, puede utilizarse el servidor proxy para autentificarlos. Aquellos que puedan conectarse satisfactoriamente al servidor serán los únicos capaces de acceder a Internet. Así mismo, es posible restringir los servicios disponibles para los usuarios a nivel individual.
* El servidor proxy puede mantener una caché de datos recientemente recuperados. Si un cliente necesita acceder a una información que ya se encuentra en la caché, el servidor puede suministrarla sin necesidad de acceder a la red.
* El servidor proxy puede filtrar las peticiones salientes. Esto es útil para bloquear el acceso a sitios determinados.
* El aislamiento de la red privada y de Internet es tan completo que ni siquiera es necesario ejecutar TCP/IP. Puede utilizarse IPX/SPX.

Un Web Proxy trabaja exclusivamente con el protocolo HTTP (puerto 80). Pero un servidor de proxy que soporte el estándar Socks[12] (o Winsock, en el caso de la plataforma Win32), llamado Socks Proxy o Winsock Proxy, soporta todos los protocolos de Internet, y también cualquier tipo de aplicación cliente/servidor que se desarrolle con posterioridad.

Para realizar este trabajo unicamente es necesario que el proxy soporte HTTP, pues el objetivo es trabajar con objetos de la WWW y no con todos los servicios disponibles. No tendría sentido plantear la utilización de la memoria caché de forma predictiva como medio de aceleración de servicios tales como, por ejemplo, mail, noticias, transferencia de archivos o streaming de video.

3.2 Servidores Proxy a Bajo Nivel

Esta sección se ocupa de los conceptos de funcionamiento más internos de los Servidores Proxy.

3.2.1 Registro de Actividades

El registro de actividades, comúnmente denominado «log», es una estructura de datos que contiene información sobre cada pedido que un cliente le ha realizado al servidor, además de datos propios del objeto y del servidor remoto que lo contiene.

Este archivo puede verse como un almacén de las transacciones que se realizan a medida que las terminales acceden a Internet.

MS Proxy Server 2.0 admite dos formas de mantener estos registros. La primera, y la más común, es escribir las transacciones en un archivo de texto plano. Cuando se quiere acceder a los datos almacenados de esta manera es necesario un proceso de parsing de las cadenas de caracteres para reconocer cada campo de información.

Pero también existe otra característica, menos utilizada, que permite que el servidor grabe directamente las transacciones en una base de datos (como MS SQL Server y MS Access) a la que se accede vía ODBC. Esto permite un mayor acceso a la información histórica, primero porque no es necesario el proceso de parsing, y segundo porque pueden realizarse consultas que en otro caso serían complejas y poco intuitivas.

Por las necesidades de acceder a estos registros se decidió aplicar la segunda técnica de logging, utilizando MS SQL Server 7.0.

La estructura de los datos que se almacenan en el registro es la indicada en la Tabla 3.1:

Nombre del Campo	Tipo de Dato (SQL)	Tipo de Dato (Access)	Longitud
BytesRecvd	Int	Long Int	-
BytesSent	Int	Long Int	-
CacheInfo	Int	Long Int	-
ClientAgent	Varchar	Text	100
ClientAuthenticate	Char	Text	5
ClientIP	Varchar	Text	50
ClientUserName	Varchar	Text	50
DestHost	Varchar	Text	255
DestHostIP	Varchar	Text	50

12 El estándar Socks 5 se detalla en [RFC1928]

DestHostPort	Int	Long Int	-
LogTime	Datetime	DateTime	-
MimeType	Varchar	Text	25
ObjectSource	Varchar	Text	25
Operation	Varchar	Text	255
ProcessingTime	Int	Long Int	-
Protocol	Varchar	Text	25
ReferredServer	Varchar	Text	100
ResultCode	Int	Long Int	-
ServerName	Varchar	Text	50
Service	Varchar	Text	25
Transport	Varchar	Text	25
Uri	Varchar	Text	255

Tabla 3.1: Estructura del registro de actividades del MS Proxy Server 2.0

De todos estos campos, los que nos interesan son:
- **ClientIP:** almacena la dirección IP de red del cliente que realizó el pedido.
- **ClientUserName:** contiene el nombre de usuario NT del cliente.
- **DestHost:** guarda el nombre del sitio remoto al que se accedió.
- **Protocol:** indica si el objeto fue accedido por HTTP o FTP.
- **Uri:** contiene la dirección (URL) completa del objeto accedido.

3.2.2 Caché de Objetos

La caché de objetos, en el caso de los servidores proxy, es un espacio de almacenamiento local donde se almacenan los objetos recibidos desde el exterior para que estén disponibles de una forma más rápida cuando otro usuario lo solicita. Utilizar cachés mejora la performance, disminuye la latencia y ahora ancho de banda. Algunos autores, como Luotonen [LUO97], consideran que esta es la característica más importante de los servidores proxy, junto con las capacidades de seguridad y monitoreo.

No obstante, implementar cachés tiene desventajas que no deben olvidarse. Primero, que se corre el riesgo de recibir un objeto que no es el más actualizado que existe en el servidor origen. De todas formas hay mecanismos para minimizar estas posibilidades. Estos mecanismos se basan en las instrucciones explícitas que contiene el protocolo HTTP y las heurísticas acerca de cuándo un objeto puede ser cacheado y cada cuánto se necesita verificar si todavía está actualizado.

Segundo, dado que el espacio de almacenamiento de los servidores es limitado, no es posible cachear todos los objetos solicitados. Por ello se implementan rutinas que deciden qué prioridad debe darse a los contenidos para eliminarlos de la caché en el caso que no quede lugar disponible para nuevos objetos.

En relación cuándo un objeto puede ser cacheados, el Microsoft Proxy Server 2.0 sigue los siguientes criterios. Si un objeto no cumpliese con alguno de ellos, no será almacenado en la memoria caché luego de ser recuperado desde su sitio origen:

* El pedido debe estar hecho a través del comando GET.
* El objeto debe transferirse utilizando el protocolo HTTP.
* El header de la respuesta al pedido no puede contener alguno de los siguiente:
  - WWW-Authenticate
  - Set-Cookie
* La fecha de vencimiento del objeto (campo «Expires») debe ser posterior al a fecha actual del objeto (campo «Date»).
* El código de estado de respuesta debe ser 200 (éxito).
* El objeto no debe estar encriptado o protegido por SSL (Secure Sockets Layer).

Para profundizar sobre este tema recomendamos leer [MSC06].

3.2.2.1 Tiempo de Vida

El tiempo de vida (TTL por Time To Live) se define como el intervalo de tiempo en el que un objeto es válido. Mientras se encuentre en este estado, cualquier pedido por parte de un cliente hará que el proxy devuelva la copia que se encuentra en la caché. Una vez terminado el TTL, el proxy deberá buscar nuevamente en el sitio de origen para comprobar la existencia de una versión más actualizada del objeto y renovarlo en caso de ser necesario. Una vez que se tiene el nuevo objeto (o que se ha comprobado que no ha habido modificaciones), el TTL se vuelve a establecer.

3.2.2.2 Caché Pasiva y Activa

El MS Proxy Server soporta dos tipos de configuración para la caché: pasivo y activo. Cada una de estas modalidades serán explicadas a continuación.

Cuando la caché funciona de forma **pasiva**, los objetos que se encuentran almacenados en ella solamente se actualizan cuando un usuario solicita dicho objeto. La consecuencia directa de esta forma de funcionar es que si no hay clientes conectados, el servidor no realiza ninguna actividad. Por otro lado, si después de un período de tiempo medianamente largo un cliente solicita un objeto de la caché que ha extendido su TTL, el tiempo de espera será mayor porque será necesario realizar la actualización desde el sitio de origen. A esta forma de trabajo se la denomina «on-demand» (sobre demanda).

La otra forma de funcionamiento de la caché se denomina **activa.** Este tipo de administración de la caché busca distribuir mejor el trabajo que realiza el servidor proxy. Cuando se vence el TTL de un objeto, el servidor buscará una actualización aunque ningún cliente lo haya solicitado. De esta forma estará disponible para un próximo pedido sin necesidad de demorar un tiempo extra. Desde el punto de vista del usuario, este sistema tiene mayor velocidad. Esto se debe al balanceo de la carga de trabajo en los tiempos de menor actividad. A esta forma de trabajo se la denomina «on-command» (por instrucción).

3.2.2.3 Estructura de Almacenamiento

El MS Proxy Server 2.0 organiza el almacenamiento de los objetos de la caché de la siguiente forma:

Los objetos que se mantienen en el almacenamiento intermedio se guardan en la carpeta \urlcache del servidor. Esta carpeta, a su vez, se organiza en 5 sectores dónde se guardan los archivos que contienen los objetos. Estos sectores reciben el nombre *\dir1, \dir2,* etc.

Un aspecto interesante a tener en cuenta es que el tamaño de estas 5 sub-carpetas está limitado y es igual para todas, de forma que los objetos que exceden ese tamaño no son almacenados en la caché.

3.3 Protocolo HTTP

Los servidores proxy, en su función de i[illegible]arios, están en la intersección de varios protocolos. Como se explicó más arriba, deben ser capa[illegible]terpretar los pedidos que transmiten y por esta razón deben conocer los protocolos en forma completa. En la Web el protocolo principal es el HTTP (HyperText Transfer Protocol, Protocolo de Transferencia de Hipertextos). Dado que este trabajo se centra en el manejo de los objetos disponibles en sitios Web, explicaremos los conceptos fundamentales de este protocolo a efectos de comprender cómo funcionan los pedidos y respuestas entre servidores.

HTTP es un protocolo de pedido/respuesta. Los clientes envían los pedidos al servidor y éste les contesta con una respuesta. No hay inicios de conexión de varios pasos (como ocurre con el protocolo FTP) además de los necesarios para establecer la conexión TCP. La Figura 3.1 muestra este simple mecanismo:

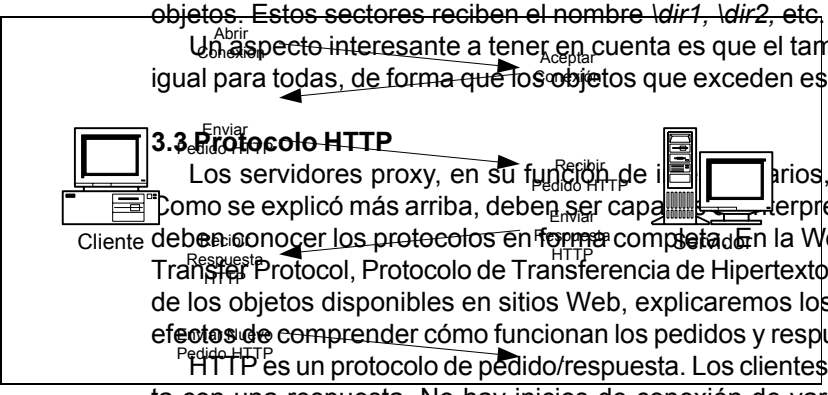


Figura 3.1: Pedido simple con HTTP

En casos más reales, los proxys se ubican de forma intermedia entre el cliente y el servidor. De esta forma el cliente le envía el pedido al servidor proxy, y este se lo envía al servidor o a otro servidor proxy. Esto se denomina **cadena de pedido.** La respuesta viaja a través del mismo camino, es decir la **cadena de respuesta**. Estos proxys intermediarios pueden almacenar recursos temporalmente y responder a pedidos utilizando esta caché, sin enviar el pedido al servidor original.

En la Figura 3.2 se presenta un caso en el que la respuesta no puede ser obtenida de la caché de ningún servidor proxy, y el pedido debe llegar hasta el sitio original (Servidor Web) para poder completarse. Este es el caso, por ejemplo, de la solicitud de una página dinámica escrita en ASP.

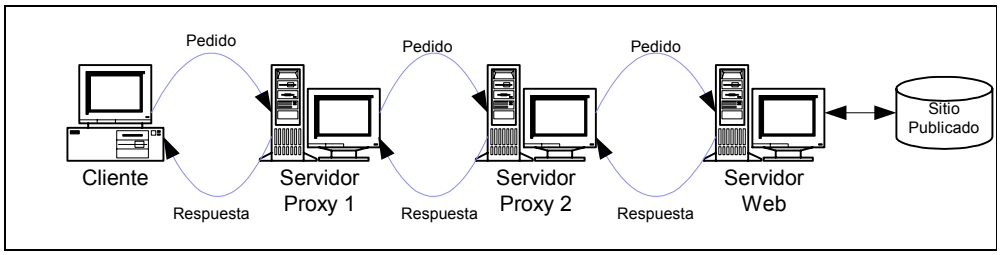


Figura 3.2: Pedido en cadena con HTTP (cadena completa)

La Figura 3.3 muestra un caso más deseable (por la mejora de performance) en el que el pedido del cliente puede satisfacerse utilizando la información almacenada en la caché de uno de los servidores proxy intermediarios.

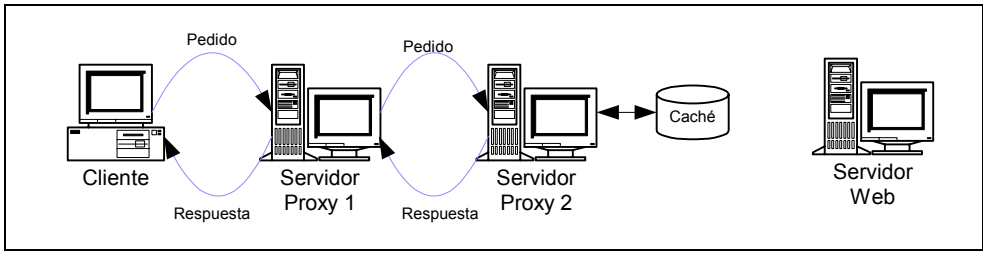


Figura 3.3: Pedido en cadena con HTTP (aprovechamiento de la caché)

3.3.1 Versiones

Desde su aparición hasta nuestros días, el protocolo HTTP ha pasado por un proceso de utilización continua; por este motivo existen tres versiones del estándar (y otra más que está en desarrollo):

HTTP/0.9

Fue la primera versión. Era bastante simple ya que solamente soportaba el método GET. Permitía recuperar documentos pero sin ningún tipo de autentificación ni capacidades de control de acceso, excepto aquellas basadas en el IP del cliente. La respuesta solamente contenía el documento solicitado, sin ninguna información adicional. Inmediatamente luego de enviada la respuesta, la comunicación se da por terminada.

HTTP/1.0

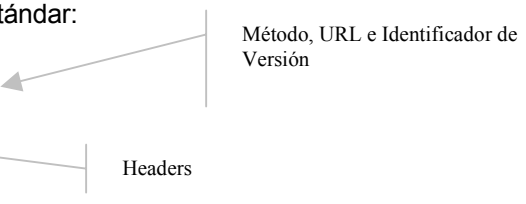
Esta versión se encuentra documentada completamente en la RFC 1945 [RFC1945] titulada «Hypertext Transfer Protocol - HTTP/1.0».

Su principal mejora es que introduce los encabezados (headers): un formato extendido para los pedidos y las respuestas, lo que permite mayor flujo de información entre los sistemas conectados. Los campos de encabezado son pares con un nombre simple (una sola palabra), y se emplean de forma similar al MIME[13]. De manera análoga, las respuestas también contienen una sección con encabezados, junto a una línea de estado que confirma si la operación se realizó con éxito o no.

Otro cambio fundamental es que introduce el concepto de Servidor Proxy (visto anteriormente), y por consecuencia, los datos a tener en cuenta en la forma de realizar los pedidos cuando se hacen a un servidor intermedio en lugar del servidor que contiene la página. Principalmente en estos casos se debe tener cuidado de indicar el sitio de origen de forma completa y con el prefijo que indique el protocolo a utilizar (http://)

13 El estándar de MIME (Multipurpose Internet Mail Extensions, Extensiones Multipropósito de Mail por Internet) está indicado en la RFC 1341 [RFC1341].

Ejemplo 1: En este ejemplo se muestra un pedido HTTP realizado por una computadora que accede directamente a Internet e intenta conectarse con el servidor www.misitio.com.ar para recuperar la página «carpeta/pagina.html» a través de un navegador estándar:

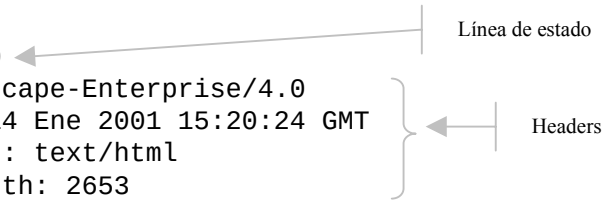



```
GET /carpeta/pagina.html HTTP/1.0
Host: www.misitio.com.ar
User-Agent: Mozilla/5.0
Accept: text/html
```

Ejemplo 2: Este pedido intenta recuperar la misma página que en el caso anterior, pero con la diferencia que la conexión se raliza a través de un servidor Proxy:

```
GET http://www.misitio.com.ar/carpeta/pagina.html HTTP/1.0
Host: www.misitio.com.ar
User-Agent: Mozilla/5.0
Accept: text/html
```

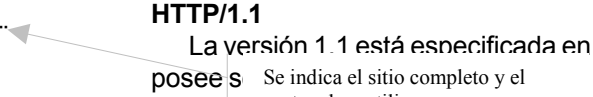
Ejemplo 3: Una respuesta a cualquiera de los pedidos anteriores podría tener la siguiente forma:




```
HTTP/1.0 200
Server: Netscape-Enterprise/4.0
Date: Lun, 14 Ene 2001 15:20:24 GMT
Content-Type: text/html
Content-length: 2653
... contenido del documento ...
```

HTTP/1.1

La versión 1.1 está especificada en el estándar de la RFC 2068 [RFC2068]. Las principales mejoras que posee s



Cone te mantener la conexión abierta para realizar varios pedidos sin necesidad de do el overhead que introduce el TCP en este sentido[14].

Pipeline de pedidos: Permite que se realice un nuevo pedido antes de que se termine de recibir por completo la respuesta anterior. Esto reduce la latencia en cargar un sitio.

Control de Caché: Introduce directivas para controlar mecanismos propios de la caché tanto de los clientes como de los proxys.

Todo pedido HTTP está formado, como se vio en el ejemplo anterior de la siguiente forma:

* Método: identifica el tipo de operación a realizar.
* URL destino.
* Identificador de versión del protocolo HTTP.
* Conjunto de headers: contienen información adicional respecto del método a utilizar.

3.3.2 Headers

Como se dijo, la utilidad de los headers es agregar información adicional a los pedidos y respuestas. El estándar de la versión 1.1 de HTTP define un total de 46 headers, divididos en cuatro categorías: generales, de pedido, de respuesta y de entidades. Los generales pueden aparecer tanto en los pedidos como en las respuestas, mientras que los de pedido y de respuesta son específicos para ese tipo de operación. Los encabezados de entidad (u objetos) describen el contenido del objeto solicitado o enviado.

La estructura de un pedido es la siguiente:

METODO URL HTTP/versión
... Headers generales ...
... Headers de pedido ...

14 Cuando se establece una conexión TCP hay un mínimo de 3 paquetes de datos que necesitan ser enviados. Estas transmisiones agregan tiempo a la comunicación.

... Headers de entidad (opcional) ...
(línea en blanco)
... Objetos enviados (si los hubiera)...

La estructura de las respuestas son similares:
HTTP/versión código-de-estado línea-de-motivo
... Headers generales ...
... Headers de respuesta ...
... Headers de entidad (opcional) ...
(línea en blanco)
... Objetos solicitados (si los hubiera)...

Además de los definidos por el estándar, HTTP permite agregar los propios en caso de ser necesario. Estoy hace que el protocolo sea fácilmente extensible para nuevas aplicaciones. Aunque siempre hay que recordar que el soporte de una versión implica incondicionalmente mantener la compatibilidad con las versiones anteriores.

A continuación se da una explicación de los headers más comunes que se utilizarán en este trabajo:

Cache-Control:

- (Permite controlar varios aspectos del caching. Las directivas definidas para este header en los pedidos son las siguientes:
  No-cache: Solicita una validación de extremo a extremo, esto significa que el servidor origen debe ser alcanzado para asegurar que se accede al objeto más actualizado existente, sin importar los niveles intermedios de proxy que existan.
  No-store: Los proxy no deberán guardar ninguna parte del mensaje de pedido ni tampoco de la respuesta respectiva en un medio no volátil (HD u otro).
  Max-age: Determina el tiempo máximo de edad que es aceptable para el cliente, sin importar los valores que tenga el servidor origen.
  Max-stale: Indica que el cliente está esperando aceptar una respuesta vencida, es decir, que ha estado más tiempo en la caché que lo indicado en su tiempo de vida.
  Min-Fresh=seconds: Requiere que los objetos devueltos tengan un tiempo mínimo de validez (en segundos) antes de vencerse.
  Only-if-cached: Solicita un objeto solamente si existe en la caché.

- (Las directivas de este header en las respuestas son:
  Public: La respuesta puede ser mantenida en cualquier caché
  Private: La respuesta sólo puede mantenerse en la caché local del usuario pero no en servidores proxy.
  No-cache: La respuesta no debe ser mantenida en ninguna caché.
  No-store: Ni los proxy ni el cliente deberán almacenar esta respuesta en memoria no volátil (HD o similar).
  Must-revalidate: Obliga a todas las cachés (tanto del proxy como del cliente) a realizar una confirmación de punta a punta con el servidor origen cuando un objeto se exceda de su tiempo de vida.
  Proxy-revalidate: Como el anterior pero solamente se aplica para los servidores proxy.

  Accept: Indica que tipos de objetos se esperan como respuesta. Es común utilizar text/html (texto en formato html), text/plain (texto sin formato), image/gif (imagen en formato gif), image/* (cualquier imagen) y */* (cualquier objeto de cualquier tipo).

  Connection: Indica si debe cerrarse la conexión luego de completado el pedido o no.

  Host: Indica el nombre del sitio origen.

  User-Agent: Indica la denominación de la aplicación que realiza el pedido. El formato es nombre_del_software/versión.

3.3.3 Códigos de Estado de Respuesta

Cada respuesta que un servidor envía está precedida por un código que indica el estado del pedido que se procesó (o se está procesando). Conocer el significado de estos valores permite realizar algoritmos más eficientes que solamente trabajen con los registros de log que han tenido resultados esperados. Esto se verá más adelante cuando se vea qué tipo de limpieza se aplicará a los datos para el proyecto.

Los códigos de estado de HTTP están organizados en 5 categorías. El primer dígito indica a qué categoría pertenece el estado, los siguientes dos dígitos especifican la condición con más detalle.

Las 5 categorías se muestran en la Tabla 3.2:

Código	Categoría	Descripción
1xx	Información	Información provisional sobre el código de estado. El código real se enviará una vez terminado el proceso.
2xx	Éxito	El pedido fue recibido y procesado con éxito.
3xx	Redirección	Se requiere otra acción por parte del programa cliente. Generalmente se da en casos de redirección hacia otro URL.
4xx	Error del Cliente	Indica un error por parte del cliente en el pedido.
5xx	Error del Servidor	Indica un error del lado del servidor.

Tabla 3.2: Categorización de los Códigos de Respuesta

La Tabla 3.3 muestra los códigos de estado más comunes:

Código	Significado	Descripción
100	Continuar	El cliente puede continuar con su pedido.
101	Cambiando Protocolos	Se ha cambiado de versión o de protocolo de trabajo
200	OK	El pedido fue exitoso y la respuesta se envía a continuación.
204	No hay Contenido	Es una página intencionalmente en blanco.
206	Contenido Parcial	Se ha recuperado solamente una parte del objeto solicitado.
301	Movido de forma Permanente	El recurso ha sido movido definitivamente hacia otra ubicación.
302	Movido Temporalmente	Similar al 301 pero es una situación temporal.
303	Ver Otro	Redirección automática a una dirección como resultado de un POST.
304	Sin Modificar	Indica que la copia encontrada en la caché del cliente (o proxy) se encuentra actualizada.
400	Pedido Erróneo	El pedido no pudo ser comprendido por el servidor.
401	Se Requiere Autorización	El objeto solicitado no puede ser recuperado hasta que no se realice la autorización correspondiente.

403	Prohibido	El Servidor ha negado el acceso a un determinado recurso.
404	No Encontrado	El documento solicitado no existe en el servidor.
408	Tiempo de Espera Excedido	El cliente no envió una respuesta dentro del tiempo en el que el servidor estuvo esperando.
500	Error Interno del Servidor	Un error genérico que indica un fallo inesperado en el servidor.
501	No Implementado	Se ha realizado un pedido no válido y no fue posible procesarlo.
503	Servicio No Disponible	El servicio se halla temporalmente fuera de servicio.
505	Versión de HTTP no Soportada	La versión de HTTP solicitada por el cliente no está soportada por el servidor.

Tabla 3.3: Códigos de Estado

3.4 Resumen

En este Capítulo se desarrolló el concepto de los Servidores Proxy, detallando su funcionamiento y profundizando en el protocolo HTTP hasta el nivel de conocer cómo se realizan los pedidos entre un HOST y un cliente de la red. Además, se introdujo el concepto de la caché incorporada a los Servidores Proxy y se brindaron detalles sobre las características de su funcionamiento. Estos conceptos resultan indispensables para poder comprender la solución que se implementa en este trabajo, y que se explica en detalle en el siguiente Capítulo.

Para profundizar sobre la estructura y funcionamiento interno de los Servidores Proxy recomendamos [LUO97]. En cuanto a las características del Microsoft Proxy Server 2.0 puede encontrarse más información en [HEI99].

4. Análisis y desarrollo de la solución

Este capítulo describe la solución que implementa la conexión entre Servidor Proxy y Servidor de Data Mining.

4.1 Funcionamiento del Sistema

En la Figura 4.1 se muestra el estado final del sistema. Los módulos se indican con un rectángulo, mientras que los repositorios de datos se marcan con una forma cilíndrica. Las flechas indican el sentido del flujo de la información.

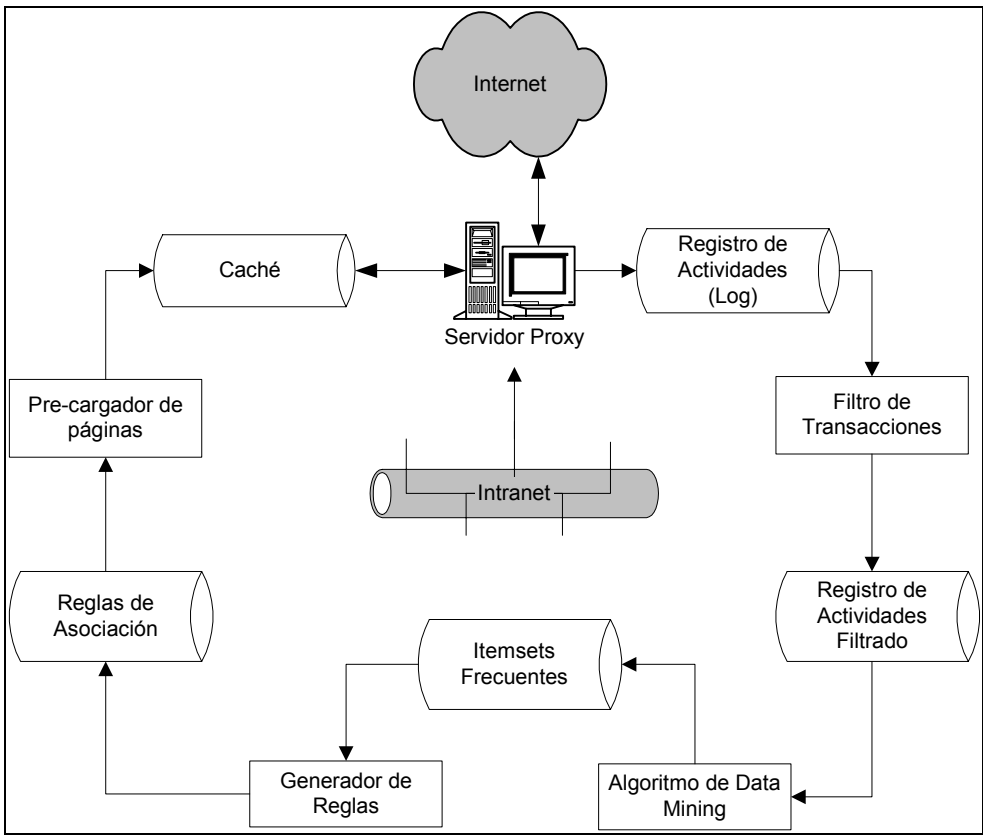


Figura 4.1: Esquema final del Sistema

Los desarrollos se efectuaron en diferentes lenguajes de programación, debido a los distintos requerimientos de los módulos. A continuación se describe cada uno de los módulos y sus características de implementación:

Filtro de Transacciones: Fue escrito en TransactSQL y trabaja sobre tablas de MS SQL Server. Su desarrollo se detalla en la Sección 4.4.

Algoritmo de Data Mining: Este módulo fue desarrollado por otros integrantes del grupo de trabajo y no forma parte de esta Tesina. Utilizando el algoritmo Apriori, este módulo toma como entrada los distintos pedidos que los clientes realizan al Servidor Proxy y genera una lista de los itemsets frecuentes. Se implementó como un Stored Proceduce en Transact SQL.

Generador de Reglas: Este módulo también fue desarrollado por otros integrantes del grupo de trabajo y no forma parte del trabajo desarrollado para esta Tesina. Toma como input la lista de itemsets generada por el Algoritmo de Data Mining y genera reglas de asociación que permiten determinar las posibles futuras páginas a ser visitadas por los usuarios. Su implementación se llevó a cabo en Visual Basic.

Pre-Cargador de Páginas: Este es el otro módulo que implementa los servicios mencionados dentro de este trabajo y se detalla en la Sección 4.5. Fue desarrollado en Java.

El funcionamiento del sistema podría describirse en los siguientes pasos:

1. Los usuarios realizan accesos a Internet a través de su red local (Intranet) que se conectan con el exterior a través de un Servidor Proxy que tiene implementada la solución desarrollada en este trabajo.

2. La información de los pedidos es almacenada por el Servidor Proxy en su registro de actividades (Log).
3. El **filtro de transacciones** realiza una copia de aquellos registros que efectivamente se considerarán útiles para el proceso de Data Mining.
4. El **Algoritmo de Data Mining** se ejecuta de forma periódica (pero no constantemente ya que consume muchos recursos del sistema) y utiliza el algoritmo Apriori para generar una lista de itemsets frecuentes a partir de los datos de transacciones que se han filtrado.
5. Una vez que se tienen todos los itemsets correctamente generados, se ejecuta el **Generador de Reglas** que arma las reglas de asociación a partir de la información suministrada por el algoritmo Apriori.
6. El **Pre-Cargador de páginas** se ejecuta también de forma periódica (e independientemente del algoritmo de data mining y el generador de reglas) y se ocupa de mantener actualizados los contenidos de la caché teniendo en cuenta las últimas reglas de asociación que se encuentran en la base de datos.
7. Cuando los usuarios repiten sus patrones de navegación, el sistema prevé los posibles próximos sitios a ser accedidos y mantiene una copia de éstos actualizada en la caché; de esta forma, la solicitud del usuario puede ser satisfecha sin necesidad de acceder a Internet y obteniendo una ventaja significativa en el tiempo de respuesta.

4.2 Servicios a Implementar

Como se definió inicialmente, el propósito de esta tesina es brindar un método de conexión bidireccional entre el Servidor Proxy y el de Data Mining, para que el módulo de Data Mining pueda acceder a los datos generados por el Servidor Proxy y permitir que las páginas con alta probabilidad de ser visitadas por los usuarios sean pre-cargadas en la memoria caché. Esto implica mantener dos vías de comunicación: una desde el Proxy hacia el módulo de Data Mining y otra de sentido inverso.

El primer servicio que debe implementarse es uno que permita al Módulo de Data Mining obtener los datos necesarios para trabajar en la generación de reglas de asociación. Este servicio implementa la **comunicación desde el Servidor Proxy**, que se detalla en la Sección 4.3. Como fue explicado anteriormente es necesario realizar un proceso de «data cleaning» previo a la utilización de estos datos. Algunos de ellos pueden ser resultado de operaciones exitosas de recuperación desde sitios remotos, otros provienen de accesos directos a la caché y algunos pueden ser el resultado de un error de acceso o del servidor origen (los tipos de errores se mencionaron en el Capítulo 3). Luego, para transformar los datos «crudos» en datos que puedan ser procesados sin que se corra el riesgo de generar resultados inconsistentes (o no veraces) es necesario contar con un servicio que implemente un cierto grado de **pre-procesamiento de los datos,** que se explica en detalle en la Sección 4.4.

Para cumplir con el objetivo de brindar un método que permita mantener objetos (páginas) a voluntad en la memoria caché de un Servidor Proxy, asegurándose en todo momento que la información se encuentre actualizada y no se utilicen copias obsoletas, se requiere un servicio que permita al Módulo de Data Mining indicar al Servidor Proxy qué objetos desea mantener en memoria caché y poder tener un grado de control sobre dicho almacenamiento. Esto supone una correcta puesta a punto del Servidor y una implementación del servicio que tenga en cuenta este aspecto, llamado **comunicación hacia el Servidor Proxy.** En la Sección 4.5 se verá como se lleva a cabo esta funcionalidad.

4.3 Comunicación desde el Servidor Proxy

Distintas alternativas se estudiaron para la implementación de este servicio.

La primera solución posible a este enlace de comunicación sería poder utilizar funciones propias del Servidor Proxy para informar, de una manera directa, las acciones ejecutadas por los clientes remotos. No obstante, los servidores proxy en general, y el MS Proxy Server en particular, no poseen dichas funciones para establecer la conexión.

Dado que se está utilizando un Servidor Proxy comercial y no desarrollado particularmente para este caso, podemos decir que se trata de un componente pasivo. Es decir, sus funcionalidades están pre fijadas y no pueden extenderse a efectos de satisfacer las necesidades puntuales del presente proyecto. El Servidor de Data Mining, por el contrario, se desarrolla a la medida de los requerimientos y por ello podemos considerarlo un objeto activo. Esta diferencia fundamental define (y al mismo tiempo restringe) el tipo de comunicación entre ambos módulos. Se puede afirmar que el Proxy Server se limita a almacenar la información de los accesos de los clientes en un registro, sin brindar otro mecanismo de comunicación que el propio acceso a los datos. Esta característica obliga a implementar el mencionado mecanismo de comunicación accediendo a los datos de forma externa a la aplicación y dándoles el pre-procesamiento adecuado para que puedan brindar un resultado más eficiente.

Una forma de permitir el mencionado acceso a los datos generados por el Servidor Proxy es utilizando un repositorio común. El servidor accede a él para dejar registro de las transacciones ocurridas y el módulo de

Data Mining toma los datos de ésta misma ubicación. Gráficamente se vería cómo se indica en la Figura 4.2:

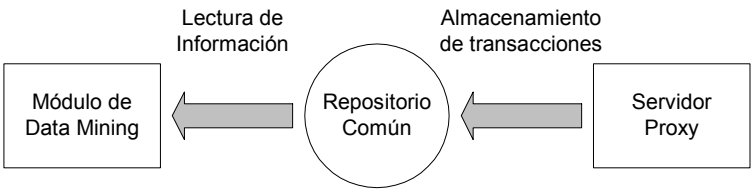


Figura 4.2: Un repositorio común para el Servidor Proxy y el módulo de Data Mining

Para brindar acceso a los datos, y que el mismo sea accesible en forma concurrente por varias aplicaciones, se utilizó una interfaz ODBC (Open DataBase Conectivity, de Conectividad Abierta de Bases de Datos). Esto permite tener una API, es decir, un conjunto de funciones comunes, para acceder a los datos utilizando SQL. De esta forma, para tener acceso a los datos desde otras aplicaciones, es suficiente con adaptar la funcionalidad del Servidor Proxy para que trabaje con una base de datos que soporte el estándar ODBC.

Para conseguir que el Servidor Proxy almacene el registro de actividad en una base de SQL Server, se necesita configurarlo de forma apropiada. Esto se consigue con los siguientes pasos:

1. Crear una base de datos en el Servidor SQL Server para contener tanto a la tabla que se utilizará para el log como a otras tablas de trabajo y stored procedures necesarios.
2. Crear un DSN para la base de datos. El DSN (Data Source Name) es un nombre simbólico utilizado por ODBC para hacer referencia al administrador y a otra información requerida para acceder a los datos, tal como la denominación del servidor dónde los datos se hallan alojados. Debe asignarse, entonces, un nombre de DSN a la base de datos que será utilizada como repositorio para que el Servidor Proxy almacene las transacciones.
   Para crear un DSN en Windows NT hay que indicar por lo menos la siguiente información:
   * Nombre del DSN
   * Nombre del Server SQL
   * Método de autenticación con el SQL Server
   * Usuario y Clave (en caso de ser necesario) para acceder.
   * Nombre de la Base de Datos a utilizar

   También pueden configurarse opciones del manejador de ODBC, que permiten mantener registro de todas las operaciones que se le soliciten sobre la base de datos indicada en el presente DSN.

La Figura 4.3 es una captura de pantalla que muestra la ventana de creación de DSN llamado PSL y una ventana tipo pop-up en la que se indican las opciones de configuración. En ella puede verse el nombre de la base a utilizarse (ProxyLog):

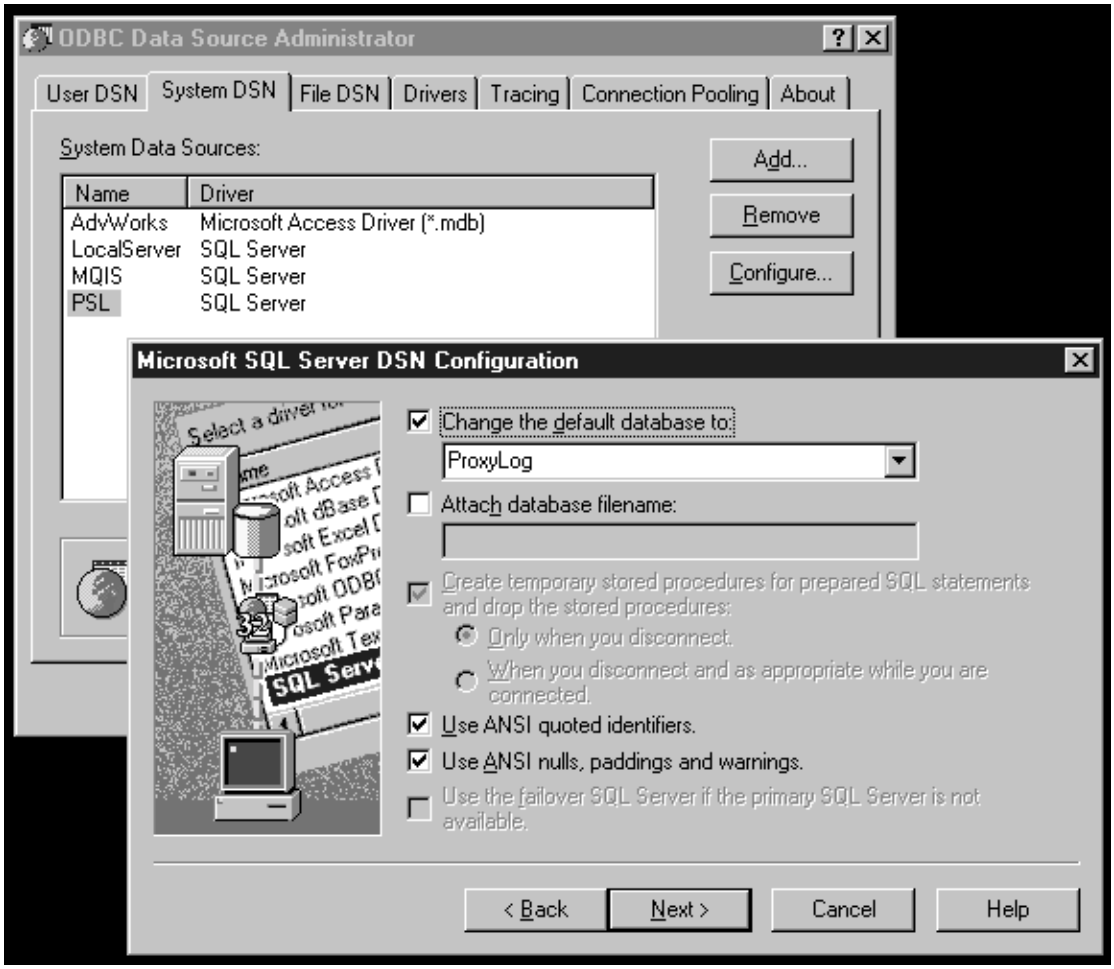
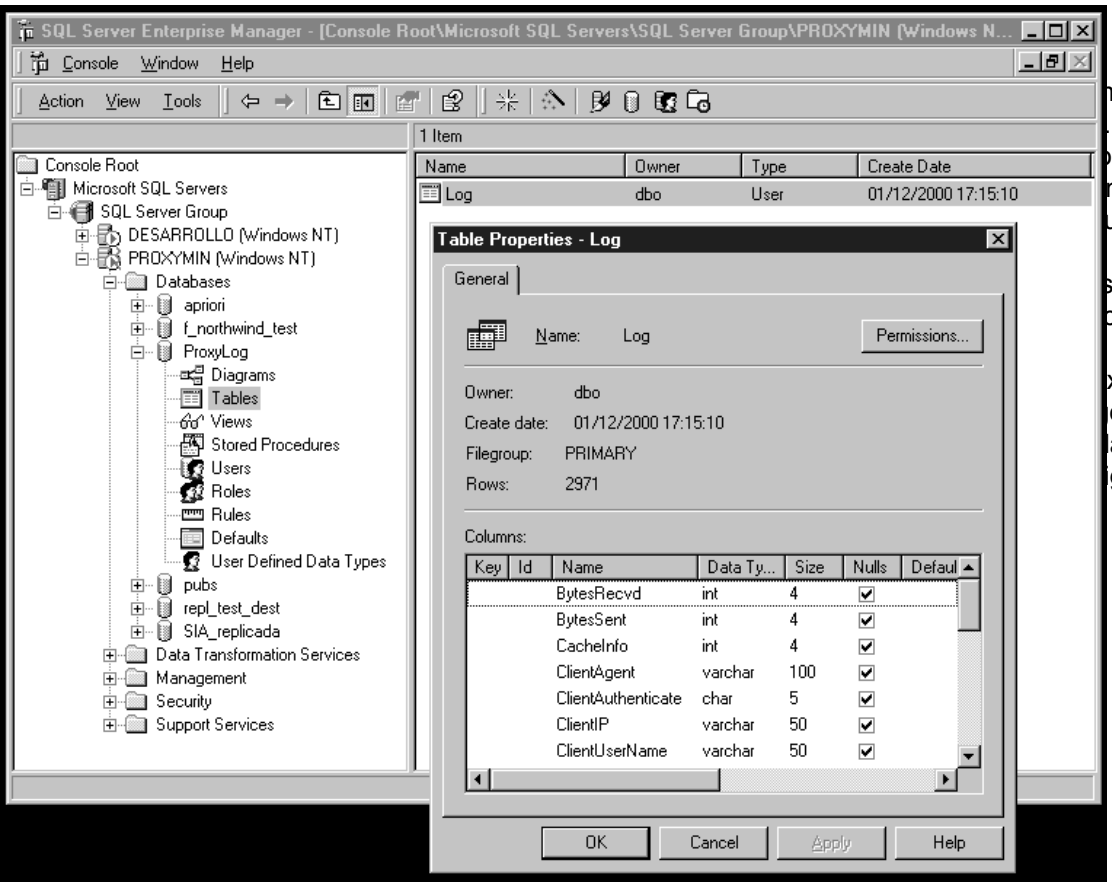


Figura 4.3: Creación de un DSN para la base de datos

3. Crear una tabla dentro de la base de datos según la estructura requerida por el log del Servidor Proxy, ya mencionada en el Capítulo 3. La Figura 4.4 muestra los campos de la tabla «Log» creada particularmente para este trabajo dentro de la base «ProxyLog». Los campos de la tabla son los mencionados en la Sección 3.2.1.



no se dijo en el Capítulo
. Siempre que se instala
rmación de transaccio-
n al usuario de cambiar
utilizar una tabla de una

structura de control del
cer una tabla indicada a

xy Server 2.0. Para ver
o puede configurarse el
la combinación usuario/
gura 4.5:

15 Este es un bug reconocido por Microsoft. Véase el documento [MSC01] para profundizar sobre el mismo.

Figura 4.5: Configuración del archivo de log del MS Proxy Server 2.0

La segunda alternativa para evitar este procedimiento sería desarrollar e implementar íntegramente un Servidor Proxy propio, de forma de dotarlo de funciones para poder controlar sus actividades y brindar soporte al trabajo cooperativo con otros módulos. Si bien la posibilidad de transitar por este camino surgió en las primeras etapas del proyecto, se optó por trabajar con un Servidor ya desarrollado y probado comercialmente debido, principalmente, al tiempo necesario para que obtener un producto estable y de performance similar al finalmente utilizado, ya que trabajar con una implementación poco optimizada de un administrador de caché sería contraproducente con el objetivo final del trabajo. No obstante en el Anexo I se explican los lineamientos generales necesarios para el desarrollo de un Servidor que implemente una caché de objetos de web, que puede ser tenido en cuenta como base de futuras ampliaciones de este trabajo.

Para este trabajo, se ha elegido la primera alternativa, debido a que el desarrollo de un Servidor Proxy comercial escapa al alcance del proyecto y porque el sistema resultante puede ser puesto en producción sobre una red que se encuentre en funcionamiento sin necesidad de realizar cambio alguno.

4.4 Pre-Procesamiento de los Datos

Como se explicó en el Capítulo 3, algunas páginas no pueden ser almacenadas en la caché debido a que pueden causar que los datos presentados al usuario sean obsoletos. Este problema se acrecienta con las páginas generadas dinámicamente, que en general dependen de los datos almacenados en una base de datos, que constantemente sufre modificaciones. Estos tipos de páginas (generalmente programadas en ASP, PHP, Perl y otros lenguajes) no son tenidas en cuenta a la hora de almacenarlas localmente debido al inconveniente planteado. Esto sugiere que en el desarrollo tampoco deberán ser consideradas para evitar perder el tiempo procesando reglas sobre páginas que de cualquier forma no podrán ser almacenadas en la caché.

A continuación se describen los diferentes aspectos a tener en cuenta en el pre-procesamiento a llevar a cabo sobre el log generado por el Proxy Server antes de que el módulo de Data Mining tenga acceso a los mismos.

Requerimientos fallidos: Sólo serán tenidas en cuenta aquellas tuplas que representen una efectiva recuperación del objeto deseado. Un objeto inexistente carece de utilidad a la hora de determinar una regla

de asignación. De la misma forma, existen varias situaciones que provocan que el resultado de un pedido realizado por un cliente no sea utilizable para el proceso.

Puede determinarse si una tupla debe o no ser «limpiada» de la base a partir del campo que indica su código de estado. Los códigos que determinan la inutilidad de una tupla son:

206: Requerimiento incompleto. Si no pudo completarse el pedido, el objeto no habrá sido completamente cargado.

400: Requerimiento erróneo realizado por el usuario. No se ha recuperado objeto alguno que pueda ser almacenado.

401: Se requiere autorización. Cómo el objeto está protegido por un nivel de seguridad (usuario-clave) no pudo ser recuperado.

403: Acceso prohibido. No se pudo acceder al objeto y, por consiguiente, no se lo puede almacenar localmente.

404: No se encuentra documento. El objeto no existe en el servidor de origen.

501: Comando incompleto. Debido a un error en el pedido del cliente, no se ha recuperado ningún objeto.

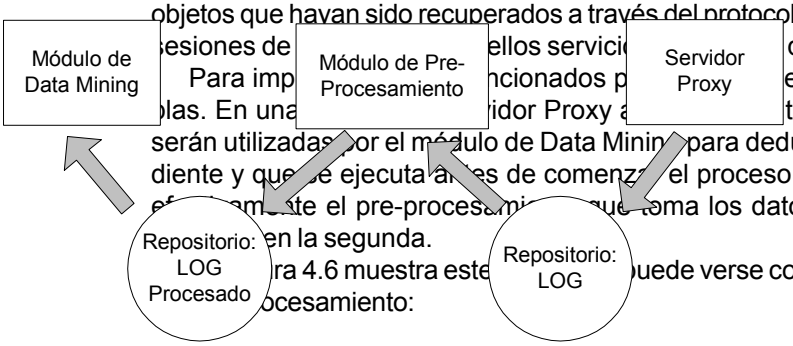
Tuplas incompletas: Aquellas tuplas dónde un dato necesario para el análisis se encuentre ausente (el campo URI por ejemplo) no serán tenidas en cuenta para el análisis. Cómo se estudió en el Capítulo 2, los atributos críticos no pueden ser intuidos por el sistema y completarlos de forma manual sería costoso.

Agentes automáticos: Los pedidos que realiza un usuario suelen enviarse a través de un navegador de Web. Aquellos que sean generados por herramientas automáticas, tales como programas que permiten bajar el contenido completo de un sitio, no registran el comportamiento de los usuarios. El filtrado de estos pedidos permite disminuir la cantidad de tuplas sobre las que se debe trabajar y evita la aparición de reglas inútiles que pueden aparecer cuando se utilizan estos agentes inteligentes.

Una posible forma de determinar el origen del pedido es utilizar el campo ClientAgent y buscar en la cadena de identificación del programa cliente, si ésta contiene la palabra clave Mozilla (común tanto para MS Internet Explorer como para Netscape Navigator). Si no se encuentra, debe desestimarse la tupla por haber sido producida por alguna herramienta «extra» de navegación.

Protocolos: Dado que el Servidor Proxy permite manejar una amplia gama de protocolos, y este proyecto está pensado para aplicarse sobre páginas web, un buen filtro es utilizar solamente las tuplas de aquellos objetos que hayan sido recuperados a través del protocolo HTTP. De esta forma no se tomarán en cuenta las sesiones de ellos servici que se pueden brindar a través de un cliente proxy.

Para imp ncionados p entos, se utilizó un sistema compuesto por dos tablas. En una idor Proxy a transacciones. En la otra se ubican las tuplas que serán utilizadas por el módulo de Data Mining para deducir las reglas de asociación. Un proceso independiente y que se ejecuta antes de comenzar el proceso de deducir las reglas es el encargado de realizar efectivamente el pre-procesamiento que toma los datos de la primera tabla y escribe los datos que se usarán en la segunda.



ra 4.6 muestra este uede verse como una extensión de la Figura 4.2 más el agregado ocesamiento:

Figura 4.6: Pre-procesamiento de los datos

Log es la tabla que utiliza el Proxy para registrar las solicitudes realizadas por los usuarios. *Log_Procesado* es la nueva tabla, con la misma estructura que la anterior, que se utiliza como entrada para el módulo de Data Mining y que solo contiene las tuplas que se consideran útiles para general las reglas. El proceso de filtrado se lleva a cabo copiando de una tabla hacia la otra solamente las tuplas que cumplan con las condiciones necesarias para ser consideradas válidas.

El filtrado propuesto en los puntos anteriores, se implementa a través de la consulta mostrada en la Figura 4.7:

```
delete from log_procesado
insert into log_procesado
        select * from log
    where ClientAgent like 'Mozilla%'
        and Protocol='http'
        and ResultCode not in (206,400,401,403,404,501)
        and ResultCode < 10000
        and ResultCode > 99
    and Uri is not null
```

Figura 4.7: Consulta para Pre-Procesamiento

En base a una prueba realizada con datos reales sobre el log del Servidor Proxy, se obtuvieron reducciones sustanciales en la cantidad de tuplas. Esta disminución del volumen de datos permite acelerar el proceso de generación de las reglas de asociación entre objetos y, además, evita el trabajo con datos que podrían disminuir la utilidad del sistema en general.

Utilizar la consulta anterior (Figura 4.7) cada vez que se recalculan las reglas de asociación requerirá, como es de suponer, un tiempo de procesamiento que producirá demoras en la duración total del proceso y una disminución de la eficiencia del sistema en general. Como alternativa para mantener el mismo nivel de prestación y, al mismo tiempo, evitar el overhead necesario para realizar el filtrado antes de cada procesamiento, se ha preferido realizar una implementación basada en triggers. Un trigger es una acción programable que se ejecuta como respuesta a un evento identificable dentro de los datos a través de un conjunto de sentencias SQL que se disparan automáticamente cuando ocurre una operación específica (como un cambio en los datos de una tabla). El trigger está formado por un evento y un acción. En este caso, el evento sería la inserción por parte del Servidor Proxy de una tupla en la tabla «log» y el evento sería ejecutar la consulta anteriormente mencionada que copia la tupla a «log_procesado» si esta cumple con las condiciones especificadas para asegurar su validez.

La Figura 4.8 muestra el filtrado implementado como un trigger en la base de datos SQL Server:

```
CREATE TRIGGER trFiltro ON [log]
FOR INSERT
AS

insert into log_procesado
    select 'http://'+convert(varchar(42),rtrim(desthost))+'/',
       logtime
    from inserted
    where ClientAgent like 'Mozilla%'
    and Protocol='http'
    and ResultCode not in (206,400,401,403,404,501)
    and URI is not null
    and ResultCode <10000
    and ResultCode > 99
```

Figura 4.8: Trigger para Pre-Procesamiento

4.5 Comunicación hacia el Servidor Proxy

Una vez que el módulo de Data Mining, utilizando los conceptos del algoritmo *Apriori* vistos en el Capítulo 2, genera las reglas de asociación, es necesario contar con un método que permita precargar páginas en la caché del Servidor Proxy.

Este método debe poder identificar el tipo de objeto a recuperar y asegurarse que el mismo sea recuperado desde el servidor origen en una versión lo más actualizada posible. Este manejo, además, debe contemplar y ser capaz de trabajar a través de la configuración (tan común en la actualidad) de servidores en cascada. De otra forma, el proxy más cercano al usuario estaría almacenando información desactualizada y esto producirá mayor tiempo de espera ante la necesidad de recargar la página.

El MS Proxy Server carece de una interfase de aplicación (API) que permita controlar los objetos que se hallan almacenados dentro de su caché. De existir tal funcionalidad, sería sumamente sencillo trabajar con la comunicación hacia el Proxy. Por otro lado, la falta de capacidad por parte del Servidor Proxy de recibir instrucciones predefinidas sobre el contenido de su almacenamiento intermedio es una oportunidad para buscar soluciones efectivas y novedosas que permitan cumplimentar el objetivo propuesto.

Una forma de conseguir que se almacenen en la caché del Servidor Proxy los objetos web a voluntad es forzar un pedido desde la red interna hacia el servidor origen, pasando por la estructura Proxy. Si se toman los recaudos necesarios en cuanto a la configuración y formato utilizado para el pedido, puede lograrse que los objetos mencionados queden almacenados en el repositorio del Proxy y, de esta forma, que se encuentren en un medio de acceso más rápido para los usuarios de la red interna. Así, la precarga de objetos puede implementarse símplemente como un pedido GET de HTTP como fue explicado en la Sección 3.3.

Esta solución podría verse con un ejemplo práctico. Imagínese a dos personas utilizando terminales diferentes y accediendo a Internet a través de un mismo Servidor Proxy. El primer individuo ingresa en una página y permanece un momento leyendo su contenido. Durante este lapso de inactividad, la segunda persona realiza pedidos de varias páginas de manera consecutiva y sin intervalos de tiempo para mirar su contenido. Como ambos están detrás de un Servidor Proxy (con las políticas de caché correctamente configuradas) todos los objetos recuperados desde Internet en respuesta a sus pedidos quedan almacenados en el repositorio local. De esta forma, si ahora el primer usuario quisiera entrar a una de las páginas que ya fue accedida por el segundo, la transferencia sería directamente desde la caché del Proxy hasta el navegador del cliente, aprovechando el ancho de banda de la red local (en general 10Mbps) y sin latencias causadas por routers externos o servidores congestionados.

Más aún, si el segundo usuario tuviera una lista de las páginas más comúnmente accedidas por el primero, y solicitara desde su navegador todas estas páginas en el primer momento posible, la mayoría de las páginas que sean accedidas por este último serán cargadas de forma casi inmediata.

Este ejemplo es una analogía del funcionamiento del sistema que proponemos. El primer usuario podría verse como todos los clientes conectados a un Servidor Proxy. El segundo usuario es, en realidad, el servicio que se desea implementar en esta sección (comunicación hacia el Servidor Proxy) y la lista de páginas es el resultado de lo generado por el módulo de Data Mining.

La implementación del mecanismo para obtener páginas almacenadas en la caché puede realizarse simulando un navegador convencional, con la diferencia de que no es necesario realizar un parsing del contenido del objeto, sino simplemente enviar el pedido a través del Proxy y esperar una respuesta. Una vez hecho esto para cada página frecuente (según lo indicado por las reglas de asociación) la caché debería encontrarse pre-cargada de forma de acelerar el acceso de los usuarios a sus páginas más visitadas.

Estos pedidos que emulan el funcionamiento del navegador deben realizarse según el protocolo de transferencia HTTP (ver Capítulo 3). La aplicación que lleve a cabo esta tarea deberá conectarse con el Servidor Proxy a través del protocolo TCP y generar las solicitudes de páginas para pre-cargar. El funcionamiento y manejo de éste último no se detalla aquí, pero puede encontrarse en Heywood [HEI99].

Como se vio anteriormente, el pedido en HTTP se realiza a través del comando GET y un conjunto de headers que permiten definir las características de la operación. Para llevar a cabo la recuperación de los objetos web asegurando al mismo tiempo que sean almacenados en la caché y que su contenido sea el más actualizado posible, se utilizarán los siguientes encabezados:

```
Accept: */*
Host: nombre_de_sitio_origen
User-Agent: WebMining/1.0
Cache-Control: no-cache
Cache-Control: min-fresh=120
```

A continuación se muestra el código de las rutinas principales de un ejemplo desarrollado en Visual Basic 6.0 para recuperar un objeto desde un sitio Web a través de un Servidor Proxy utilizando los headers mencionados, asegurándose que una copia de éste quede almacenado en la memoria caché[16].

La Figura 4.9 incluye las declaraciones necesarias para trabajar correctamente con un socket, utilizando el protocolo TCP y a través del puerto 80 (HTTP). Se ha indicado un puerto local (2500) para realizar la conexión.

16 Deberán tenerse en cuenta los criterios mencionados en la Sección 3.2.2 para saber si un objeto podrá o no ser almacenado en la memoria caché.

```
Private Sub Form_Load()
    Winsock1.RemotePort = 80
       Winsock1.LocalPort = 2500
    Winsock1.Protocol = sckTCPProtocol
End Sub
```

Figura 4.9: Función para establecer parámetros de la conexión

La Figura 4.10 muestra el código necesario para realizar la conexión entre el programa y el Servidor Proxy (llamado ProxySrv), con el servicio que se encuentra escuchando en el puerto 80. El proceso se asegura en una primera instancia que el puerto necesario se encuentre cerrado y posteriormente intenta conectarse, quedando a la espera de una respuesta (winsock1.state) tanto positiva (sckConnected) como negativa (sckError).

```
Private Sub cmdConectarse_Click()
    If Winsock1.State <> sckClosed Then Winsock1.Close
    Winsock1.RemoteHost = «ProxySrv»
    Winsock1.Connect
    Do Until Winsock1.State = sckConnected
        DoEvents: DoEvents: DoEvents: DoEvents
        If Winsock1.State = sckError Then
            MsgBox «Problema al conectarse»
            Exit Sub
        End If
    Loop
End Sub
```

Figura 4.10: Función para establecer la conexión con el Servidor Proxy

La Figura 4.11 muestra el código que termina la conexión TCP:

```
Private Sub Command2_Click()
    Winsock1.Close
    End
End Sub
```

Figura 4.11: Función para terminar la conexión

La Figura 4.12 muestra la realización del pedido al Servidor Proxy, una vez que ya se ha establecido la conexión. Se puede observar la forma en la que el pedido se arma, conteniendo el pedido «GET» y los distintos headers que se utilizan. Una vez que el string conteniendo la totalidad del pedido está listo, se envía a través del socket abierto con el código de la Figura 4.10.

```
Private Sub Winsock1_Connect()
    Dim comando As String
    comando = «GET « + pagina + « HTTP/1.0» + vbCrLf
   comando = comando + «Accept: */*» + vbCrLf
   comando = comando + «Host: « + servidor + vbCrLf
   comando = comando + «User-Agent: WebMining/1.0» + vbCrLf
   comando = comando + «Cache-Control: no-cache» + vbCrLf
   comando = comando + «Cache-Control: min-fresh=120» + vbCrLf
   comando = comando + vbCrLf
    Winsock1.SendData (comando)
End Sub
```

Figura 4.12: Función para realizar el pedido

4.6 Características del Sistema

El sistema, tal como fuera ilustrado en la Figura 4.1, dispone de variables que permiten alterar su funcionamiento. Estos parámetros del sistema son:

a. Soporte (del algoritmo Apriori)
b. Confianza (del algoritmo Apriori)
c. Tiempo de Regeneración de Reglas
d. Tiempo de Pre-carga de objetos
e. Configuración específica del Servidor Proxy

Cada uno de estos parámetros permite configurar un elemento particular del sistema y deben modificarse con cautela porque un cambio mínimo puede traducirse en un aumento considerable de los recursos requeridos y/o en una caída de la perfomance obtenida por el usuario final.

En cuanto al valor de los parámetros (a) y (b), éstos dependen del grado de precisión que se le quiera dar al algoritmo de Data Mining. Se relacionan directamente con el volumen de accesos de la red y con los recursos de hardware disponibles. Se aconseja utilizar confianza=80% y soporte=5,5%

Los parámetros (c) y (d) deben determinarse en cada caso según la cantidad y distribución de los accesos en el tiempo. Esto permite obtener un mejor aprovechamiento de los tiempos «muertos» del sistema.

Por último, el parámetro (e) no necesita ser modificado más allá de lo mencionado en el Capítulo 3, pero en algunos casos peculiares puede ser necesario modificar algunas condiciones de funcionamiento del Servidor Proxy. Por este motivo se deja abierta la posibilidad de, por ejemplo, cambiar la configuración del mecanismo de «Caché Activa» para optimizar la actualización de objetos cacheados.

4.7 Configuración del Servidor Proxy

Junto con la implementación de las funciones mencionadas anteriormente es necesario que el Servidor Proxy se encuentre correctamente configurado para lograr resultados eficientes.

A continuación se enumeran los aspectos de configuración que deben tenerse en cuenta:

* La principal acción que debe realizarse es, lógicamente, habilitar la utilización de la caché por parte del Servidor Proxy. En caso de no hacerse esto, el sistema carecería de todo sentido al no poder realizar precarga de objetos.
* Activar la funcionalidad de caché activa para mejorar la performance a través de la reducción del overhead generado en la actualización de los objetos vencidos. Se prefiere configurar el funcionamiento de la misma para utilizar un algoritmo que tenga en cuenta tanto el tiempo de respuesta como la cantidad de accesos a la red exterior.

Estas opciones se configuran en la pantalla principal de configuración de la caché del MS Proxy Server 2.0, como se ve en la Figura 4.13:

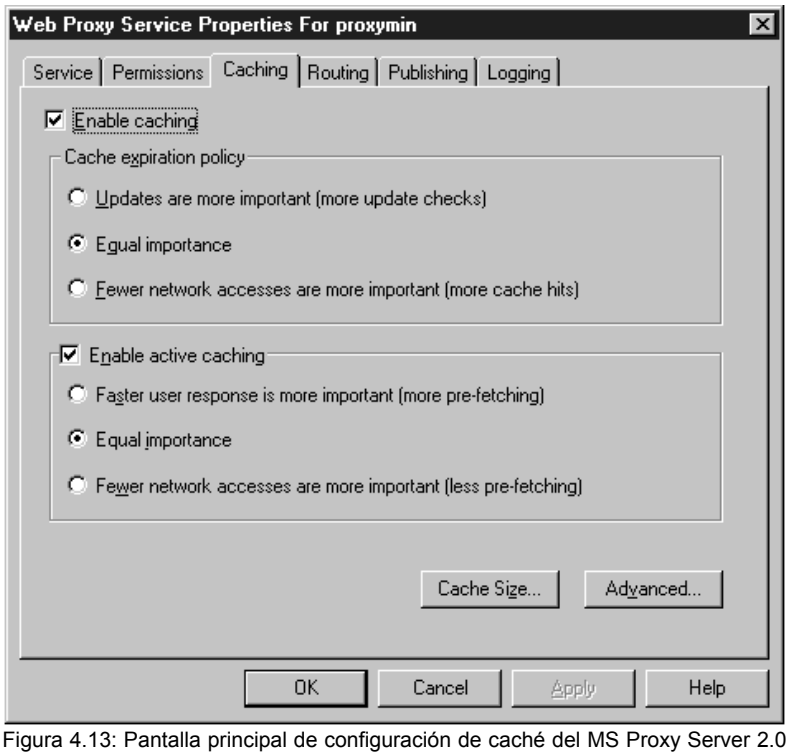


Figura 4.13: Pantalla principal de configuración de caché del MS Proxy Server 2.0

Al establecerse el uso de la caché, debe determinarse qué política se utilizará para la expiración (vencimiento) de los objetos almacenados localmente. Hay dos enfoques extremos: uno valora la actualización sobre la velocidad, realizando comprobaciones constantemente (y, por ende, aumentando el tiempo de respuesta hacia el cliente) y otro que valora el tiempo de respuesta, dejando de lado (en parte) el tiempo que llevan los objetos almacenados localmente. Existe además una opción intermedia que balancea las dos alternativas anteriores. Para el sistema en cuestión se recomienda utilizar esta última opción.

También puede definirse manualmente el tamaño de la caché. Se recomienda utilizar un espacio de almacenamiento lo más amplio posible. Para el sistema que se implementa se está utilizando un espacio de 1000Mb, que es el valor estándar de la instalación de un MS Proxy Server.

Dentro de las opciones avanzadas de configuración, pueden ajustarse los siguientes parámetros:

* Limitar el tamaño de los objetos de la caché no es recomendable porque implicaría no poder hacer precarga de objetos de gran tamaño, que son justamente los que más tiempo tardan en bajarse desde el sitio origen.
* El tiempo de expiración (TTL) de los objetos almacenados puede configurarse según el caso particular. Esta configuración depende del tipo de páginas que sean accedidas y de la carga que tenga la red (relación usuarios/ancho de banda). La ventana que permite modificar estos valores se muestra en la Figura 4.14

Figura 4.14: Pantalla de configuración avanzada de la caché del MS Proxy Server 2.0

Para este sistema en particular no fue necesario realizar modificaciones en el tiempo de vida que el Servidor Proxy asigna a los objetos de la caché. No obstante debe tenerse en cuenta que en implementaciones comerciales de esta solución propuesta, ligeras modificaciones estos parámetros pueden resultar en sustanciales cambios de performance, tanto en beneficio como en detrimento.

Para más información acerca de cómo conseguir la configuración óptima de un servidor proxy puede recurrirse a los siguiente papers de Microsoft: Planificación de Configuración [MSC03], Optimización [MSC04], Integración en una red [MSC05].

4.8 Resumen

En este capítulo se ha analizado qué necesidades debería cumplir el sistema que de desarrolla y se ha explicado como satisfacer cada una de ellas. Entre otros, se han abordado la configuración del Servidor Proxy y los detalles de programación utilizados para lograr el efecto deseado en las solicitudes a través de conexiones Winsock. Toda esta información es suficiente para construir los módulos necesarios que cumplan el objetivo del proyecto.

En el próximo Capítulo se explican las pruebas realizadas con el sistema y se discuten los resultados obtenidos.

5. Evaluación de resultados

Una vez implementada la aplicación propuesta para esta tesina y todos los demás componentes que forman el sistema completo de WebMining, se procedió a realizar una integración de los módulos y, posteriormente, se puso en funcionamiento durante un período de pruebas. Con los datos recolectados durante este tiempo de utilización normal se realizaron evaluaciones estadísticas con el objetivo de ver la utilidad del sistema y extraer conclusiones prácticas respecto del resultado obtenido.

5.1 Objetivos

Las pruebas realizadas tuvieron como objetivo principal determinar las características del sistema desarrollado y comprobar que su funcionamiento cumpla con el objetivo planteado al inicio de este trabajo.

Primero se realizaron pruebas sobre el algoritmo de Data Mining para determinar cómo la variación de los parámetros soporte y confianza afectan los resultados del proceso. Conocer esta relación permitió determinar una configuración adecuada del sistema.

El segundo objetivo que se planteó al realizar las pruebas fue determinar las características que tienen los conjuntos de transacciones que se generan dentro del registro de actividades del Servidor Proxy, a fines de tener una idea genérica de las conductas de los usuarios. Esta información permite ajustar el funcionamiento del sistema y evaluar en qué medida la aplicación de este trabajo conduce a mejoras de performance.

Finalmente, se evaluó el funcionamiento completo del sistema identificando en cada paso si los procesos ocurren de la forma esperada, obteniendo así los resultados planificados inicialmente.

5.2 Entorno del Experimento

Todas las pruebas se realizaron sobre una computadora con procesador AMD K6-2 de 500Mhz, que contenía instaladas al mismo tiempo las aplicaciones de Servidor y las de Cliente. Todas las experiencias realizadas pueden repetirse utilizando como clientes computadoras remotas que accedan a través de TCP/IP al Servidor propiamente dicho. No obstante, en este caso se ha utilizado la mencionada configuración por cuestiones de practicidad y velocidad en la comunicación entre procesos.

Las aplicaciones instaladas en el Servidor fueron:

* Microsoft Windows NT 4.0 Server (con Service Pack 4, 5 y 6)
* Microsoft SQL Server 7.0
* Microsoft Proxy Server 2.0

Las siguientes aplicaciones fueron instaladas en el cliente:

* Navegador (Internet Explorer 5.0) configurado para realizar los pedidos a través del Servidor Proxy.

Por otro lado, el conjunto de datos de prueba utilizados fue generado de forma manual simulando el acceso de usuarios habituales a sitios de Internet. La base de transacciones que se empleó para las experiencias quedó formada por 30 transacciones de 3 usuarios diferentes, dando un total de 287 visitas a sitios diferentes con resultado positivo (es decir, sin contar errores, páginas no accesibles, etc) y de casi 15000 registros en total. Todas estas transacciones se distribuyeron a lo largo de tres semanas y teniendo en cuenta la jorada laboral, es decir, tuvieron lugar de lunes a viernes entre las 9:00 y las 18:00hs.

5.3 Descripción de las Pruebas

Esta sección describe cada una de las pruebas llevadas a cabo.

Prueba 1: Cantidad de Reglas Generadas

Objetivo: Determinar cómo la variación del soporte y la confianza influyen en la cantidad de reglas generadas por el sistema.

Procedimiento: El siguiente procedimiento fue repetido 9 veces, utilizando para la confianza los valores 100%, 80% y 50%, mientras que para el soporte se utilizaron los valores 4%, 5,5% y 7%, teniendo en cuenta todas las combinaciones posibles de estos valores.

a. Se configuró el sistema para utilizar el par de valores deseados.
b. Se ejecutó manualmente el módulo contiene el algoritmo de Data Mining, generándose las tablas temporarias con los itemsets frecuentes.
c. Se ejecutó manualmente el módulo que genera las reglas de asociación.
d. Se realizaó un conteo de la cantidad de reglas generadas.

Resultado Esperado: Obtener un conjunto de valores que muestren cómo el aumento de la confianza disminuye la cantidad de reglas al mismo tiempo que el incremento del soporte produce resultados contrarios. Esto es el resultado que debería obtenerse a partir de la definición de soporte y confianza que se explicó en el Capítulo 2.

Prueba 2: Tiempo Necesario para Generar las Reglas

Objetivo: Determinar cómo la variación del soporte y la confianza influyen en el tiempo de generación del conjunto total de reglas.

Procedimiento: El siguiente procedimiento fue repetido 9 veces, utilizando para la confianza los valores 100%, 80% y 50%, mientras que para el soporte se utilizaron los valores 4%, 5,5% y 7%, teniendo en cuenta todas las combinaciones posibles de estos valores.

a. Se configuró el sistema para utilizar el par de valores deseados.
b. Se registró el tiempo de inicio del proceso (t=0).
c. Se ejecutó manualmente el módulo contiene el algoritmo de Data Mining, generándose las tablas temporarias con los itemsets frecuentes.
d. Se ejecutó manualmente el módulo que genera las reglas de asociación.
e. Se registró el tiempo de finalización del proceso, calculándose la duración del mismo.

Resultado Esperado: Obtener resultados que muestren cómo el soporte y la confianza influyen en los tiempos de generación de las reglas.

Prueba 3: Frecuencia de Acceso a Sitios

Objetivo: Determinar cuáles son las características que se dan en el acceso reiterado a diferentes sitios.

Procedimiento: Primero se determinaron cuáles eran los sitios más frecuentemente accedidos y se contó la cantidad de accesos que se había realizado a cada uno de ellos. En función de esto, se determinó (porcentualmente) qué parte del total de los pedidos contenidos en los datos de prueba correspondían a cada uno de ellos.

Resultado Esperado: Encontrar un grupo reducido de sitios que concentre la mayoría de los accesos.

Prueba 4: Distribución de los Acceso a Sitios

Objetivo: Determinar cuál es el comportamiento de los usuarios en relación a cómo se distribuye la totalidad de pedidos de accesos a sitios a lo largo del tiempo.

Procedimiento: Tomando los accesos diarios que se realizaron, se realizó una acumulación teniendo en cuenta en qué momento del día se produjo cada uno. Posteriormente, se graficó dicha distribución y se calculó una tendencia diaria de accesos.

Resultado Esperado: Encontrar períodos de mayor concentración de pedidos y otros en los cuales la cantidad de pedidos llegue a ser casi nula. Esto permitiría conocer el comportamiento de los usuarios y los momentos en los que se requiere mayor performance del sistema.

Prueba 5: Categorización de los Pedidos al Proxy

Objetivo: Analizar que tipos de pedidos llegan al Servidor Proxy y en qué proporción son realizados. Esto permitirá conocer sobre qué subconjunto del total de los datos almacenados en el registro del Servidor deberá trabajar el sistema.

Procedimiento: Se tomó el conjunto de todos las solicitudes almacenadas en el registro del Servidor Proxy, sin la utilización del filtrado de datos propuesto anteriormente en este trabajo, y se determinaron los diferentes tipos de pedidos y en que proporción contribuían cada uno de ellos al total.

Resultado Esperado: Determinar, de forma general, que porcentaje de los datos almacenados en el registro del Servidor Proxy resultan utilizables para el algoritmo de Data Mining y cuáles son los motivos por los que no todos los pedidos son utilizables.

Prueba 6: Mejora de Tiempos al Utilizar el Sistema

Objetivo: Determinar si la aplicación desarrollada consigue obtener mejoras sensibles en los tiempos de espera que perciben los usuarios.

Procedimiento: A medida que utilizaba el navegador como cliente del sistema para acceder a los sitios, se comprobaba que los diferentes módulos del sistema fueran funcionando en la forma esperada y que las tablas intermedias reflejaran las operaciones que se estaban realizando. Finalmente se realizó una comparación entre los tiempos de acceso a varios sitios antes y después que el sistema generara reglas relacionados a ellos y llevara a cabo la precarga de las páginas correspondientes.

Resultado Esperado: Se espera que el sistema funcione correctamente y que se obtengan mejoras sensibles en los tiempos de acceso.

5.4 Resultados

A continuación se describen los resultados obtenidos en cada una de las pruebas mencionadas en la sección anterior.

Prueba 1: Cantidad de Reglas Generadas

Puede verse en los resultados (Tabla 5.2) que la variación de la confianza es en forma inversamente proporcional a la cantidad de reglas generadas. No obstante la variación del soporte también es inversamente proporcional a la producción de reglas.

SOPORTE	CONFIANZA	CANTIDAD DE REGLAS
4%	100%	472
5.5%	100%	15
7%	100%	2
4%	80%	473
5.5%	80%	16
7%	80%	3
4%	50%	606
5.5%	50%	16
7%	50%	3

Tabla 5.2: Cantidad de Reglas Generadas según el Soporte y la Confianza

De hecho, se observa que el soporte tiene una influencia mucho mayor en la cantidad de reglas generadas, mientras que la variación de la confianza tiene un peso relativamente menor sobre los resultados obtenidos. La Figura 5.1 permite apreciar esta evaluación desde un punto de vista gráfico.

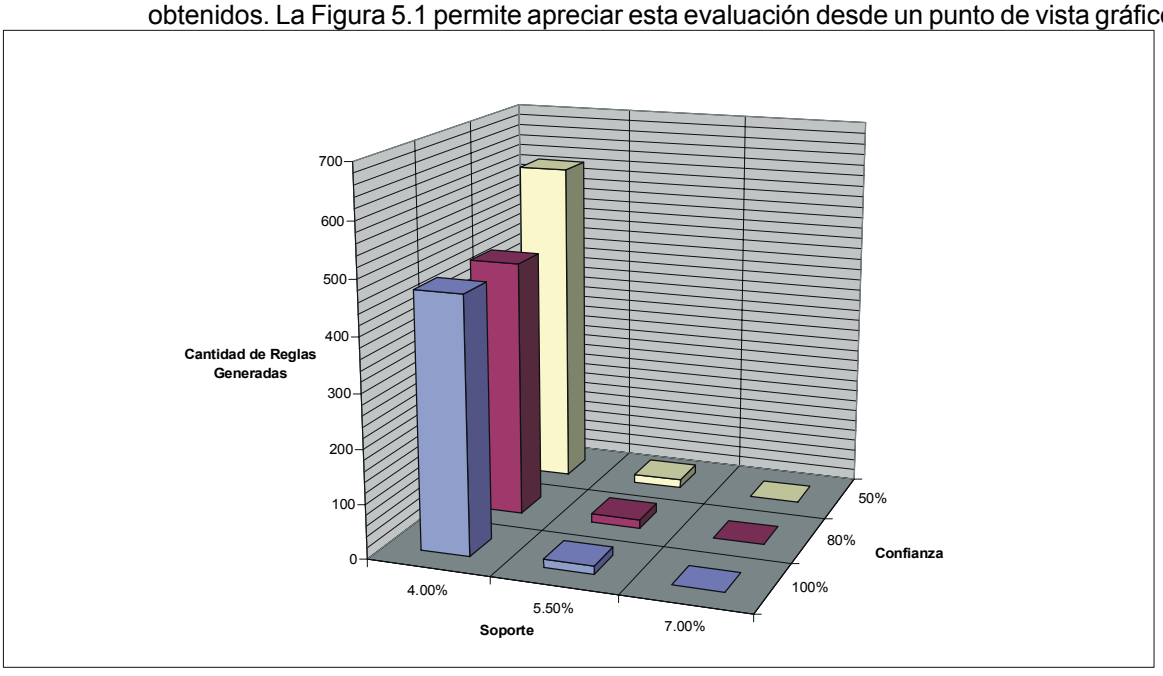


Figura 5.1: Cantidad de Reglas Generadas

Prueba 2: Tiempo Necesario para Generar las Reglas

Sobre la base de los mismos datos utilizados en la prueba anterior, se calculó el tiempo (en segundos) que tardaba el sistema en terminar de calcular todas las reglas que se generaban. De forma análoga a lo ya realizado, se realizaron variaciones en el soporte y la confianza. Los resultados se muestran en la Tabla 5.3:

SOPORTE	CONFIANZA	TIEMPO INVERTIDO
4%	100%	75s
5.5%	100%	3s
7%	100%	1s
4%	80%	70s
5.5%	80%	2s
7%	80%	0s
4%	50%	60s
5.5%	50%	2s
7%	50%	0s

Tabla 5.3: Cantidad de Reglas Generadas según el Soporte y la Confianza

En este caso se observa un cambio substancial respecto de la prueba anterior. Si bien el soporte también afecta de forma inversamente proporcional al tiempo de procesamiento, se observa que la confianza se relaciona a éste de manera directamente proporcional. La Figura 5.2 muestra gráficamente la relación tiempo/soporte/confianza.

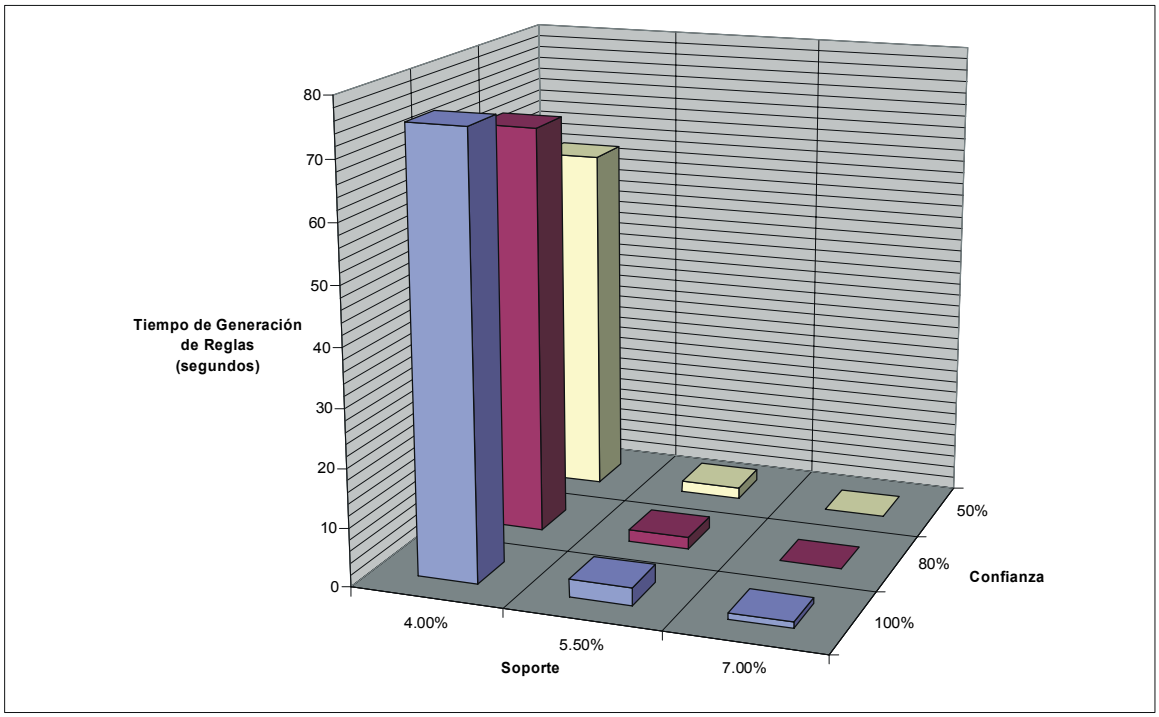


Figura 5.2: Tiempo para la Generación de Reglas

Prueba 3: Frecuencia de Acceso a Sitios

Lo primero que se evaluó fue la frecuencia con la cual los sitios eran consultados. Esto es, determinar qué cantidad de los sitios solicitados por los clientes resultaban accedidos de manera repetida a lo largo del día y cuales eran independientes y aleatorios. Dado que el sistema implementado trabaja a través de la predicción de sitios accedidos frecuentemente, resulta sumamente útil conocer que porción del total de los pedidos son solicitudes recurrentes.

Los resultados fueron los siguientes: el 62% de los sitios son accedidos una sola vez, mientras que solamente el 8% de los pedidos corresponde a páginas que se acceden 5 o más veces. Eso podría indicar que las páginas visitadas por los clientes son el resultado de la utilización de buscadores para satisfacer necesidades concretas.

La Figura 5.3 muestra el porcentaje de sitios que fueron accedidos una cantidad de terminada de veces, es decir, la proporción de los sitios que fueron solicitados una vez, dos veces, tres veces, etc.

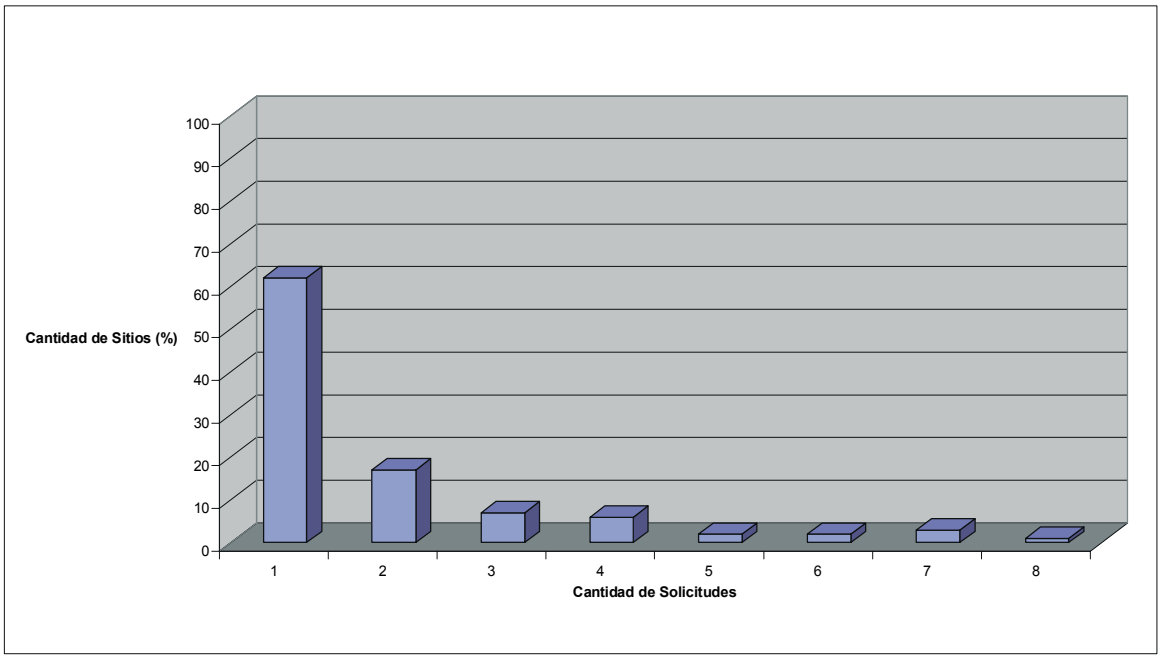


Figura 5.3: Cantidad de Pedidos por Sitio

Analizando en qué proporción los clientes solicitan páginas que tarde o temprano volverán a ser requeridas, los resultados obtenidos indicaron que en el 30% de los pedidos se trataba de sitios que no volverían a ser accedidos. Lógicamente, el restante 70% de las solicitudes se producían sobre sitios que se accedieron al menos dos veces. Esta información es fundamental para determinar la utilidad del sistema de predicción de accesos, que trabaja (en principio) sobre un subconjunto de sitios que conforman el 70% mencionado. Este valor variará según se modifiquen los parámetros de soporte (principalmente) y la confianza.

Prueba 4: Distribución de los Acceso a Sitios

Según lo indicado en la definición de estra prueba en la Figura 5.4 dónde puede verse (a intervalos de 30 minutos) la cantidad de sitios que son accedidos por un conjunto de clientes. Se ha marcado una línea de tendencia para simplificar el comportamiento a lo largo de una jornada laboral. Durante el horario que no se muestra en el gráfico (desde las 18:30 hasta las 9:00) no hay utilización del servicio de Internet.

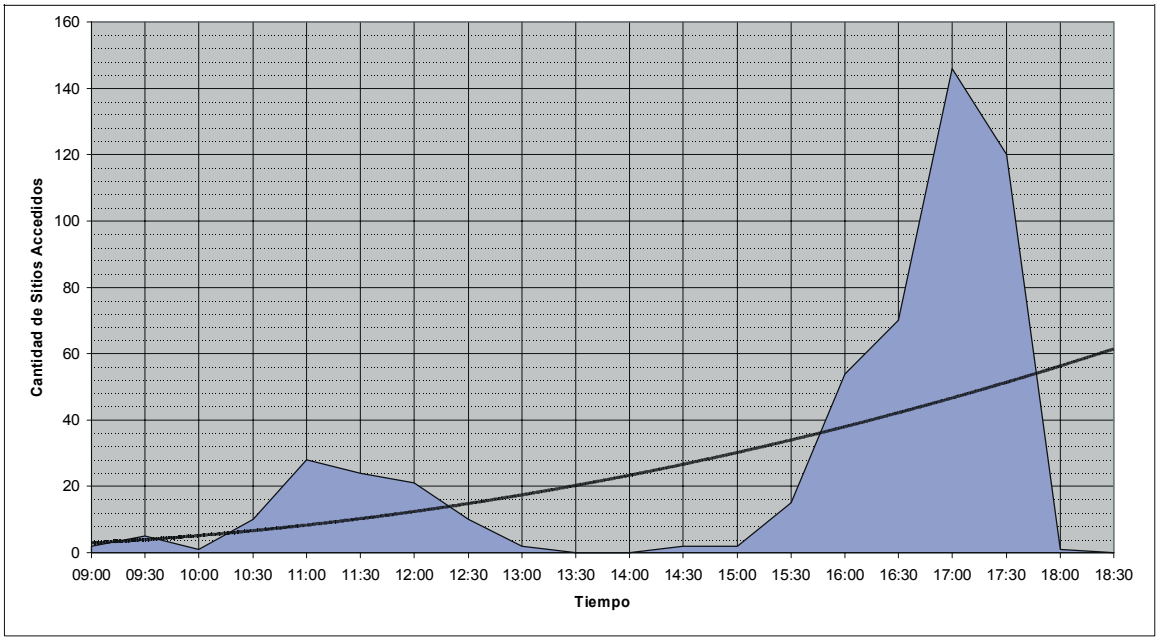


Figura 5.4: Sitios Accedidos en el Tiempo

Se observa que hay un momento de acceso «masivo» que se produce entre las 15:30 y las 18:00, con un pico máximo alrededor de las 17:00hs. Además, se nota cerca de las 11:00hs un aumento ligero en la cantidad de accesos, que continua hasta aproximadamente las 12:00hs, pero que no es significativo respecto del pico máximo antes nombrado. Entre las 13:00hs y 14:00hs hay un período en el cual casi no se producen accesos (posiblemente relacionado con el horario de almuerzo). Hay que tener en cuenta que estos son los valores generales para éste entorno particular, ya que los resultados pueden ser diferentes en otras situaciones.

Prueba 5: Categorización de los Pedidos al Proxy

Como se mencionó anteriormente en este trabajo, los pedidos que los clientes realizan al Servidor Proxy (y que son almacenados en el registro) son de características diferentes. Cada tipo de solicitud puede, además, resultar en diferentes tipos de respuesta. Según se determine el tipo y resultado de un pedido, éste puede considerarse como válido o no para ser tenido en cuenta dentro de la generación de reglas.

Los pedidos que no se tienen en cuenta son:

* Aquellos no producidos por un cliente a través de un navegador (browser). Estos pedidos pueden ser el resultado programas que se ocupan de mostrar publicidades, recoger información de la navegación de los usuarios, actualizaciones automáticas que hace el Servidor Proxy, etc.
* Solicitudes que no sean de páginas web. Aquellos pedidos que no se realizan a través del protocolo HTTP corresponden a conexiones de control del sistema o servicios del tipo FTP que se canalizan a través del Proxy.
* Solicitudes que generaron errores de conexión. Esto puede deberse a condiciones temporarias de Internet o a problemas de configuración del servidor remoto. Utilizar estos pedidos para generar reglas puede producir que, a la hora de pre-cargar la caché el Servidor Proxy se sature de pedidos erróneos.
* Solicitudes que no pudieron completarse («Página no Encontrada» y otros errores similares). No tendría sentido procesar estos pedidos ya que, al demostrarse que son incorrectos y no hacen referencia a ningún objeto concreto de la web, no podrían pre-cargarse en la caché y no se conseguiría así ninguna mejora de performance.

Basándose en el registro de un Proxy Server se generó la Figura 5.5 que muestra las proporciones en las que se dan los distintos tipos de solicitudes mencionadas más arriba. Se muestra la sección separada con el segmento de solicitudes que sí podrán ser utilizadas por el algoritmo de Data Mining.

Además, debe tenerse en cuenta que del 53% de solicitudes correcta hay un promedio del 60% de páginas estáticas (htm, html, etc) que sí pueden almacenarse en la memoria caché y un 40% de páginas dinámicas que no pueden pre-cargarse debido a que se generan cuando el cliente las solicita a través de programas escritos en ASP, Perl, PHP y otros.

Figura 5.5: Categorización de Pedidos al Proxy

Prueba 6: Mejora de Tiempos al Utilizar el Sistema

El seguimiento realizado mostró la correcta interacción entre los diferentes módulos. Primero, la información de navegación fue almacenada en el registro del Servidor Proxy controlada por el filtro que implementa la limpieza de los datos. Posteriormente algoritmo de Data Mining se ejecutó a intervalos de 25 minutos y a continuación el Generador de Reglas produjo las reglas de asociación extraidas del conjunto de transacciones. Con una frecuencia establecida (cada 5 minutos), el módulo encargado de cargar las próximas páginas a ser accedidas recorría las reglas de asociación y realizaba los pedidos necesarios al Servidor Proxy para almacenar los objetos necesarios en la Caché del mismo.

Luego de que se comprobara que se había realizado una secuencia completa del sistema, es decir, que la información de las primeras transacciones ya se había traducido en objetos pre-almacenados en la memoria caché, se procedió a comprobar los tiempos de acceso.

Durante la primera etapa de accesos (es decir, antes de que el sistema entrara en funcionamiento) los resultados estuvieron en el orden de los 200 a los 1100 milisegundos - con un promedio de 600ms - dependiendo esto del tipo de servidor, de su ancho de banda, de la carga de requerimientos de ese momento y del tamaño de cada página. Luego de que el sistema terminara de precargar los objetos necesarios se repitieron las mismas solicitudes. El resultado mostró una mejora significativa en los tiempos de acceso, ya que éstos se redujeron hasta estar entre los 50 y 150ms, con un promedio de 85 ms.

Eso muestra, en promedio, una reducción del 85% del tiempo de espera del usuario desde que solicita la página hasta que se termina de cargar en su navegador.

Cabe aclarar que estas pruebas se realizaron sobre sitios locales (situados en Argentina) ya que el acceso a sitios Internacionales involucra tiempos mucho mayores de acceso (alrededor de los 2 y 3 segundos) debido a la cantidad de routers y enlaces intermedios. No obstante, una vez que las páginas están en la caché, el tiempo de acceso a las mismas es aproximadamente idéntico, dependiendo solamente del tamaño del objeto, sin importar la ubicación original. Esto deja ver que la mejora del 85% en la velocidad que se observa para sitios nacionales es inferior a la que se lograría con sitios extranjeros. Por ejemplo, la página del grupo Megadeth (www.megadeth.com) tarda 4,3 segundos en cargar directamente desde Internet, pero solamente 50ms una vez que se encuentra en la memoria caché.

El gráfico de la Figura 5.6 muestra las diferencias que se dan en los tiempos de acceso para cuatro sitios nacionales, y un promedio de todo el conjunto de datos que se usó como prueba.

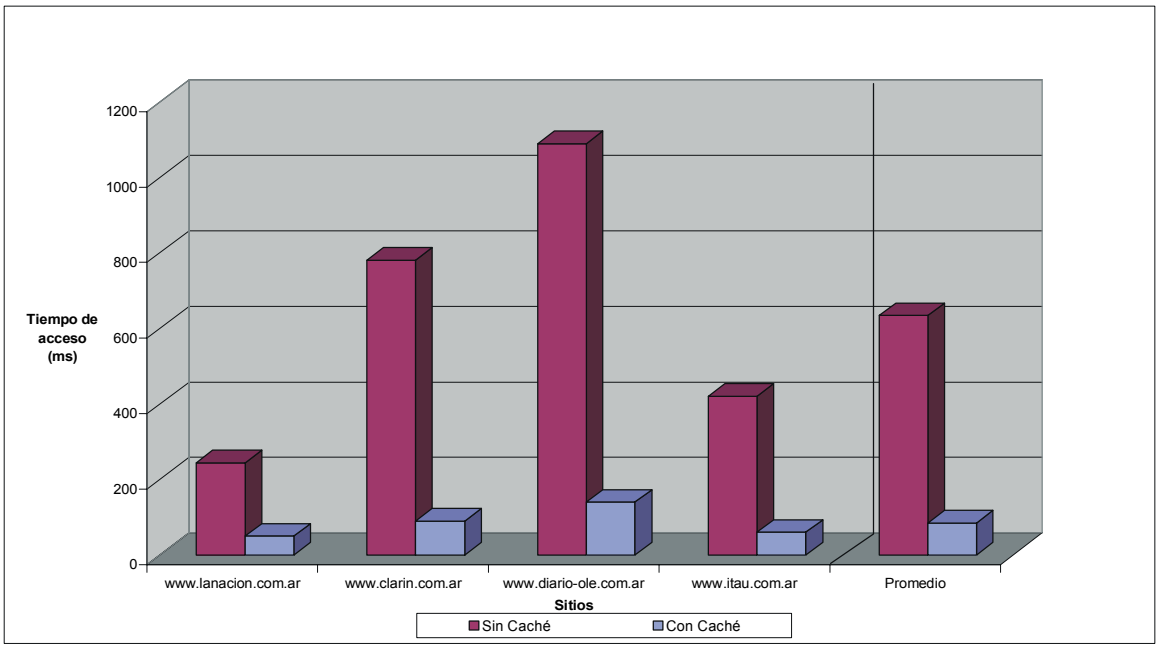


Figura 5.6: Tiempos de acceso antes y después de la pre-carga

Los resultados que figuran «Sin Caché» se refieren al tiempo de acceso que se obtuvo al acceder a las páginas por primera vez, es decir, antes de que hubiera reglas generadas que pudieran provocar su precarga. En cambio, la serie de datos llamada «Con Caché» muestra el tiempo de acceso logrado después que el sistema se ejecutara de forma completa hubiera almacenado en la memoria caché las páginas con alta probabilidad de ser solicitadas.

5.5 Discusión de los Resultados

Prueba 1: Cantidad de Reglas Generadas

En la primera evaluación realizada se trató de determinar la relación directa existente entre los valores asignados a los parámetros «confianza» y «soporte» del generador de reglas.

Determinar estos dos parámetros permite controlar las características que brindará el sistema una vez que se haya puesto en producción. En casos en los que se quiera maximizar las capacidades de «predicción» del sistema, deberán utilizarse valores que aumenten la cantidad de reglas generadas; mientras que si se desea un funcionamiento rápido y que requiera menos recursos, deberá usarse una configuración que genere un reducido conjunto de reglas.

De hecho, se observa que el soporte tiene una influencia mucho mayor en la cantidad de reglas generadas, mientras que la variación de la confianza tiene un peso relativamente menor sobre los resultados obtenidos.

Siempre debe tenerse en cuenta lo discutido en el Capítulo 2 acerca de la importancia del soporte y la confianza dentro del algoritmo Apriori.

Prueba 2: Tiempo Necesario para Generar las Reglas

Los resultados de la prueba anterior sólo son útiles si, al mismo tiempo, se los compara con el tiempo necesario para que la cantidad de reglas determinadas sea generada. No sería correcto asumir que generar más reglas lleva más tiempo, debido a que la duración de éste proceso está determinada por la complejidad de las consultas involucradas.

En este caso se observa un cambio substancial respecto de la prueba anterior. Si bien el soporte también afecta de forma inversamente proporcional al tiempo de procesamiento, se observa que la confianza se relaciona a éste de manera directamente proporcional. Esto quiere decir que debido a la complejidad de consultas que se está ejecutando, hay casos en los que generar más reglas puede resultar en un proceso más rápido que otro que genera pocas.

Prueba 3: Frecuencia de Acceso a Sitios

Como se vio, los algoritmos de Data Mining dependen del conjunto de datos sobre los que se apliquen, es decir, conjuntos de datos diferentes pueden causar resultados opuestos aunque se hayan mantenido los mismos parámetros (confianza y soporte) durante el proceso. Esto lleva a pensar que la correcta configuración del sistema depende específicamente de las características de los datos que vayan a procesar.

Al mismo tiempo, poder definir las tendencias de los accesos que los usuarios van realizando conforme pasa el tiempo permite observar la potencial utilidad del sistema, ya que permite estudiar los patrones de utilización de Internet y las frecuencias y tiempos de accesos a los sitios más populares.

Analizando en qué proporción los clientes solicitan páginas que tarde o temprano volverán a ser requeridas, los resultados obtenidos indicaron que en el 30% de los pedidos se trataba de sitios que no volverían a ser accedidos. Lógicamente, el restante 70% de las solicitudes se producían sobre sitios que se accedieron al menos dos veces. Esta información es fundamental para determinar la utilidad del sistema de predicción de accesos, que trabaja (en principio) sobre subconjunto de sitios que conforman el 70% mencionado. Este valor variará según se modifiquen los parámetros de soporte (principalmente) y con fianza.

Aquí puede hacerse una relación aproximada entre lo observado en el análisis de los datos y la Ley de Pareto, en relación a que el 20% de los sitios conforman el 80% de los pedidos que realizan los clientes.

Prueba 4: Distribución de los Acceso a Sitios

Otro aspecto a tener en cuenta es el nivel de utilización de la salida a Internet. No es lo mismo un sistema pensado para trabajar en un ambiente dónde se accede a 5 páginas distintas por minuto que en uno dónde solamente se visitan 10 páginas en todo un día. Además, también deben tenerse en cuenta la distribución de estos pedidos en el tiempo, para detectar horarios «pico» y tiempos de baja demanda. Esto permitirá determinar intervalos variables de actualización para la regeneración de reglas, de forma de aprovechar los tiempos «muertos» del servidor y liberando recursos durante las horas pico para que los usuarios no pierdan performance.

Se observa que hay un momento de acceso «masivo» que se produce entre las 15:30 y las 18:00, con un pico máximo alrededor de las 17:00hs. Además, se nota cerca de las 11:00hs un aumento ligero en la cantidad de accesos, que continua hasta aproximadamente las 12:00hs, pero que no es significativo respecto del pico máximo antes nombrado. Entre las 13:00hs y 14:00hs hay un período en el cual casi no se producen accesos (posiblemente relacionado con el horario de almuerzo).

De lo anterior puede concluirse que durante el mediodía y después de las 18:30, es decir, cuando ya no se producen solicitudes al servidor serían momentos propicios para que se ejecute el algoritmo que actualiza las reglas de asociación. Al mismo tiempo, dado que estas fases de «calma» se producen luego de momentos de notoria utilización de la salida a Internet, es lógico suponer que el registro de actividades del Servidor Proxy contendrá muchos registros de accesos y esto permitirá que la actualización de reglas tenga en cuenta toda esta información para la generación de las mismas.

Prueba 5: Categorización de los Pedidos al Proxy

Cómo puede verse, solamente el 53% de los pedidos resulta relevante para ser consideradas dentro del conjunto de las transacciones sobre el que se utilizarán las técnicas de Data Mining. Para asegurarse que el sistema trabaje sobre esta porción de las solicitudes es que se implementó el filtrado del registro del Proxy (explicado en el Capítulo 4).

No tener en cuenta este aspecto y considerar todo el conjunto de datos provocaría un aumento en los tiempos de procesamiento (debido a que se estaría trabajando con el doble de información) y se correría el riesgo de generar reglas no válidas y saturar los recursos del Servidor Proxy de una manera inútil.

Prueba 6: Mejora de Tiempos al Utilizar el Sistema

El sistema ha sido probado sin encontrarse problemas para su uso cotidiano. Se utilizó de forma contínua durante una semana sin mostrar caídas de performance ni incompatibilidades con otras aplicaciones.

La reducción conseguida en los tiempos de acceso demuestra dos cosas. En primer lugar que la predicción de páginas resulta efectiva hasta el punto de acelerar los accesos que son reiterados, y además que los tiempos de acceso se reducen considerablemente gracias a la utilización de la memoria caché del Servidor Proxy.

5.6 Aplicación de los Resultados

La realización de las pruebas descriptas permitió comprobar que la definición de los parámetros del sistema para obtener resultados apropiados en su utilización es una tarea de suma importancia. De hecho, se recomienda que antes de poner en producción un sistema de estas características, deberá realizarse previamente una estadística de utilización del servicio, para determinar correctamente las características de configuración. También deberán tenerse en cuenta los tipos de pedidos que llegan al servidor, para verificar que el filtrado de datos propuesto en este trabajo se adecua a esa situación específica o si necesita algun tipo de ajuste.

Principalmente debe encontrarse un equilibro en el funcionamiento total del sistema teniendo en cuenta los factores tiempo y cantidad de forma conjunta sin perder de vista los alcances que se pretenden cubrir con la generación de reglas, es decir, qué tipos de reglas se desean generar y cuales son los umbrales necesarios para que un conjunto de acciones de los usuarios genere reglas. En este sentido, no pueden darse parámetros estándar para realizar la configuración, ya que dependen exclusivamente de las características del entorno al que se aplique el sistema.

Por otro lado, en estas pruebas los períodos de regeneración de reglas y de precarga de páginas fueron establecidos ex profeso de forma periódica a intervalos de tiempo cortos. Esto fue así debido a que las pruebas se realizaban con una utilización contínua del servicio. En casos más realistas, como se muestra en el análisis de la frecuencia de uso, es posible detectar picos de máxima utilización y períodos donde el sistema se encuentra totalmente inactivo. Esto lleva a pensar que para mejorar la performance general de la implementación podría pensarse en realizar una pre-carga masiva de objetos en la caché del Proxy momentos antes del inicio de los períodos críticos, exactamente después de que todas las reglas de asociación (que ya tendrán en cuenta los patrones obtenidos en el último pico de accesos) están generadas y listas para usarse.

De esta forma se logra una utilización balanceada del servidor y se intenta mejorar los tiempos de respuestas que obtienen los usuarios del servicio en general de acceso a Internet.

5.7 Resumen

Este capítulo ha presentado las pruebas que se realizaron, las conclusiones que se extrajeron de las mismas y conceptos que permiten aplicar ese conocimiento a futuras aplicaciones del sistema.

6. Conclusiones y trabajos futuros

Después de haber desarrollado, construido y probado el sistema completo, podemos concluir lo siguiente:

** El sistema cumple con el objetivo de acelerar el tiempo de respuesta que obtienen los usuarios, al intentar acceder a páginas que le son «frecuentes».*

A través de la generación de reglas y de la precarga de las páginas con posibilidad de ser re-accedidas, los pedidos de objetos que ya fueron almacenados en la caché se resuelve de una forma substancialmente más rápida que en los casos en los que no se implementa el sistema. El empleo de este almacenamiento intermedio mejora la velocidad tanto para el usuario que accede a páginas precargadas como para el resto de los usuarios que ven más ancho de banda disponible para tramitar pedidos no previstos por el algoritmo de Data Mining. Esto último se debe a que los pedidos que pueden satisfacerse directamente en el Servidor Proxy no se propagarán hasta el servidor origen de los datos (ver Sección 3.3) y de esta forma el ancho de banda total se reservará exclusivamente a aquellos pedidos que deban ser enviados hacia el host ubicado en Internet.

Por cuestiones de implementación (el programa fue escrito en TransactSQL) el algoritmo de Data Mining tiene un nivel de eficiencia que podría mejorarse. Por ejemplo, podría reescribirse en un lenguaje de mayor eficiencia (como Visual C+) lo cual seguramente potencie la performance total del sistema.

** Los beneficios que se obtienen con el sistema dependen principalmente del tipo de usuarios.*

Para usuarios que realizan búsquedas constantes sobre temas variados no resultará posible determinar patrones de accesos. En cambio, con usuarios que tengan costumbres y «sitios favoritos» se obtendrán mejores resultados. Para éstos, las ventajas de implementar el sistema serán más notorias por los cortos tiempos de respuesta en las páginas «habituales», mientras que para los primeros la única ventaja que encontrarán será la mayor disponibilidad de ancho de banda, debido a que no se procesan muchos de los pedidos del otro grupo.

** La performance y nivel de resultados que brinda el sistema depende directamente de la correcta configuración de sus parámetros (confianza, soporte, tiempo de regeneración de reglas, tiempo de precarga de objetos, configuración del proxy).*

Como se vió en el Capítulo 5, la variación de los parámetros del sistema (en particular el «soporte» y la «confianza») puede producir cambios notorios tanto en la cantidad de reglas generadas como en los tiempos necesarios para generar las reglas. Estos cambios, sumados al volumen de acceso a sitios de Internet que tenga una red local puede hacer que los tiempos de procesamiento se hagan extremadamente grandes como para brindar un servicio útil al usuario.

Además es de recalcar que el sistema implementado es de fácil instalación y portabilidad debido a su característica modular y a utilizar un Servidor Proxy ampliamente conocido del mercado, el sistema resulta fácil de instalar en otro entorno y de soporte sencillo para usuarios con experiencia mínima de administración de redes.

Las características de configuración que se han utilizado en este trabajo (ver Capítulo 5) han demostrado funcionar con una exigencia no muy alta de recursos, pero generando resultados interesantes. Utilizando hardware más poderoso podría incrementarse la capacidad del algoritmo de Data Mining para generar reglas más específicas y purificadas.

Por otro lado, la construccion modular permite modificar algunos procedimientos sin tener que alterar el funcionamiento total del sistema. Incluso pueden reescribirse componentes y, mientras se mantenga la interfase, no se necesita modificar lo ya implementado. Gracias a esto pueden pensarse en realizar modificaciones al sistema que mejoren su eficiencia y performance. Entre ellas pueden citarse:

* Reescribir el algoritmo de Data Mining en un lenguaje como C++ (ya mencionado).
* Desarrollar un Servidor Proxy propio, de forma de poder manejar una interfase a medida entre éste componente y el resto del sistema.
* Desarrollar un Servicio de Almacenamiento de Páginas (caché) que sea administrado por el propio sistema y que permita mejorar la eficiencia en la administración de objetos web sin necesidad de tener que usar el incorporado al Servidor Proxy.
* Ampliar el Generador de Reglas para poder trabajar con otros tipos de reglas de asociación, como ser reglas multidimensionales.

Anexo I: Esquema del desarrollo de un servicio de caché

Como se mencionó en el Capítulo 4, una posibilidad de ampliación de este trabajo consta de la implementación de un Servidor Proxy propio. Construir dicho Servidor se puede traducir como el desarrollo de los siguientes módulos:

1. Interfase con los clientes.
2. Interfase con la red exterior (es decir, con servidores web, servidores proxy, gateways, etc).
3. Administrador de Multiplexación.
4. Servicio de Caché.

El objetivo de este anexo es exponer un posible desarrollo del Servicio de Caché para utilizar como base a una futura implementación de este trabajo, o para otro proyecto relacionado.

I.1 Introducción

Construir un Servicio de Caché implica tener en cuenta los siguientes aspectos:

* Los objetos almacenados en la caché deben mantener una forma estándar que permita su administración y manejo.
* El proceso de purging que implica el limpiado la caché (para evitar que se llene hasta el límite de la memoria disponible)
* Establecer un proceso de coordinación que ejecute la limpieza de objetos poco utilizados en determinados intervalos de tiempo.
* El acceso simultáneo de varios clientes puede producir caídas de performance debido al almacenamiento de objetos duplicados dentro de la memoria.
* Determinar la duración del tiempo de vida de cada objeto antes que caduque y sea necesario recuperar un nuevo objeto desde el Servidor original.
* Establecer la forma de proceder en caso de que, estando la caché vacía, dos clientes soliciten el mismo objeto de forma simultánea. Determinar si ambos deben intentar dejar una copia almacenada.
* Analizar qué ocurre con los objetos de la caché que nunca son accedidos.

Todos los pedidos de objetos deben primero verificarse a través del servicio de caché antes de intentar ser cargados desde el exterior. Esto significa que siempre debe buscarse en la caché antes que nada. De no hacerse así, la ganancia en performance puede ser nula. Si el objeto existe dentro de los almacenados por la caché, entonces debe verificarse que no se encuentre vencido. Si no se encuentra en la caché, o si su tiempo de vida ha caducado, el objeto debe recuperarse desde el Servidor original y una copia de este debe almacenarse.

Una buena implementación de la caché debería cumplir con los siguientes requisitos:

1. Cualquier aplicación puede acceder al servicio de caché
2. Existe un método para colocar objetos en la caché
3. Existe un método para recuperar objetos desde la caché
4. Los objetos pueden determinar si han expirado. Debido a la diferente naturaleza de los objetos, sería incorrecto plantear que todos siguen la misma política de expiración.
5. Un proceso de baja prioridad que se ejecuta continuamente deberá detectar los objetos que se han vencido para retirarlos de la caché.

I.2 Implementación

El requerimiento 1 puede satisfacerse si se realiza la implementación de la caché en un entorno Java, permitiendo que los objetos que contienen la funcionalidad del servicio sean accedidos por otras aplicaciones desarrolladas bajo las mismas condiciones. De esta forma, se pueden aprovechar los beneficios de una programación totalmente orientada a objetos.

Desarrollando métodos públicos de recuperar (get) y almacenar (set) en el servicio de caché se permitiría a cualquier módulo (u objeto) desarrollado un control total sobre el contenido de la caché, cumpliendo así con los requerimientos 2 y 3.

Para poder cumplir el requerimiento 4 (que cada objeto pueda determinar si ha expirado) se necesita un método público que pueda ser consultado en cada objeto. Esto implica que todos los objetos deben adaptarse a ciertas reglas comunes.

Para cumplir con el requerimiento 5 se puede implementar una programación multi-threaded, en dónde un hilo (thread) de programa se ocupe de recorrer la lista de objetos que se encuentran almacenados para verificar su estado de validez y desechar aquellos que se hayan vencidos. Este hilo deberá ser declarado como estático (static) dentro del objeto que administra la caché. Estas reglas que gobiernan los objetos almacenados podrían ser:

1. Todos los objetos deben tener un método público llamado isExpired(), que devuelve un valor lógico (booleano) indicando si ha expirado o no.
2. Todos los objetos deben tener un método público llamado getIdentifier() que devuelva un valor que permita diferenciar a ese objeto de todos los demás ubicados en la caché.

La Figura I.1 es una especificación en UML de una interfase (Cacheable) que contiene las reglas mencionadas anteriormente y una clase (CachedObject) que implementa la interfase y es la que representa a los objetos que se almacenan en la caché:

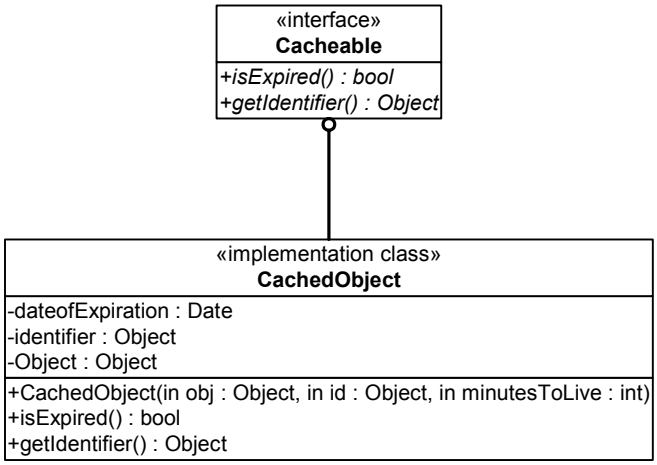


Figura I.1: Representación de la interfase Cacheable y la clase CacheObject

La Figura I.3 muestra la clase que realiza la administración de la caché. Utiliza un objeto tipo HashMap (incluido en las librerías de Java) para mantener una lista de todos los objetos que se encuentran almacenados en la caché. Contiene además el método estático bgThread implementado como un hilo de ejecución y que se encarga del procedimiento de purging. Finalmente, soporta además los métodos públicos put y get necesarios para agregar y recuperar objetos desde la caché.

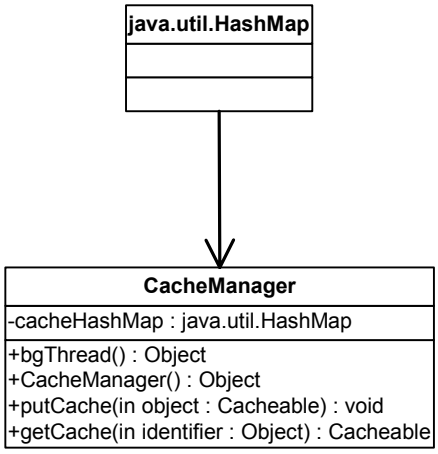


Figura I.2: Representación de la clase CacheManager

I.3 Resumen

Este anexo brindó los lineamientos generales que podrían seguirse en caso de embarcarse en la tarea de ampliar lo realizado en este trabajo a través del desarrollo de un Servidor Proxy propio. Como último comentario sugerimos leer [LOU97] que contiene una descripción de diferentes arquitecturas para el desarrollo de un Servicio de Caché.

Referencias

[AGR93] R. Agrawal, T. Imielinski y A. Swami. Mining association rules between sets of items in large databases. ACM SIGMOD Conference on Management of Data, Washington D.C., 1993.

[AGR94] R. Agrawal y R. Srikant. Fast Algorithms for Mining Association Rules. 20th International Conference on Very Large Databases, Santiago de Chile, 1994.

[BOR00] J. Borges and M. Levene. Data mining of user navigation patterns. Lecture Notes in Artificial Intelligence (LNAI 1836), Berlin, 2000

[BRI97] S. Brin, R. Motwani, J. Ullman y S. Tsur. Dynamic itemset counting and implication rules for market basket data. SIGMOD Record (ACM Special Interest Group on Management of Data), 26(2):255, Abril de 1997.

[BRO97] A. Broder, S. Glassman y M. Manasse. Syntactic clustering of the Web. 6th International World Wide Web Conference, 1997.

[BUC99] A. Bchner, M. Baumgarten, M. Mulvenna, S. Anand y J. Hughes. Navigation Pattern Discovery from Internet Data, ACM Workshop on Web Usage Analysis and User Profiling (WebKDD'99), 1999.

[CHY96] M. Chen, J. Han y P. Yu. Data mining: An overview from database perspective. IEEE Transactions on Knowledge and Data Eng., 8(6):866—883, Diciembre de 1996.

[GRA96] J. Gray, A. Bosworth, A. Layman y H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. 12th ICDE, páginas 152—159, New Orleans, Marzo 1996.

[HAN01] J. Han y M. Kamber. Data Mining, Concepts and Techniques. Morgan Kaufmann Publishers, 2001.

[HAN95] J. Han y Y. Fu. Discovery of multiple-level association rules from large databases. 21st International Conference on Very Large Databases, Zurich, Septiembre de 1995.

[HEI99] D. Heywood. Redes con Microsoft TCP/IP. Prentice Hall, 1999.

[HID99] C. Hidber. Online Association Rule Mining. SIGMOD Conference, Philadelphia, 1999.

[INM96] W. Inmon. Building the Data Warehouse. J. Wiley & Sons, Inc., segunda edición, 1996

[KIE98] J. Kleinberg. Authoritative sources in s hyperlinked environment. ACM-SIAM Symposium on Discrete Algorithms, 1998.

[LAN99] B. Lan, S. Bressan, B. Ooi y Y. Tay. Making Web Servers Pushier. Workshop on Web Usage Analysis and User Profiling (WEBKDD'99), Agosto de 1999.

[LUO97] A. Luotonen. Web Proxy Servers. Prentice Hall, 1997.

[MEN00] A. Mendelzon y D. Rafiei. What do Neighbours Think? Computing Web Pages Reputation. IEEE Data Engineering Bulletin, 2000.

[MHB98] J. Moore, E. Han, D. Boley, M. Gini, R. Gross, K. Hastings, G. Karypis, V. Kumar y B. Mobasher. Web page categorization and feature selection using association rule and principal component clustering. 7th Workshop on Information Technologies and Systems, Diciembre de 1997.

[MSC01] Microsoft Corporation. ODBC Error - DSN may be Invalid. Product Support Services. 1999.

[MSC02] Microsoft Corporation. Proxy Server 2.0 Service Pack 1: List of Fixes. Product Support Services. 2000.

[MSC03] Microsoft Corporation. Planning an Effective Proxy Server Configuration. TechNet. 2000.

[MSC04] Microsoft Corporation. Monitoring and Optimizing Proxy Server. TechNet. 2000.

[MSC05] Microsoft Corporation. Integrating Proxy Server with the Network. TechNet. 2000.

[MSC06] Microsoft Corporation. Microsoft Proxy Server Caching Criteria and Troubleshooting. Product Support Services. 2000.

[PAR95] J. Park, M. Chen y P. Yu. An effective hash algorithm for mining association rules. ACM - SIGMOD Conference, San Jose, California, 1995.

[PEI00] J. Pei, J. Han, B. Mortazavi-Asl y H. Zhu. Mining Access Patterns Efficiently from Web Logs. Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'00), 2000.

[RFC791] J. Postel. RFC 791: Internet Protocol: Darpa Internet Program Protocol Specification. Septiembre de 1981.

[RFC793] J. Postel. RFC 793: Transmission Control Protocol: Darpa Internet Program Protocol Specification. Septiembre de 1981.

[RFC1341] N. Borenstein y N. Freed. RFC 1341: MIME (Multipurpose Internet Mail Extensions). Junio de 1992.

[RFC1928] M. Leech, M. Ganis, Y. Lee y otros. RFC 1928: SOCKS Protocol Version 5. Marzo de 1996.

[RFC1945] T. Berners-Lee, R. Fielding y H. Frystyk. RFC 1945: Hypertext Transfer Protocol - HTTP/1.0. Mayo de 1996.

[RFC2068] R. Fielding, J. Gettys, J. Mogul y otros. RFC 2068: Hypertext Transfer Protocol - HTTP/1.1. Enero de 1997.

[SAV95] A. Savarese, E. Omiecinski y S. Navathe. An efficient algorithm for mining association rules in large databases. 21st VLDB Conference, 1995.

[TOI96] H. Toivonen. Sampling large databases for association rules. 22nd VLDB Conference, 1996.

[ZXH98] O. Zaiane, M. Xin y J. Han. Discovering Web Access Patterns and Trends by Applying OLAP and Data Mining Technology on Web Logs. Advanced in Digital Libraries(ADL'98), Santa Barbara, CA, Abril de 1998.