UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

BUSCADOR SEMÁNTICO PARA COMERCIO ELECTRÓNICO

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS MENCIÓN COMPUTACIÓN

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

ANDRÉS EDUARDO BILBAO BERNALES

PROFESOR GUÍA:

CLAUDIO GUTIÉRREZ GALLARDO

MIEMBROS DE LA COMISIÓN:

CARLOS HURTADO LARRAÍN
PABLO BARCELÓ BAEZA
MARCELO ARENAS SAAVEDRA

SANTIAGO DE CHILE

NOVIEMBRE 2010

RESUMEN

El presente trabajo tiene como objetivo desarrollar un buscador de productos de comercio electrónico, orientado a las tiendas en línea chilenas, y que provea facilidades para identificar productos iguales y similares.

Los buscadores de comercio electrónico concentran productos ofrecidos por diferentes sitios otorgándo al usuario una interfaz de consulta única, y mostrando como resultado productos de diferentes tiendas en un listado unificado. Todo esto con el objetivo básico de comparar precios y ayudar al usuario en la elección de su compra.

BuscaPé y Confronte son los únicos buscadores de productos de comercio electrónico orientados a tiendas en Chile. Su principal deficiencia es que no tienen un modelo genérico para identificar automáticamente los productos iguales, por lo que identifican los productos iguales sólo para algunas categorías. Por otro lado, en las categorías que se identifican los productos iguales es difícil comparar productos de similares características, ya que el resultado de una búsqueda es una lista de productos sin orden y sin más información que el título del producto.

En estos sitios la única forma de saber si dos productos tienen características similares es ver los detalles de los productos, o seleccionar la opción "Comparar"que presenta una matriz de comparación con los valores de algunas características para cada producto. El problema de la matriz es que el usuario no sabe de antemano las características de los productos, y por lo tanto debe usar prueba y error para formar una matriz de comparación que realmente sirva para ayudar en una decisión de compra.

Como parte del trabajo realizado en la presente tesis se desarrolló un buscador de productos de comercio electrónico denominado YOSH. YOSH es un meta buscador de comercio electrónico, cuya informaciíon se estructura en una ontología, y que posee la particularidad de detectar y recomendar productos similares basándose en medidas de distancia entre productos.

La construcción de este buscador contempla el desarrollo de cinco módulos que forman el sistema de recuperación de información completo: Crawler, Extractor de información, Clasificador, Motor de recomendación, Interfaz de consulta. Cada uno de los módulos presenta problemáticas y áreas de estudio particulares, y los aportes de este trabajo están relacionados directamente cada uno de estos módulos.

En consecuencia los aportes de esta tesis son cinco: (a) Un framework para implementación de Crawlers de comercio electrónico, (b) Una Ontología básica para comercio electrónico, (c) Un clasificador SVM para asociar productos a categorías (d) Una estrategia para detección de productos productos iguales y similares usando medidas de distancia entre prductos. (e) Una aplicación Web que provee una interfaz para búsqueda de productos por keyword y navegación por categorías.

Dedico este trabajo a mi hija Amaia, a Marysol,
mi familia y mis amigos.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

La masificación de Internet en la ultima década y su diversificación de contenidos, han atraído a la industria del comercio, la cual ha obtenido excelentes resultados utilizando internet como canal de ventas. El año 2008 las ventas del comercio electrónico en Chile en formato Business to Consumer(B2C) superaron los US$ 380 millones, mostrando un 20 % de crecimiento en el periodo 2006-2007 y un 27 % en el periodo 2007-2008, según el estudio "Economía Digital" [30] realizado anualmente por la Cámara de Comercio de Santiago(CCS).

Por otro lado, según el estudio WIP Chile 2008 [25] los usuarios de Internet en Chile ascienden a los 7 millones, lo que corresponde al 48 % de la población. De éstos, 28 % realizó al menos una compra en línea en los últimos 12 meses, lo que equivale aproximadamente a 2,1 millones de consumidores en línea. Además el 47 % de los usuarios chilenos cotiza y compara productos en internet antes de comprar en tiendas físicas.

En este escenario de crecimiento en ventas de comercio electrónico destacan los sitios de subastas y tiendas de retail. Estas últimas han visto incrementadas sus ventas lanzando tiendas virtuales que administran como una sucursal más, y que han aumentado sus ventas año a año.

En general las tiendas que proveen ventas en línea, ofrecen una gran cantidad de productos

organizados por categorías. Debido a esto cada sitio provee sistemas de búsqueda en sus catálogos, principalmente basándose en keywords, marca y SKU (número de referencia único para cada producto según las normas de cada tienda). A estos sistemas de búsqueda se les denomina E-commerce Search Engine (ESE).

El resultado final es que el usuario tiene una gran cantidad de productos disponibles para comprar por internet, pero sólo puede acceder a un conjunto limitado de estos. Este límite depende del conocimiento de tiendas en línea que posea el usuario. Por otro lado esta gran cantidad de productos disponibles están distribuidos en distintos portales, cada uno con distintas formas de estructurar la información, lo que dificulta la comparación de productos entre distintas tiendas y atenta contra la elección de la mejor opción por parte del usuario. Por estas razones aparecieron los E-commerce Metasearch engine (EMSE)

Los EMSE son meta buscadores que concentran productos de diferentes sitios de comercio electrónico, otorgándole al usuario una interfaz única de consulta, y mostrando como resultado productos de diferentes tiendas en un listado unificado. Todo esto con el objetivo básico de comparar precios y ayudar al usuario en la elección de su compra.

BuscaPé Chile [3] y Confronte [5] son los únicos EMSE orientados a tiendas de comercio electrónico en Chile. Su principal deficiencia es que no tienen un modelo genérico para identificar automáticamente los productos iguales y vendidos por distintas tiendas. Debido a esto es que identifican los productos iguales sólo para algunas categorías. Por otro lado, en las categorías que se identifican los productos iguales es difícil comparar productos de similares características, ya que el resultado de una búsqueda en esas categorías es una lista de productos sin orden y sin más información que el título del producto.

En estos sitios la única forma de saber si dos productos tienen características similares es ver los detalles de los productos, o seleccionar la opción “Comparar”que presenta una matriz de comparación con los valores de algunas características para cada producto en la matriz, lo que

facilita en parte la experiencia del usuario. El problema de la matriz es que el usuario no sabe de antemano las características de los productos, y por lo tanto usa prueba y error para formar una matriz de comparación que realmente sirva para ayudar en una decisión de compra. Este proceso es equivalente a ver la ficha de cada producto, y anotar las características principales en un papel o planilla de cálculo, para luego realizar un análisis visual de la mejor opción.

En el mundo los EMSE más reconocidos son PriceGrabber [9] y Ciao [4]. Estos presentan funcionalidades muy similares a las de BuscaPé y Confronte, por lo que padecen de los mismos problemas al momento de comparar productos.

Tomando en cuenta lo expresado en el párrafo anterior queda en evidencia que es necesario mejorar las técnicas para comparar productos en los EMSEs, principalmente aportando información al usuario para que realice comparaciones entre productos que sean similares, sin tener que ver la ficha de cada producto.

Este trabajo tiene como objetivo desarrollar un buscador de productos de venta en línea, orientado a las tiendas de comercio electrónico en Chile, que provea facilidades para identificar productos iguales y productos similares. Las conribuciones asociadas al trabajo son: un método para detección de productos iguales o similares y la forma en que se despliega la información en pantalla, la que permite tomar decisiones sin necesidad de seguir navegando.

Entre las tareas a desarrollar destaca el diseño de un modelo que permita obtener la similitud entre dos productos basándose en las características, de tal manera de agrupar los productos que tengan alta similitud y reflejar dicha agrupación en el resultado de las búsquedas realizadas por el usuario. Esto resulta fundamental para mejorar la experiencia de un usuario que desea comparar productos.

Además se contempla el desarrollo de estrategias para recolección y estructuración de información desde las tiendas, y la implementación de una aplicación Web que provea una interfaz de consulta para búsqueda de productos por keyword, y navegación por categorías.

La presente Tesis está organizada de la siguiente manera: El Capítulo 2 presenta el trabajo relacionado y los antecedentes ocupados para realizar esta tesis. Luego en el Capítulo 3 se describen los objetivos del presente trabajo. El diseño e implementación del buscador se describe en el Capítulo 4. En el Capítulo 5 se detalla el funcionamiento del motor de recomendaciones, que es el módulo encargado de detectar la similitud entre productos. Finalmente el Capítulo 6 presenta las conclusiones y el trabajo futuro.

Capítulo 2

Trabajo Relacionado

En este capítulo se analizan trabajos relacionados a esta tesis y se discuten los métodos empleados, con sus virtudes y falencias. En primer lugar se describen las características de un Buscador Web como un sistema de recuperación de información, identificando las técnicas más usadas para el funcionamiento de sus componentes. Luego se explica el concepto de Web Semántica y sus aplicaciones en la mejora de motores de búsqueda. Finalmente se describe el funcionamiento de los E-commerce Metasearch Engine(EMSE), identificando las falencias y posibles mejoras que presentan los EMSEs más reconocidos.

2.1. Buscadores Web

Un Buscador Web es un sistema de recuperación de información cuyo espacio de búsqueda está constituido por un conjunto de recursos obtenido desde la Web. La gran diferencia entre los Buscadores Web y otros sistemas de recuperación de información, es que todas las consultas se deben responder sin acceder directamente al texto que compone el espacio de búsqueda [18]. Esto ocurre básicamente por la gran cantidad de información que compone la Web y su crecimiento

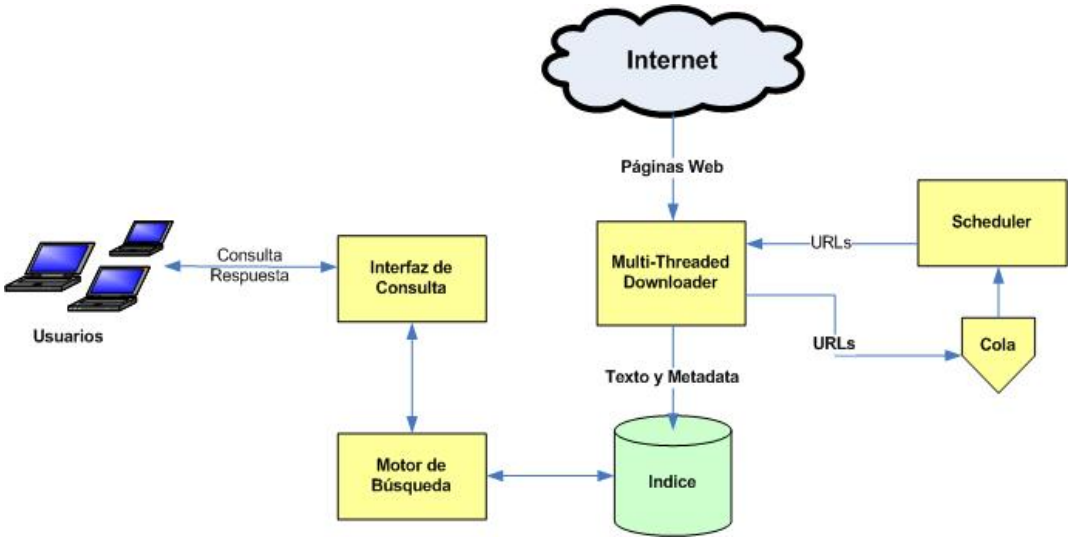

Figura 2.1: Arquitectura general de un buscador basado en Crawler-Indexador

diario.

La arquitectura general de un buscador usa el modelo crawler-indexador. Crawlers son programas que navegan por la Web en busca de información nueva o actualizada dentro de las páginas o recursos que son visitados. Esta información es enviada a un servidor donde es indexada. Por otro lado la interfaz de consulta obtiene la información desde el índice, usando distinos modelos de recuperación. En la figura 2.1 se muestra la arquitectura general de un buscador.

2.1.1. Crawler

Un crawler es un programa que obtiene recursos Web, usualmente para ser usado por Motores de Búsqueda [60]. En términos prácticos un crawler comienza con una URL inicial, lee la página asociada a dicha URL y extrae todas las URLs que encuentre en su contenido. El procedimiento de lectura y extracción sigue sucesivamente con las URLs obtenidas. En [27] se estudia el orden en que se recorren las URLs para mejorar la eficiencia del proceso.

Actualmente existen varios trabajos que describen distintas arquitecturas de Crawlers, usadas

para distintos fines. La descripción general de la arquitectura de un Crawler se puede encontrar en [23], y en [26] se describe la implementación de un Crawler paralelo, que permite optimizar el tiempo del proceso de crawler usando múltiples instancias. Por último en [17] se describe el problema de recorrer páginas generadas dinámicamente y modelos para afrontar este problema.

Los crawlers son unos de los pilares de los Sistemas de búsqueda. Las técnicas de crawling usadas por los grandes buscadores como Google [7] o Altavista [1] no son de dominio público, ya que cualquier mejora es una ventaja competitiva que no es deseable compartir.

2.1.2. Modelo de Recuperación

Un modelo de recuperación consiste principalmente en la representación de los documentos y la información necesitada por el usuario. Los documentos corresponden a los recursos que forman el espacio de búsqueda del Sistema. La información necesitada por el usuario es la consulta que realiza el usuario al Sistema. Completan un modelo las funciones que permiten relacionar y generar ranking entre documentos y consultas.

El modelo booleano [64, 53] fue ampliamente desarrollado en años pasados debido a su simplicidad, pero su principal problema es que carece de la capacidad de asignar un puntaje a cada documento y ordenar los documentos usando este puntaje. Según [66], actualmente la mayoría de los modelos de recuperación tienen la capacidad de ordenar los documentos por puntaje (ranking), donde los tres modelos más usados en Sistemas de Recuperación de Información sobre texto son el modelo vectorial [62], el modelo probabilístico [48, 55], y el modelo de red de inferencia [70].

Estos modelos fueron adaptados para crear algoritmos de búsqueda en la Web, pero actualmente predominan los modelos que usan la estructura de links entre páginas para búsqueda en la Web. Sin embargo en el ámbito de los EMSE el modelo vectorial basta para generar buenos resultados, ya que el espacio de búsqueda es acotado y la información en las fichas de productos de distintas

tiendas, resulta ser más homogenea que la información que obtienen los buscadores orientados a toda la Web.

Modelo Vectorial

El modelo vectorial se basa en la representación de documentos y consultas de usuarios como vectores, donde cada dimensión del vector corresponde a una palabra del vocabulario. El vocabulario es el conjunto de palabras que forman parte de los documentos del espacio de búsqueda. El valor de cada elemento corresponde al peso de la palabra dentro del documento.

Para obtener un puntaje númerico de cada documento dada una consulta, se calcula la similitud entre los vectores. La función de similitud no es propia del modelo, pero generalmente se usa el ángulo que forman los vectores como concepto de distancia, y el coseno de dicho ángulo como función de similitud. La función de similitud entre el documento d y una consulta q está dada por

$$sim(d,q) = \frac{\vec{d} \bullet \vec{q}}{|\vec{d}| \times |\vec{q}|}$$

Donde $\vec{d}$ es el vector que respresenta al documento d, $\vec{q}$ es el vector que respresenta la consulta q, $\bullet$ es la función producto punto vectorial, y $|\vec{d}| \times |\vec{q}|$ es el factor de normalización.

Existen varios esquemas para el cálculo del peso de una palabra en un documento, y la elección de qué técnica usar depende de la funcionalidad del Sistema. Sin embargo la técnica más usada es *tf-idf* [63], la que usa frecuencia de términos para obtener el peso de cada término.

Tf (term frequency) es la frecuencia de un término (generalmente normalizada) dentro del documento, sirve para expresar la importancia del término en el documento. La frecuencia normalizada para una palabra t en un documento d, está dada por

$$tf_{d,t} = \frac{freq_{d,t}}{max_i\, freq_{d,i}}$$

Donde $freq_{d,t}$ es la frecuencia de la palabra t en el documento d, y el máximo es calculado sobre todas las palabras que forman el documento d.

Idf (inverse document frequency) indica que tan frecuente es el término dentro de la colección de documentos. La idea es que mientras más aparezca un término en la colección, menos sirve para distinguir qué documento es relevante. La frecuencia inversa para una palabra *t* esta dada por:

$$idf_t = log\frac{N}{f_t}$$

Donde N es el número total de documentos en el espacio de búsqueda, y f_t es la cantidad de documentos en los que aparece la palabra t.

Con esto, el peso asociado a la palabra t para el documento d está dado por

$$w_{t,d} = tf_{d,t} \times idf_t = \frac{freq_{d,t}}{max_i\, freq_{d,i}} \times log\frac{N}{f_t}$$

2.1.3. Indexación, Ranking y Búsqueda

Los índices son estructuras de datos que permiten realizar búsqueda textual de manera más eficiente, en comparación con la búsqueda de ocurrencia de términos en cada documento de la colección. El índice más usado con el modelo vectorial es el índice invertido.

El índice invertido consiste en una lista palabras, en donde cada palabra está asociada a la lista de documentos en los que la palabra aparece. Cada documento en la lista puede estar asociado a un valor que indique el peso de la palabra dentro del documento, el lugar del documento donde aparece, etc. En general se usa el peso de la palabra dentro del documento para determinar el orden en que se retornarán los documentos que satisfagan la consulta.

La mayoría de los Buscadores usa variaciones del modelo vectorial y/o booleano para realizar operaciones de ranking. Este ranking es realizado usando el índice, ya que al igual que la búsqueda se debe relizar sin acceder directamente a los documentos.

Por otro lado, existen rankings que usan la estructura de links entre las páginas. Dentro de los más conocidos se encuentran PageRank, cuya primera versión fue presentada en [21], y HITS [50].

Para el caso de los EMSE, el modelo vectorial cumple con las expectativas de ranking debido a lo acotado del vocabulario y del ámbito en general. Además, la estructura de links no es relevante en la información de los productos.

La búsqueda se realiza tomando como entrada la consulta realizada por el usuario. Dicha consulta es separada para obtener una lista de términos. Luego se buscan los documentos en donde se encuentran los términos de la lista y son retornados en forma de lista ordenada por ranking.

Para que la vista lógica de los documentos sea compatible con las consultas del usuario, se realizan operaciones de texto en ambos ámbitos. Las operaciones de texto más usadas son la eliminación de *stopwords*, lematización y uso de *thesauros*.

La eliminación de *stopwords* consiste en filtrar las palabras que no sirven para discriminar documentos al momento de la búsqueda. En general *stopwords* se asocian a artículos, preposiciones y conjunciones, pero dependiendo del espacio de búsqueda se pueden agregar otros tipos de palabras.

Lematización es el proceso de reducir las palabras a formas léxicas que son representativas. Esto produce que se reduzcan las variantes de palabras de igual significado. Ejemplos de variantes de palabras son plurales, gerundios y los distintos tiempos verbales.

Un *thesauro* es una colección de palabras relacionadas, donde cada palabra está asociada a un conjunto de palabras en su variación más común siguiendo una relación de sinónimos. La definición formal y otras aplicaciones sobre *thesauros* se describen en [37].

2.1.4. Navegación en Directorios

Los directorios son taxonomías donde se encuentran categorizados los documentos visibles por un buscador. El mejor y más antiguo directorio Web es Yahoo! [13].

Para que cada documento pertenezca a alguna categoría del directorio se debe realizar un proceso de clasificación. Pese a que se han realizado gran cantidad de estudios para clasificación

automática, ésta no se ha podido lograr con la confiabilidad deseada y la calsificación en grandes directorios sigue siendo obra de personas dedicadas a esa labor, ya sea como único medio de clasificación o como control de fallas de clasificadores automáticos.

Actualmente la gran mayoría de EMSE y buscadores en general, proveen la capacidad de navegar a través de un directorio. Además permiten realizar búsquedas acotadas a un sub-árbol de la taxonomía.

2.1.5. Clasificación

La clasificación consiste en asignar cada documento a un grupo tomando en cuenta sus características. La asignación se realiza basándose en un conjunto de documentos de entrenamiento, los cuales tienen un grupo ya asignado, generalmente mediante un proceso manual. Al necesitar un conjunto de entrenamiento y de categorías previamente determinados, el proceso de clasificación forma parte de los procesos supervisados.

Este proceso de clasificación es similar al descrito en el ámbito de Data Mining. La mayor diferencia es que en Data Mining los datos están estructurados en registros con pares atributo-valor, en cambio en clasificación textual los datos no están estructurados o están semi-estructurados. Naturalmente los documentos se pueden representar como vectores de palabras, lo que genera el problema de tener un número de atributos muy grande. En [36] se describen en detalle distintas técnicas de clasificación en Data Mining, así como el resto de técnicas usadas en esa área.

Actualmente los modelos de clasificación más usados son Support Vector Machines [68] (SVM), redes neuronales [78] y bayesianos [71]. SVM se presenta como el modelo de clasificación más poderoso y de aplicación directa en clasificación textual usando el modelo vectorial. En [46] y [34] se describen las ventajas que tiene SVM para clasificación y se presentan experimentos comparativos con otros modelos.

Support Vector Machines

La idea general bajo los procesos de Support Vector Machines (SVM) es la de separar los datos en dos conjuntos, donde los datos son representados como vectores de dimensión n. SVM busca generar un hiperplano cuya distancia hacia los bordes de ambos conjuntos generados, sea máxima.

Existen varios trabajos relacionados con SVM. En [47] se describe Transductive SupportVector Machine (TSVM), una extensión de SVM que permite mejorar la precisión cuando el conjunto de entrenamiento es pequeño. Una comparación de distintas técnicas para la clasificación en multiples clases usando SVM se puede encontrar en [44]. En [32] se estudia la eficacia de SVM para clasificar Spam y finalmente en [45] se propone es uso de SVM para predecir estructuras de objetos complejas.

2.2. Web Semántica

La noción de *Web Semántica* [19], promovida por Tim Berners-Lee [12], el creador de la Web, consiste en transformar la Web actual de tal manera que la información y servicios sean entendibles y usables tanto por computadores como por humanos. La Web semántica creará un ambiente donde agentes(software) puedan realizar tareas sofisticadas y ayudar a las personas a encontrar, entender, integrar, y usar información y servicios.

La característica clave de la Web Semántica será un lenguaje estándar de marcado de metadatos y ontologías, lo que permitirá a los agentes encontrar el significado de la información en páginas web a través del seguimiento de hiper-vínculos que llevan a definiciones de términos claves y reglas de razonamiento lógico acerca de éstos términos.

En [29] se explica en detalle la visión original de Tim Berners-Lee, el concepto de *Ontología*, distintas aplicaciones en el ámbito de la industria, y las tecnologías más usadas para implementa-

ciones compatibles con la Web Semántica.

2.2.1. Ontología

Existen varias definiciones de Ontología en distintas áreas de la ciencia y la filosofía. Para el ámbito de la Web Semántica, una Ontología define las palabras y conceptos comunes usados para describir y representar un área de conocimiento. Esto implica la definición de objetos del dominio, las relaciones entre estos objetos, las propiedades, funciones y procesos que involucran estos objetos, y restricciones y reglas asociadas a estos objetos.

Generalmente la definición de una Ontología involucra los siguientes conceptos:

- **Clases**: Conjuntos, colecciones, o tipos de objetos.
- **Instancias**: Objetos.
- **Relaciones**: Representan la manera en que interactuan los objetos.
- **Propiedades**: Atributos que los objetos pueden tener y compartir.
- **Eventos**: Representan los cambios de propiedades o relaciones de los objetos.

Existen varios lenguajes que permiten describir estos conceptos. En primer lugar el lenguaje más utilizado para representar relaciones semánticas es RDF [52]. RDF es el lenguaje propuesto por el W3C [73] para describir metadatos, y se basa en la idea de generar sentencias acerca de recursos, en la forma sujeto-predicado-objeto. El sujeto corresponde a un recurso, el predicado expresa una relación entre el sujeto y el objeto, y el objeto corresponde a otro recurso relacionado con el sujeto o un valor literal.

RDF Schema(RDFS) es una extensión de RDF que permite describir Ontologías con funcionalidades básicas. El mayor aporte de RDFS [20] es la capacidad de representar relaciones de nivel

entre clases, con lo cual se pueden definir jerarquías.

Finalmente el lenguaje más expresivo para la representación de conocimiento para la Web Semántica es OWL [72]. OWL (Ontology Web Language) es un lenguaje para describir ontologías desarrollado por la W3C, y provee la capacidad de expresar todos los conceptos relacionados con una Ontología: Clases, Instancias, Relaciones, Propiedades, y Eventos.

2.2.2. Búsqueda Semántica

La Búsqueda Semántica [42] consiste en aumentar la calidad de los resultados obtenidos por la búsqueda tradicional, usando la información *entendible* que provee la Web Semántica. En [49] se clasifican los distintos tipos de Búsqueda Semantica con ejemplos de Sistemas para cada tipo. Estos Sistemas han sido desarrollados para efectos de investigación en su mayoría. La clasificación detallada indica que existen cinco tipos:

- **Meta-Buscadores Sobre Ontologías**: Consiste en aplicaciones que realizan búsqueda de Documentos de la Web Semántica, especialmente ontologías. OntoSearch [79] es un ejemplo de este tipo de buscadores.

- **Buscadores Sobre Ontologías Basado en Crawler**: Similar al anterior, pero se realizan procesos de crawler e indexación específicos para Documentos de Web Semántica. En este tipo de buscadores destaca Swoogle [31] como precursor y Sindice [69][59] como el de mayor funcionalidad en la actualidad. Otros buscadores de uso general son Hakia [8] y Sensebot [11].

- **Buscadores Basados en Contexto**: Consiste en aplicaciones de búsqueda que aumentan la presición de la búsqueda tradicional a través del entendimiento del contexto de la consulta y los documentos. Esto generalmente se raliza consultando grafos RDF después de realizar

una búsqueda tradicional. Dentro de este tipo de buscadores se encuentra la mayor parte de los trabajos en búsqueda semántica, destacando OntoWeb [67] y Corese [28].

- **Buscadores Evolutivos**: Similar al anterior pero enfocado a un tópico específico. El Sistema TAP [42] pertenece a este tipo de buscadores.

- **Descubridores de Relaciones Semánticas**: Consiste en encontrar relaciones semánticas entre terminos de entrada (generalmente dos) y luego ordenar los resultados basándose en medidas de distancias semánticas. El trabajo realizado en [15] describe un buscador de este tipo.

2.3. E-commerce Metasearch Engines

Los E-commerce Metasearch Engines (EMSEs) son meta buscadores que concentran productos de diferentes sitios de comercio electrónico, otorgándole al usuario una interfaz única de consulta y mostrando como resultado productos de diferentes tiendas en un listado unificado. Todo esto con el objetivo básico de comparar precios y ayudar al usuario en la elección de su compra.

2.3.1. Funcionamiento

Las técnicas usadas por estos buscadores no son de domino público debido al origen comercial de su implementación, pero la mayoría de sus procesos son equivalentes a otros sistemas de recuperación de información.

La obtención de los datos de productos que presentan estos buscadores se realiza principalmente de tres formas:

- **Proceso Manual**: Consiste en el ingreso o actualización de información a través de personal

humano asignados a dichas tareas y que pertenece al equipo que mantiene el meta buscador.

- **Proceso Asistido por Terceros**: Consiste en el envío de información por parte de las empresas con tiendas virtuales. Generalmente esto se realiza mediante formularios HTML provistos por el mismo buscador, Web Services, o transferencias de archivos.

- **Proceso Automático**: Es el proceso de obtención y actualización de información a través de programas que recorren los sitios de comercio electrónico y detectan información relevante para el meta buscador.

En general la búsqueda implementada por estos sitios está basada en recuperación de información clásica y el uso del modelo vectorial, explicado en el punto 2.1.2. El método mas usado por estos sitios para comparar, es ordenar por algún atributo que forme parte de la data estructurada por cada meta buscador. El principal atributo -y el mas usado- es el precio. Las principales deficiencias en estos sitios son:

- Estructuración de la información mediante ingreso de información usando formularios o manual. Al usar formularios la información ingresa estructurada al Sistema, siguiendo la pauta dada por el formulario. Pero al ser un proceso manual, existe la probabilidad de que la información no sea consistente con lo que realmente presentan las tiendas en Internet.

- Pocas funcionalidades de comparación, básicamente por precio o algún otro atributo.

- Motores de recomendación usan estadísticas de opiniones de usuario ingresadas en el Sistema. No existen recomendaciones basadas en la información de los productos.

2.3.2. Principales Buscadores

Existen variados meta buscadores desarrollados en distintas partes del mundo, entre los mas reconocidos están PriceGrabber [9] - USA, Ciao [4] - Europa, Buscape [3] - Latinoamérica. Estos

tres buscadores comparten varias características, donde las principales son que presentan resultados de gran cantidad de tiendas, facilidad de ordenar por precio, e identificación de productos iguales. Esto ultimo se hace básicamente identificando de forma manual o a través del ingreso de información por formularios HTML, en donde la marca y el modelo son seleccionables a partir de una lista pre-establecida.

También existen diversos trabajos de integración automática de buscadores, dentro de los que destaca WISE [43], que presenta integración automática de los formularios de búsqueda de los sitios de comercio electrónico.

2.3.3. Técnicas de Comparación

Los EMSEs exponen gran cantidad de productos al usuario, lo que puede transformarse en un gran problema cuando el número de alternativas crece mas allá de las capacidades cognitivas de los clientes [77]. Existen varios estudios que muestran la necesiad de aumentar el esfuerzo cognitivo en la tarea de comparación de productos, cuando la cantidad de alternativas crece en un ambiente de compras en línea [65, 22].

En [76] se clasifican los distintos tipos de software de comparación de precios, usando como referencia los principales sistemas de venta en línea. Los tipos de software de comparación en general son tres:

- **Diferenciación**: Proporcionan información de los diversos atributos de diferenciación, y para cada atributo muestra alternativas de elección. Estos atributos se determinan estructurando la información obtenida desde distintos sitios, ya sea por medio de convenios con las tiendas o por procesos propios del EMSE que implementa la comparación. Un ejemplo claro de este tipo de sistemas es PriceScan [10]

- **Evaluación**: Basado en información cuantitativa provista por los usuarios en distintas di-

mensiones (variables) según el producto. El mejor representante de este tipo de sistemas es BizRate [2], el cual recolecta experiencias de usuarios desde varias tiendas de comercio electrónico y lo normaliza en una escala de ratings para poder realizar la comparación.

- **Identificación de Preferencias**: Basado en la recolección de experiencias de los usuarios con los distintos productos que forman parte del Sistema. El sistema más emblemático de este tipo es Epinion [6] que es básicamente un sistema dedicado a la recolección de opiniones para los productos.

Los buscadores que implementan comparaciones del primer tipo usan gran cantidad de esfuerzo humano para estructurar información necesaria para determinar atributos y opciones para cada tipo de producto. Los últimos dos tipos mencionados requieren de la interacción del usuario, por lo que no pueden recomendar productos desde la puesta en marcha del Sistema.

En general los buscadores de la actualidad mezclan al menos dos de las técnicas descritas anteriormente. Es así como PriceGrabber, Buscapé y Ciao utilizan *diferenciación* en forma implícita (agrupando en subcategorías) en vez de crear filtros de búsqueda por variables, y por otro lado usan *identificación de preferencias* como apoyo a la toma de decisión por parte del usuario.

2.3.4. Problemas y Posibles Mejoras

Los principales problemas de los actuales buscadores tienen que ver con la automatización de los procesos, y la dependencia de interacción con el usuario para recomendar o identificar productos similares. La gran mayoría de los EMSE nombrados en el punto anterior proveen formularios a las empresas de comercio electrónico para actualizar e ingresar información al Sistema. El problema de esto es que se genera una dependencia en otros Sistemas o personas, lo que puede llavar a no tener información actualizada, o peor aún, información manejada por las tiendas. El precio es la información más importante de cada producto y es escencial que dicha información sea fidedigna

y lo más actualizada posible.

En cuanto a los sistemas de recomendación de los EMSE, la tendencia es generar relaciones a través de la información generada por los usuarios al navegar por este tipo de sitios. Las secciones del estilo “Los compradores que compran este producto también compran...” son una muestra de esta técnica. Otra forma usada es estructurar la información desde que es ingresada por las empresas de comercio electrónico, de tal manera que el producto ingrese a la Base de Datos con todas sus caracteísticas estructuradas en la forma que necesita el EMSE para clasificar y generar pantallas comparativas. En definitiva no se aprovecha la información del producto en sí para encontrar similitud con otro producto.

2.3.5. Tecnologías de Web Semántica para EMSEs

El principal aporte de las tecnologías de la Web Semántica en el ámbito del comercio electrónico, es la capacidad de modelar el dominio de esta industria en una ontología, lo que permite integrar nueva información (integración de ontologías), estructurar la información en clases y relaciones consistentes, y flexibilidad al momento de incrementar dichas clases y relaciones. Esto es soportado a través de la descripción de dichas ontologías a través de lenguajes de descripción como RDF [52], RDFS [20], y OWL [72].

Capítulo 3

Objetivos

Este capítulo presenta los objetivos planteados para el desarrollo de este trabajo.

3.1. Objetivo General

Desarrollar un buscador de productos de venta en línea, orientado a las tiendas de comercio electrónico en Chile, que identifique productos iguales y similares automáticamente, y que despliegue dicha información de tal manera que sea fácil de entender para el usuario final.

3.2. Objetivos Específicos

Los objetivos específicos de este trabajo son:

- Desarrollar estrategias para recolección de información desde las tiendas
- Estructurar la información obtenida desde las distintas tiendas y guardarlas en un formato único

- Generar un árbol de tópicos para búsqueda por navegación. Clasificar cada producto en algún tópico de dicho árbol
- Desarrollar e implementar estrategias para comparación de productos, de tal manera de identificar productos iguales y productos similares.
- Implementar una aplicación Web que provea una interfaz para búsqueda de productos por keyword, y navegación por categorías.

Capítulo 4

YOSH

Como parte del trabajo realizado en la presente tesis se desarrolló un buscador de productos de comercio electrónico denominado YOSH, el cual recibió ese nombre en honor una mascota fallecida llamada Yoyi. YOSH es un meta buscador de comercio electrónico (EMSE), cuya informaciíon se estructura en una ontología, y que posee la partidularidad de detectar y recomendar productos similares basándose en medidas de distancia entre productos.

Este capítuo describe el diseño e implementación del buscador de comercio electrónico realizado en esta tesis. Primero se describe la arquitectura general del Sistema. Luego se detallan uno a uno los componentes desarrollados y las técnicas utilizadas.

4.1. Arquitectura General del Sistema

La aquitectura utilizada es una adaptación de la clásica crawler-indexer, agregando una Ontología y Motor de Recomendación para generación, almacenamiento y obtención de relaciones entre productos. El Sistema está compuesto por siete componentes:

- Crawler: Es el encargado de recorrer los sitios de comercio electrónico y detectar información interesante para el ámbito del buscador.
- Extractor de Información: Estructurar los datos obtenidos por el Crawler según la ontología asociada al buscador.
- Clasificador de Productos: Determina a que categoría pertenece un producto.
- Motor de Recomendación: Realiza los procesos de cálculo de medidas de distancia entre productos y graba dichas relaciones en la Base de Datos
- Ontología: Es el modelo conceptual sobre el cual trabaja el buscador.
- Indexador: Realiza el proceso de generación de indices para la búsqueda basada en *keywords*.
- Aplicación de consultas: Es la interfaz WEB que permite al usuario realizar consultas, navegar por las categorías, y comparar los distintos productos.
- Lucene: Es un framework para búsqueda de texto desarrollado por Apache Foundation [38], esta basado en generación de índices para búsqueda eficiente usando el concepto de documentos que contienen campos. Lucene provee una API para agregar documentos al índice y realizar búsquedas sobre éste.

La figura 4.1 muestra la arquitectura general del Sistema y la interacción entre sus componentes. En los siguientes puntos de este capítulo se describen en detalle las componentes del Sistema.

4.2. Ontología

Una ontología es la especificación formal de términos y sus relaciones dentro de un área de interés [41]. La Ontología diseñada pretende ser la estructura para la Base de Conocimientos que almacenará la información de comercio electrónico usada por el buscador.

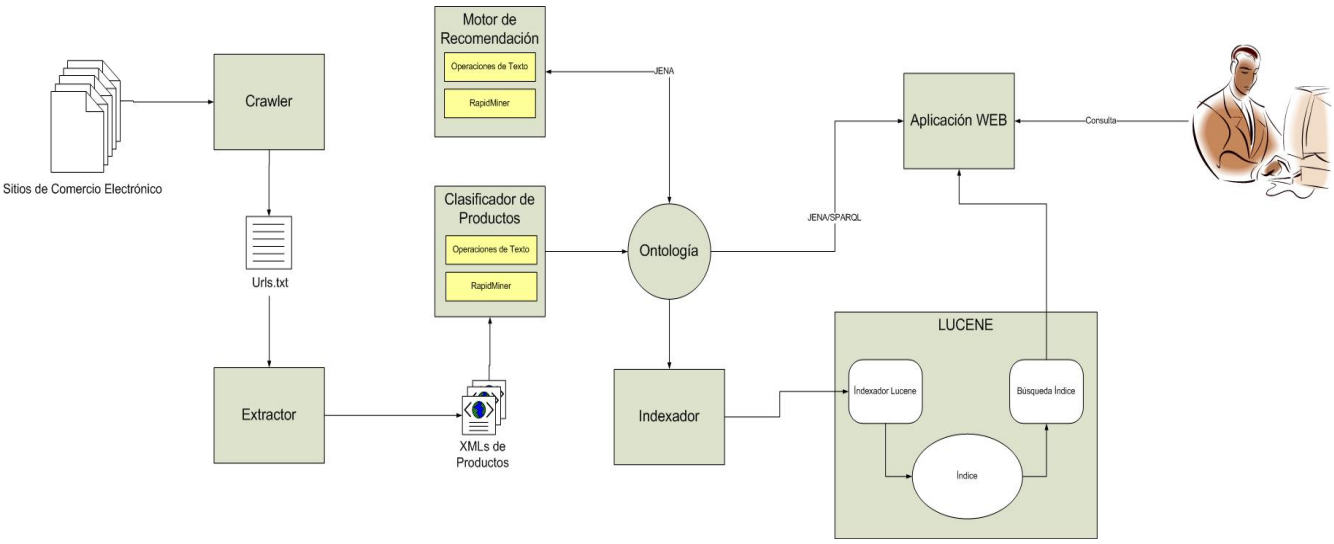

Figura 4.1: Arquitectura general del YOSH

Para la creación de la Ontología se adoptó como objetivo principal de diseño, la posibilidad de extenderla en el futuro y que su uso esté orientado una aplicación de búsqueda. Es por esto último que se decidió la generación de una nueva Ontología, ya que las existentes para el ámbito del comercio están orientadas a la integración de sistemas B2B o al comercio en tiendas físicas, ejemplos son PLIB [14] y la extención para comercio de SHOE [54] respectivamente.

La figura 4.2 muestra la Ontología resultante para este trabajo, contiene sólo las clases estrictamente necesarias para el ámbito del comercio electrónico. Las clases que componen la Ontología se describen a continuación

- **Organización Comercial**: Corresponde al concepto de entidades de giro comercial. Tiene como sub-clases a Productor y Proveedor
- **Productor**: Corresponde a una Organización Comercial encargada de producir productos que se venden en las tiendas. Se puede entender también como la *marca* de un producto.
- **Proveedor**: Corresponde a una Organización Comercial encargada de vender productos. Se puede entender también como *tienda*.

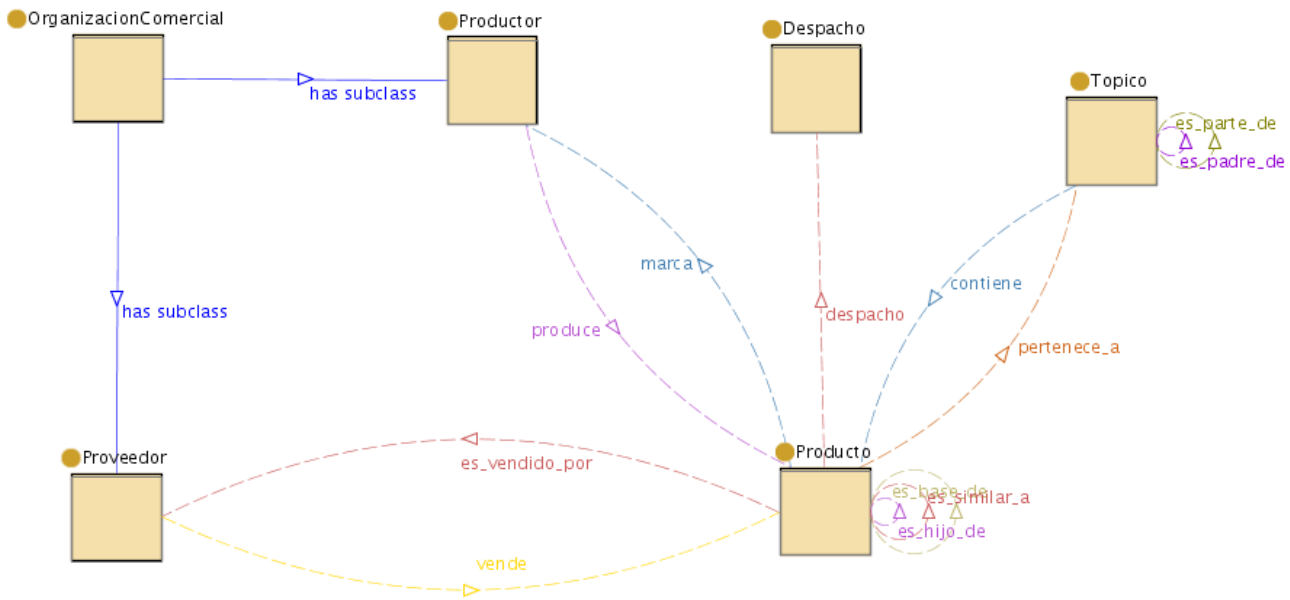

Figura 4.2: Ontología para comercio electrónico

- **Producto**: Representa cualquier objeto o servicio que pueda adquirir un usuario de un sitio de comercio electrónico.
- **Topico**: Corresponde a una categoría para organizar el universo de productos. Cada tópico es una agrupación de productos y/o una agrupación de otros tópicos. Las instancias de esta clase forman una taxonomía.
- **Despacho**: Representa la información de despacho de un producto

La Ontología se completa con propiedades que relacionan las clases. Las relaciones más importantes son:

- **es_vendido_por**: Relaciona un Producto con el Proveedor (tienda) en el cual se vende.

- **vende**: Relaciona un Proveedor con los Productos que vende.
- **marca**: Relaciona un Producto con la Organización Comercial que lo produce (Productor).
- **produce**: Relaciona un Productor con los Productos que fabrica.
- **despacho**: Propiedad que asocia los datos de despacho con un Producto.
- **pertenece_a**: Relaciona un Producto con la agrupación de productos al cual corresponde.
- **contiene**: Relaciona un Topico con los Productos que forman parte del concepto de agrupación representado por el Topico.
- **es_similar_a**: Relaciona un Producto con otro Producto de similares características.
- **es_base_de**: Relaciona una instancia de la clase Producto, con la instancia de tipo Producto que corresponde a la representación base del Producto (generalización). Esta relación se da sólo cuando ambos productos son iguales (misma marca y modelo). Por ejemplo si existen 3 instancias de la clase Producto que representan al producto real "Notebook Dell 11z" (una por cada tienda: ripley, falabella, paris), el motor de recomendación elige el primero que se procesa como "Producto Base" y se relaciona con los otros 2 usando la propiedad *es_base_de*.
- **es_hijo_de**: Relaciona la representación base de un Producto con los productos vendidos en la tienda (especialización).
- **es_parte_de**: Relaciona un Topico con un Topico que representa un concepto más general.
- **es_padre_de**: Relaciona un Topico con un Topico que representa una especialización del concepto asociado al primer Topico.

La Ontología se completa con propiedades cuyo valor corresponde a un Literal, y por lo tanto sólo permiten guardar información. Para la formalización de la Ontología se usó OWL ya que

para facilitar el uso de la Ontología en la aplicación se usaron propiedades transitivas e inversas (es_parte_de, es_padre_de), las cuales no son naturales en el lenguaje RDFS.

4.3. Crawling

Debido a las diversas implementaciones de los sitios de comercio electrónico, y al amplio uso de técnicas de generación dinámica de links (usando javascript), fue necesario hacer programas de crawler ad-hoc para cada tienda de comercio electrónico, pero dentro de un framework de crawling común que permite que para agregar una nueva tienda baste con identificar cuales son los patrones de URL que identifican una página de categoría y una de producto. Dichos patrones se deben incorporar al programa en una clase que debe mantener el contrato de la clase VisitaUrl que se especifica más adelante.

4.3.1. Modelo

La estrategia básica es recorrer el sitio a procesar, a través del árbol de categorías que presenta cada sitio, e identificar las URLs que corresponden a las páginas de productos. Dichas URLs se guardan para ser procesadas por el módulo de extracción de información.

Para estos efectos se desarrolló un programa que recibe como parámetro la URL de la página de inicio de la tienda que se desea recorrer (ej: `http://www.ripley.cl/webapp/wcs/ stores/servlet/StoreCa`
y como resultado entrega un archivo con todas las URLs de productos identificadas.

El programa analiza el contenido de la página y obtiene los links que correspondan a una página de categorías o de productos. Los links correspondientes a productos son inmediatamente grabados en el archivo resultante del proceso, y los links de categorías son usados para seguir recorriendo el sitio usando una estrategia *breadth-first*.

Por simplicidad se eliminó todo contenido javascript de las páginas para *ripley.cl* y *paris.cl*, por lo tanto no se consideran links generados dinámicamente mediante funciones javascript. En el caso particular de *falabella.cl*, todos los links de categorías y productos son generados dinámicamente usando una función javascript, por lo que fue necesario armar las URLs correspondientes usando los parámetros de las funciones invocadas en cada link.

4.3.2. Implementación

Para cumplir con el modelo descrito anteriormente, el programa implementado usa un pool de threads (para mayor eficiencia) donde cada thread se encarga de analizar una URL. Además permite agregar distintos comportamientos a través de la instanciación dinámica (reflexión) de la clase que implementa el proceso de crawler para cada sitio. Para estos efectos se uso el patrón de diseño *prototyping*. En Fig.4.3 se presenta el diagrama de clases del programa y a continuación se explica brevemente cada clase involucrada.

- **org.abb.crawler.Crawler**: Es la clase que inicia la ejecución, encargada de leer el archivo de configuración, inicializar el pool de threads, inicializar el proceso de crawler, y finalmente escribir el archivo con el resultado de la operación.

- **org.abb.crawler.CrawlerControl**: Clase encargada de guardar los resultados de las operaciones realizadas por cada thread. El acceso a los atributos de esta clase es concurrente, por lo que los métodos de obtención y establecimiento para los atributos se declaran sincronizados.

- **org.abb.crawler.ThreadPool**: Contiene el arreglo de Threads utilizados para el proceso de crawler. Además se encarga de eliminar threads inactivos y levantar nuevos threads en caso que existan URLs sin visitar encoladas.

- **org.abb.crawler.VisitaUrl**: Es la implementación base para el proceso de visita y parser de una URL.

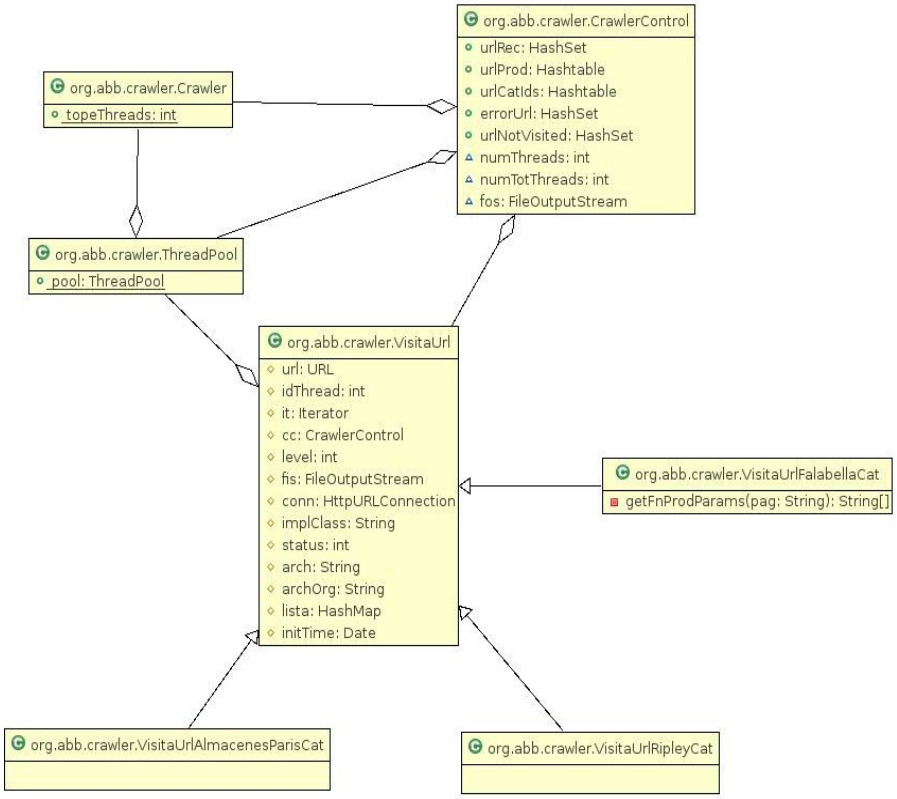

Figura 4.3: Diagrama de clases del Crawler

- **org.abb.crawler.VisitaUrlAlmacenesParis**: Implementación específica para *paris.cl*.
- **org.abb.crawler.VisitaUrlRipley**: Implementación específica para *ripley.cl*.
- **org.abb.crawler.VisitaUrlFalabella**: Implementación específica para *falabella.cl*.

4.3.3. Medición

Para este tipo de crawlers se identifican claramente dos tipos de mediciones con distintos objetivos. La primera consiste en la evaluación del número de threads óptimo para realizar los procesos

de crawler. El segundo corresponde al número de productos que fue capaz de recolectar para cada tienda.

Para la primera medición se realizó un experimento que consiste en ejecutar el crawler sobre los sitios paris.cl, ripley.cl y falabella.cl, variando el número de threads en cada ejecución. Los valores para el número de threads ocupados fueron 5, 10, 25, 50, 100.

En la Tabla 4.1 se detallan los resultados del experimento, los que se ven reflejados gráficamente en la Figura 4.4. De estos resultados se puede concluir que el comportamiento del crawler no depende de la complejidad del sitio ni de la cantidad de productos o páginas que recorra, ya que en los tres sitios el mejor resultado (en cuanto a tiempo) se consiguió con 25 threads. Esto se debe a que al aumentar el número de threads también se aumenta el número de conexiones HTTP, por lo tanto llega un momento en que la cantidad de conexiones es demasiada para la red local y se produce congestión, término de conexiones y timeouts.

En cuanto al número de ítems recolectado por tienda, podemos concluir que si bien el aumento del número de threads afecta el rendimiento del proceso, la merma es menor al usar 25 threads. El caso de *paris.cl* con 100 threads presentó una merma importante debido a la congestión de la red producida por la concurrencia de accesos HTTP al sitio.

4.4. Extracción de Información

El proceso de extracción de información consiste en recorrer las URLs resultantes del proceso de Crawling, obtener la información relevante del producto mediante un proceso de parser realizado sobre cada página recorrida, guardar la información de cada producto como un documento XML, y finalmente identificar que acciones se deben realizar con cada producto en la siguiente etapa (actualizar, despublicar y clasificar).

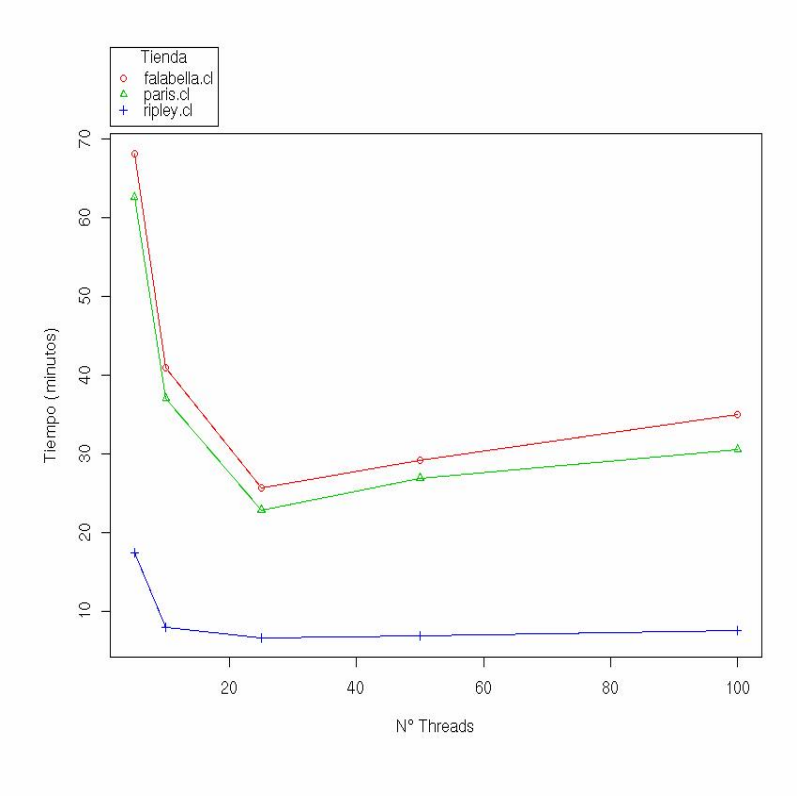

Figura 4.4: Gráfico Tiempo v/s Threads agrupado por tienda

4.4.1. Modelo

En general los sitios de comercio electrónico despliegan la información de los productos mediante una plantilla gráfica compartida para todos los productos (template de productos). El uso de plantillas se aprovecha para este proceso, ya que todos los datos a extraer ocupan la misma zona de la página sin importar el producto que se muestre.

Con esto, se puede manejar cada página de producto como un XML, y cada zona que sea necesario identificar (precio lista, precio internet, características) se puede acceder a través de expresiones XPATH [75] que dependen de la plantilla que usa cada tienda.

Las acciones a realizar en los siguientes procesos para cada producto están determindas por varios factores, principalmente por la actualización de los datos o la ausencia de un producto en

Tienda	Threads	Tiempo(s)	# Páginas
ripley.cl	5	17,38	3322
ripley.cl	10	7,95	3322
ripley.cl	25	6,6	3322
ripley.cl	50	6,88	3322
ripley.cl	100	7,45	3322
paris.cl	5	62,55	15223
paris.cl	10	37,02	15220
paris.cl	25	22,8	15219
paris.cl	50	26,88	15015
paris.cl	100	30,5	13105
falabella.cl	5	68,12	17115
falabella.cl	10	40,88	17113
falabella.cl	25	25,63	17113
falabella.cl	50	29,14	17113
falabella.cl	100	34,91	17110

Tabla 4.1: Resultados ejecución Crawler con distintos números de threads

alguna tienda. En la Figura 4.5 se muestra el diagrama de actividades del proceso de extracción de información, donde se describe el flujo que sigue cada URL obtenida en el proceso de crawler.

4.4.2. Implementación

Para estos efectos se desarrolló un programa que recibe como parámetro un archivo de texto con la lista de URLs a procesar, la tienda, y un set de sentencias XPATH. Como resultado entrega un conjunto de archivos XML, donde cada archivo contiene la información estructurada de un producto. En la Tabla 4.2 se muestra el XML Schema [74] de los XML generados. En caso que el mismo producto aparezca más de una vez (con distintas URLs) en el archivo de entrada, se agregan elementos en la lista de *topicos* y *url* según el XML Schema. Cada archivo lleva como parte de su nombre la tienda y el codigo de producto (de la tienda).

El programa recorre el archivo de entrada y por cada URL realiza el siguiente procedimiento:

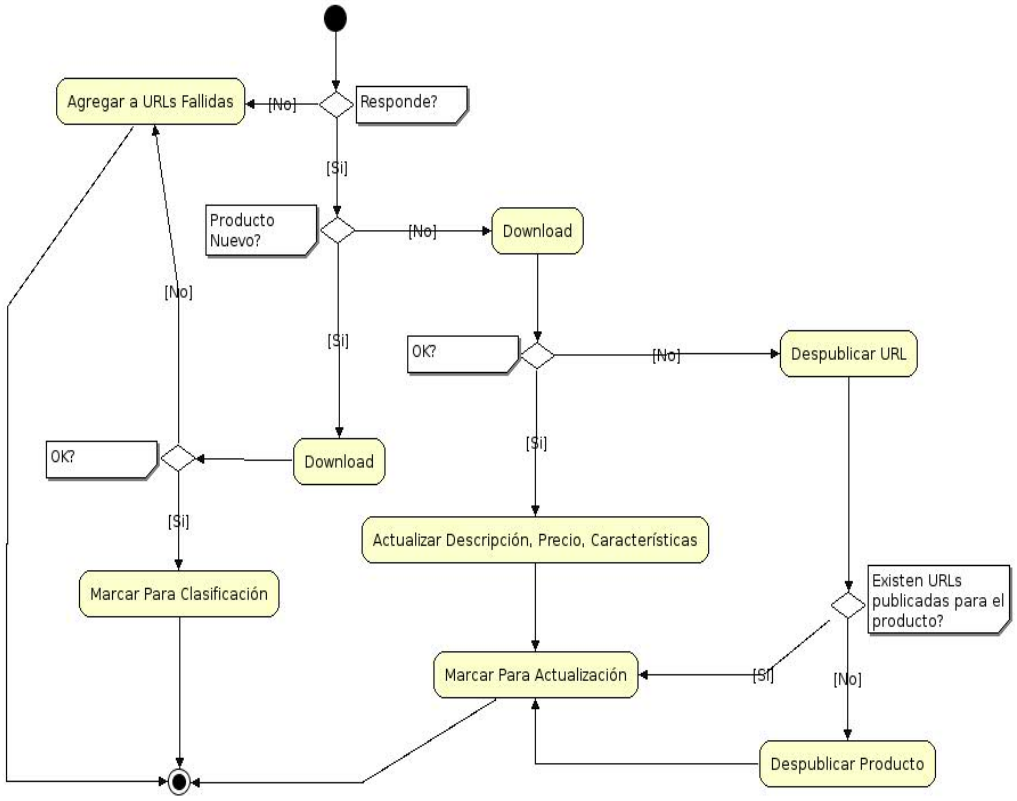

Figura 4.5: Diagrama de actividades del Extractor de Información

- Obtiene el contenido desde el servidor.
- Limpia el contenido y convierte el documento en XHTML, mediante jTidy [56].
- Reliza una transformación XSLT sobre el documento XHTML. El XSL se forma dinámicamente usando las sentencias XPATH. El resultado es un XML de producto que cumple con el schema descrito en 4.2.
- Genera o actualiza el archivo correspondiente al producto.

Además permite agregar distintos comportamientos a través de la instanciación dinámica (reflexión) de la clase que implementa el proceso de crawler para cada sitio. Para estos efectos se

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="producto">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="descripcion"/>
        <xs:element ref="precioLista"/>
        <xs:element ref="precioInternet"/>
        <xs:element ref="caracteristicas"/>
        <xs:element ref="urls"/>
        <xs:element ref="topicos"/>
        <xs:element ref="tienda"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="descripcion" type="xs:string"/>
  <xs:element name="precioLista">
    <xs:complexType/>
  </xs:element>
  <xs:element name="precioInternet">
    <xs:complexType/>
  </xs:element>
  <xs:element name="caracteristicas">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="caract"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="caract" type="xs:string"/>
  <xs:element name="urls">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="url"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="url" type="xs:anyURI"/>
  <xs:element name="topicos">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="topico"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="topico" type="xs:string"/>
  <xs:element name="tienda" type="xs:integer"/>
</xs:schema>
```

Tabla 4.2: XML Schema de archivo de productos

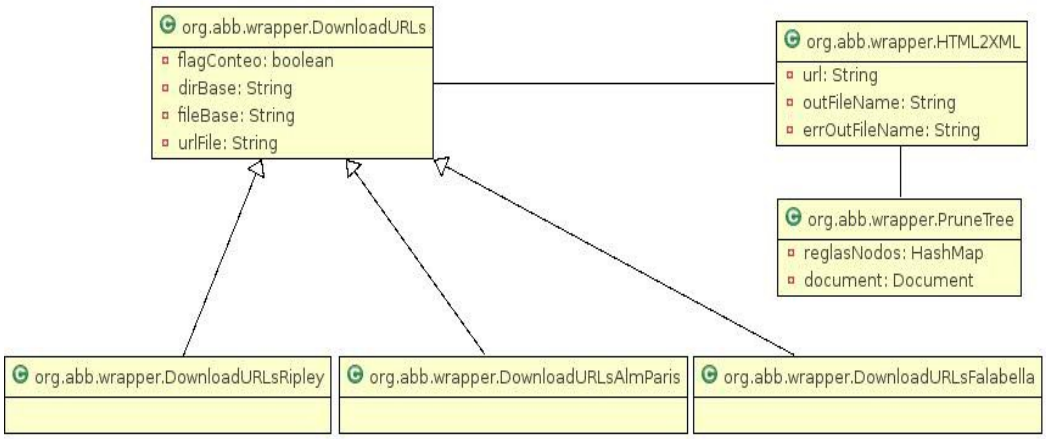

Figura 4.6: Diagrama de Clases Extractor de Información

usó el patrón de diseño *prototyping* . En Figura 4.6 se presenta el diagrama de clases del programa y a continuación se explica brevemente cada clase involucrada.

- **org.abb.wrapper.DownloadURLs**: Es la clase que inicia la ejecución, encargada de leer los archivos de configuración y entrada. Además contiene los métodos con la implementación base para el proceso de cada URL.
- **org.abb.wrapper.HTML2XML**: Clase encargada transformar la información de una página HTML en un documento XHTML válido.
- **org.abb.wrapper.PruneTree**: Clase encargada de eliminar secciones específicas del documento XHTML, por ejemplo las secciones *javascript*.
- **org.abb.wrapper.DownloadURLsAlmParis**: Implementación específica para *paris.cl*.
- **org.abb.wrapper.DownloadURLsRipley**: Implementación específica para *ripley.cl*.
- **org.abb.wrapper.DownloadURLsFalabella**: Implementación específica para *falabella.cl*.

4.4.3. Medición

Para medir el funcionamiento de éste módulo se debe tomar en cuenta la cantidad de URLs correctamente procesadas versus las procesadas sin éxito. Para esto se ejecutó el proceso de extracción para las URLs obtenidas desde paris.cl, ripley.cl y falabella.cl.

En la Tabla 4.3 se detallan los resultados del experimento. De estos resultados se puede concluir que el uso de XSL para estructurar la información tiene buenos resultados, en el caso de paris.cl y falabella.cl los errores se debieron a problemas de conexión por lo que el template en ese sitio se mantiene para todos los productos. En el caso de ripley.cl, el mayor porcentaje de fallas se debe a que tiene varios tipos de templates según el tipo de producto, por ejemplo para los productos de tipo combo (Ej:1 mesa + 6 sillas) tienen un despliegue totalmente distinto, por lo que es necesario hacer un XSL de mayor complejidad para abarcar todos los casos.

Tienda	Tiempo(min)	Url Total	Url OK	Url Error	Productos	% Fallas
ripley.cl	47,02	3790	3771	19	2393	0,5
paris.cl	129,08	14579	14531	48	4183	0,3
falabella.cl	149,37	17113	17107	6	6700	0,03

Tabla 4.3: Resultados ejecución del módulo de extracción de información

4.5. Clasificación de Productos

Esta etapa consiste en la creación de un árbol de categorías (taxonomía de productos) y la clasificación de cada producto obtenido desde las tiendas en alguna de las categorías del árbol. Para este propósito se usan dos procesos independientes que se describen a continuación.

4.5.1. Pre-procesamiento de Productos y Creación de Taxonomía

Éste módulo tiene como objetivo crear el árbol de categorías (taxonomía) para el buscador, y preparar la información para realizar las tareas de clasificación de productos sobre la taxonomía. Todo esto a partir de los archivos XML obtenidos en el proceso descrito en el punto 4.4.

La estrategia usada para la generación de la taxonomía, es tomar como base el árbol de categorías de alguna de las tiendas y filtrar las categorías que se refieren a conceptos específicos de dicha tienda, por ejemplo *K-sa digital* en *ripley.cl*. De esta manera se normaliza el conjunto de entrenamiento para que un producto quede asociado sólo a una categoría. Las categorías corresponden al último nivel del árbol(hojas).

En este módulo también se ejecuta la tarea de detectar la marca y modelo de cada produto, lo que se realiza a través de una expresión regular que depende de la tienda, aprovechando que la marca y el modelo siempre se encuentra en la misma posición de la cadena de caracteres que forman la descripción del producto. La expresión regular usa una lista de marcas conocidas como base.

Se desarrolló un programa que recibe como parámetro un archivo de configuración, el cual especifica los archivos de salida para datos de entrenamiento y datos para clasificación. Por cada tienda se especifica el directorio que contiene los archivos XML con la información de productos, una expresión regular para identificar marca y modelo, y en caso que la tienda sea marcada como principal, se especifica una lista de nombres de categorías que no se deben considerar al momento de generar la taxonomía.

El programa procesa en primer lugar la tienda marcada como principal en el archivo de configuración. Por cada archivo XML de la tienda realiza el siguiente procedimiento:

- Procesa el XML mediante un handler SAX, guardando la información en un objeto.

- Obtiene marca y modelo, y agrega esta información al objeto.
- Agrega el producto a la Ontología a través de JENA [57].
- Obtiene la referencia a la categoría hoja que le corresponde en la taxonomía. Esto se logra obteniendo la lista de topicos desde el XML, eliminando los que no correspondan según los filtros, y generando dinámicamente un árbol de categorías, agregando cada nueva categoría a la Ontología.
- Agrega el objeto a la lista de productos a procesar como conjunto de entrenamiento.

Luego se procesa el resto de las tiendas (no marcadas como principales). Por cada archivo XML de la tienda realiza el siguiente procedimiento:

- Procesa el XML mediante un handler SAX, guardando la información en un objeto.
- Obtiene marca y modelo, y agrega esta información al objeto.
- Agrega el producto a la Ontología a través de JENA [57].
- Agrega el objeto a la lista de productos a procesar para clasificación.

Finalmente se recorren las listas de productos para generar los archivos de salida. Cada archivo contiene un vector de palabras por cada linea. Cada vector corresponde a un producto y contiene las palabras obtenidas de los elementos *descripcion* y *caracteristicas* del XML del producto. En el caso del archivo con datos de entrenamiento, antes del vector de palabras se especifica el ID de la categoría hoja a la que pertenece el producto.En la Figura 4.7 se presenta el diagrama de clases del programa y a continuación se explica brevemente cada clase involucrada.

- **org.abb.preprocess.ProcessData**: Es la clase que inicia la ejecución, encargada de leer el archivo de configuración, procesar los archivos XML, y generar los archivos de salida.

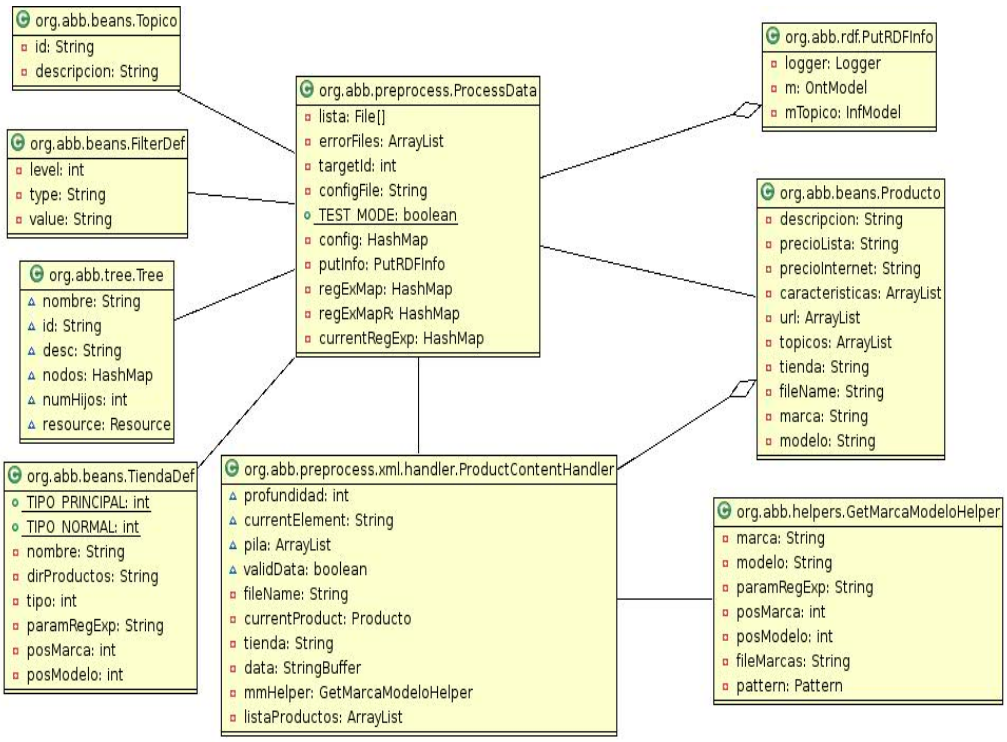

Figura 4.7: Diagrama de Clases Pre-Procesamiento de Productos

- **org.abb.preprocess.ProductContentHandler**: Handler SAX para procesamiento de los XML de productos.
- **org.abb.helpers.GetMarcaModeloHelper**: Encargado de obtener la marca y el modelo de un prodcuto mediante el procesamiento de texto usando expresiones regulares.
- **org.abb.rdf.PutRDFInfo**: Clase encargada de todas las operaciones contra la Base de Datos RDF usando JENA.
- **org.abb.beans.Producto**: Bean para almacenar la información de un producto.
- **org.abb.beans.Topico**: Bean para almacenar la información de un tópico o categoría.
- **org.abb.beans.FilterDef**: Bean para almacenar la información de la configuración de filtros para tópicos.

- **org.abb.beans.Tree**: Estructura de datos para la generación dinámica del árbol de categorías.
- **org.abb.beans.TiendaDef**: Bean para almacenar la información de la configuración de una tienda.

4.5.2. Clasificación

Usando los datos de entrenamiento generados en la etapa de pre-procesamiento, se entrena un clasificador SVM [68] [33] [40] [35], obteniéndose como resultado un modelo. Usando este modelo se clasifican los productos que aparecen en el archivo con datos a clasificar. El resultado de este proceso es la predicción de la categoría hoja que corresponde a cada producto procesado.

Los SVM por naturaleza son clasificadores binarios. Para este problema se uso la implementación LibSVM [24] que es capaz de realizar clasificaciones de múltiples clases usando la estrategia "uno contra uno". Esta estrategia es ampliamente usada en clasificadores de multiples clases y fue introducida en [51].

El proceso descrito fue implementado usando RapidMiner [58] y su Text Plugin que otorga facilidades de lematización y filtrado para el procesamiento de texto.

4.5.3. Medición

Para medir el funcionamiento de este módulo se usó el método de validación cruzada, procedimiento estándar para medir efectividad de tareas de clasificación. Por la naturaleza del problema de clasificación existen muchas clases y pocos datos de entrenamiento por cada clase. Para medir la clasificación se eligieron las categorías de ripley.cl que contengan más productos y se completaron con productos de paris.cl hasta completar 20 productos por categoría.

El clasificador SVM depende de varios parámetros que definen la precisión del clasificador de-

pendiendo del problema. El primer parámetro es el kernel usado, que en este caso se definió como *RBF* (Radial Basis Function [61]) por la característica no lineal del problema. Dado que el kernel es RBF se deben definir dos parámetros adicionales para afinar el clasificador, *C* y *gamma*. *C* es un número que representa la distancia mínima que debe existir entre el hiperplano resultante y los hiperplanos que representan los bordes de los conjuntos que separan el hiperplano resultante. *gamma* es el parámetro para el kernel RBF usado con el clasificador SVM.

Para definir la mejor combinación de parámetros se ejecutó la prueba de validación cruzada con distintos valores para el clasificador SVM. La validación cruzada consiste en dividir el conjunto de datos de entrenamiento en *k* subconjuntos, de los cuales se obtienen *k-1* para entrenar el clasificador y el restante se usa como datos de validación del modelo. Este proceso se realiza *k* veces, usando cada subconjunto como conjunto de datos de validación. Para este experimento se dividio el conjunto de datos de entrenamiento en 10 subconjuntos que es el valor estándar para este tipo de procesos. El resultado de las pruebas se muestra en la Tabla 4.4.

gamma	**C**	**Error %**	**Varianza**
1,0	3,0	17,29	3,57
0,6	3,0	15,73	3,64
0,2	3,0	17,44	2,71
1,0	4,0	17,13	3,80
0,6	4,0	14,34	2,89
0,2	4,0	14,49	4,01
1,0	2,0	17,60	3,41
0,6	2,0	15,58	3,50
0,2	**2,0**	**14,17**	**4,86**

Tabla 4.4: Resultados ejecución de validación cruzada para distintos valores de *gamma* y *C*

Los mejores resultados según pruebas de validación cruzada se obtuvieron con *gamma=0,2* y *C=2,0*, con los que el clasificador entrego un porcentaje de error de 14,17 %.

Los buenos resultados arrojados por la prueba de validación cruzada, disminuyen un poco al momento de clasificar todo el universo de productos. De igual manera sirven para los objetivos de

este trabajo. En la Tabla 4.5 se muestra el resultado de la clasificación en términos de precisión y recuperación.

La Tabla 4.5 muestra que la precisión y recuperación depende de la naturaleza d la categoría. Existen categorías que son totalmente disjuntas con todo el resto, como el caso de las categorías que muestran una precisión de 100 %. Este es el caso de aspiradoras, juegos de living, veladores y comodas, microondas, cubrecamas, y plumones. Existen otros casos en que el clasificador no logra identificar correctamente la diferencia entre categorías similares como Box Spring 2 Plazas y Combo completo Box Spring 2 Plazas.

La principal necesidad de una buena clasificación para este tipo de buscadores, es que no hayan falsos positivos, o sea que no aparezcan televisores en la categoría de lavadoras. En este sentido, de las 27 categorías usadas para el experimento, se obtuvieron 6 categorías que muestran una precisión de 100 % para el clasificador.

Finalmente los parametros totales de recuperación y precisión están cercanos al 80 %, lo cual se considera bueno y en un ambiente productivo se puede completar la tarea usando clasificación manual.

4.6. Motor de Recomendación

Es el proceso encargado de identificar productos similares y establecer explicitamente una relación entre estos mediante una propiedad de la Ontología. La similitud entre productos se estima a través de una medida de distancia. El proceso de comparación toma en cuenta sólo productos que pertenecen a la misma categoría hoja de la taxonomía. En la sección 5 se explica en detalle éste proceso y la medida de distancia utilizada.

Categoría	Precisión(%)	Recuperación(%)
Audio - Minicomponentes	98	63
Audio - Home Theaters	100	100
Box Spring - King	80	55
Box Spring - 2 Plazas	82	57
Celulares - Sobre $100.000	93	95
Celulares - Hasta $30.000	77	92
Celulares - Hasta $99.990	55	71
Coches - Coche Paragua & Paseo	60	78
Combo Completo Box Spring - 2 Plazas	67	72
Cuidado Personal - Secadores de Pelo	95	91
Deportes - Estáticas y Spinning	89	89
Electrodomésticos - Aspiradoras	100	93
Electrodomésticos - Microondas	100	83
Electrodomésticos - Máquinas de coser	100	100
Electrodomésticos - Hervidores	98	100
Electrodomésticos - Planchas y mesas	52	92
Electrodomésticos - Licuadoras	85	96
Lavado y Secado - Secadoras	93	93
Lavado y Secado - Lavadora Carga Vertical	55	73
Maletería - Mochilas	44	69
Muebles de Comedor - Juegos de comedor	30	75
Muebles de Dormitorio - Respaldos	69	94
Muebles de Dormitorio - Veladores y Cómodas	100	84
Muebles de Living - Juegos de Living	100	41
Muebles de Living - Mesas de Arrimo	54	63
Muebles de Living - Mesas de Centro	48	61
Mundo Bebé - Sillas de Auto	93	91
Refrigeración - No-Frost	85	76
Refrigeración - Side by Side	57	57
Relojes - Reloj mujer	40	90
Relojes - Reloj hombre	42	100
Ropa de Cama - Cubrecamas y Colchas	100	91
Ropa de Cama - Plumones	100	89
Ropa de Cama - Almohadas	94	98
Ropa de Cama - Frazadas	91	91
Ropa de Cama - Sábanas	95	99
Total	**79,23**	**79,56**

Tabla 4.5: Precisión y recuperación por categoría

4.7. Indexador

Proceso encargado de generar el índice sobre el cual se hará la búsqueda basada en keywords. Este proceso usa la API de Lucene [39] para la generación del índice. El uso de un índice aparte de la información almacenada en la Ontología se debe a que al tiempo de respuesta en este tipo de búsquedas es importante, y las consultas de texto directamente a la Ontología no son eficientes.

4.8. Aplicación Web

Consiste en la aplicación de consultas para realizar la búsqueda de productos. Es una aplicación WEB construida usando tecnologías J2EE y el framework Struts [16]. La aplicación provee una interfaz gráfica simple, presentando en todo momento un cuadro de texto para realizar búsqueda por keywords, y un menú lateral con el árbol de categorías.

4.8.1. Arquitectura

La figura 4.8 muestra los componentes de la aplicación Web, la cual sigue el modelo de tres capas (presentación, lógica y datos) aprovechando el patrón MVC (Model View Controller) implementado por el framework Struts. La capa de presentación está compuesta por las páginas JSP que componen la interfaz gráfica y las acciones Struts. La capa de lógica corresponde a JavaBeans que contienen las reglas de negocio para la manipulación de la información obtenida desde la capa de datos. Finalmente la capa de datos esta compuesta por una Base de Datos RDF, donde se encuentra la Ontología, y por el índice Lucene que es consultado para obtener los documentos que cumplen con un criterio de búsqueda.

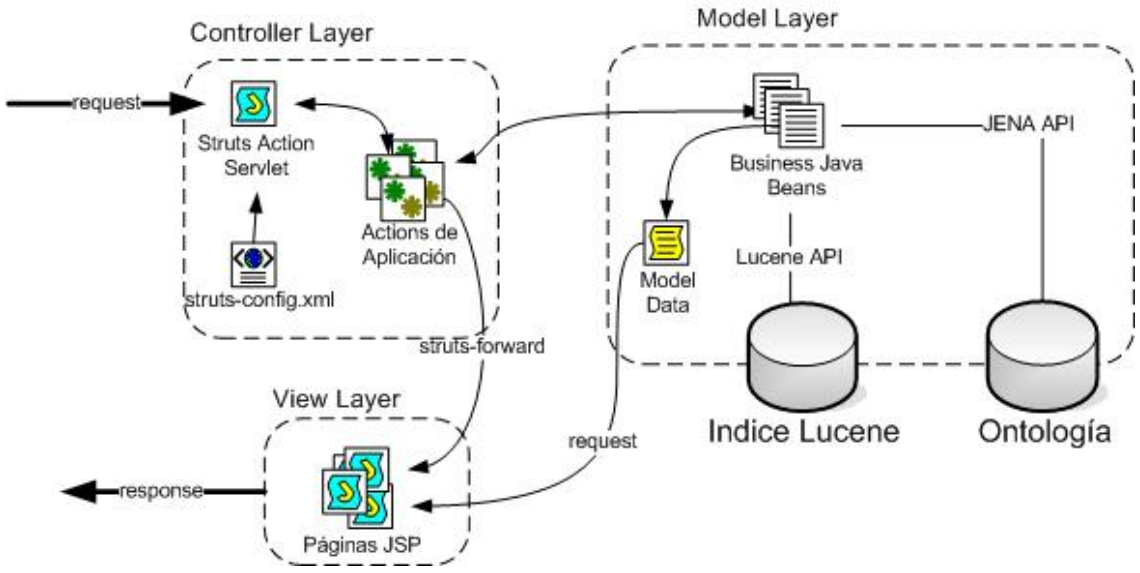

Figura 4.8: Arquitectura de la Aplicación Web

4.8.2. Funcionalidad y Visualización

El buscador presenta las principales funcionalidades de un buscador Web, cómo la búsqueda por keywords y navegación por categorías, pero se agrega la funcionalidad de ver productos similares para cada producto de la lista resultante. La visualización del sitio presenta algunas diferencias con los estándares de la industria. A continuación se detallan las funcionalidades y se presenta el sitio gráficamente.

Búsqueda por Keywords

Consiste en la búsqueda por palabras ingresadas por el usuario en el sitio. El usuario ingresa las palabras que desea buscar en el campo de texto ubicado en la parte superior del buscador y selecciona el botón ”Buscar”. El resultado se despliega en forma de lista. La búsqueda usa el operador lógico AND para multiples keywords, pero soporta el operador lógico OR si se especifica entre cada palabra que compone la consulta. La visualización de esta funcionalidad se muestra en la figura 4.9

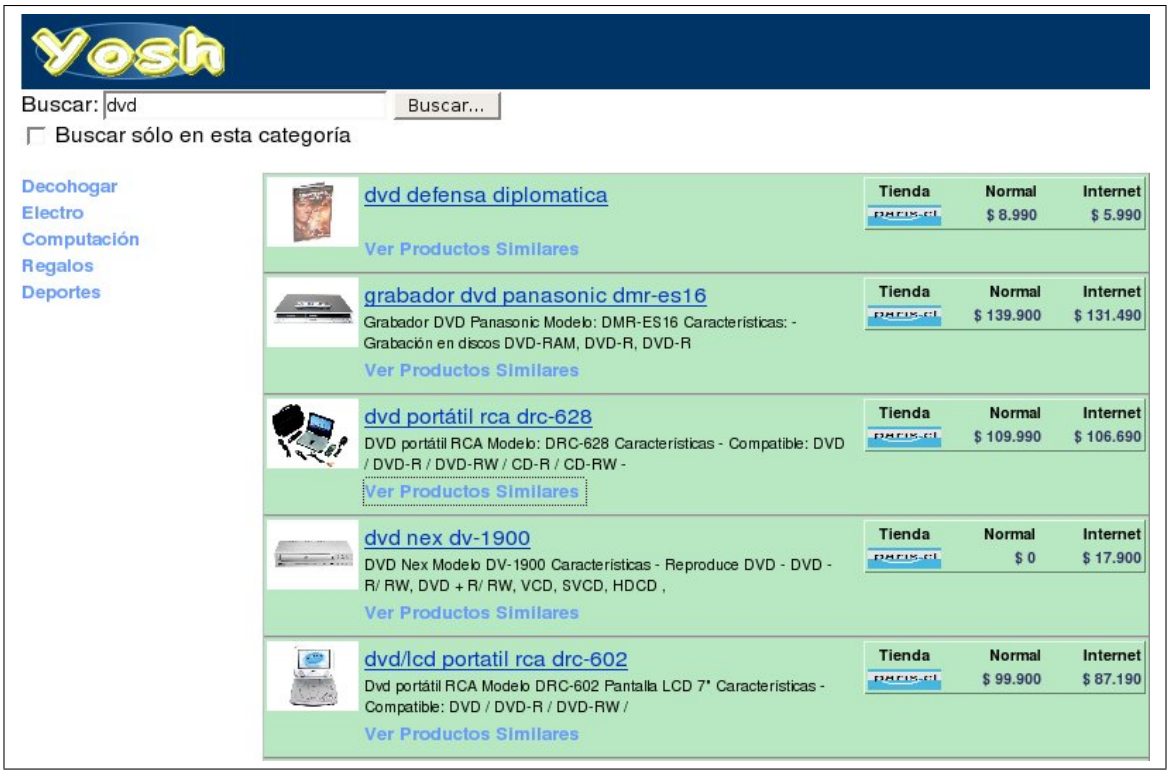

Figura 4.9: Visualización Búsqueda por Keywords

Campo	Indexado	Almacenado
ID Producto	NO	SI
Contenido	Tokenizado	NO

Tabla 4.6: Estructura del índice Lucene para YOSH

Los resultados se obtienen accediendo al índice Lucene generado previamente a través de APIs. La estructura del índice utilizado es muy simple y se detalla en la Tabla 4.6.

El campo *ID Producto* es un identificador único que genera el Sistema para hacer referencia a cada producto, y es el mismo identificador que se almacena en la Ontología. Este campo es almacenado por Lucene para poder identificar el producto resultante de una búsqueda. El campo *contenido* corresponde a todo el texto que aparece en la ficha de cada producto y es *tokenizado* por Lucene para obtener las palabras a indexar.

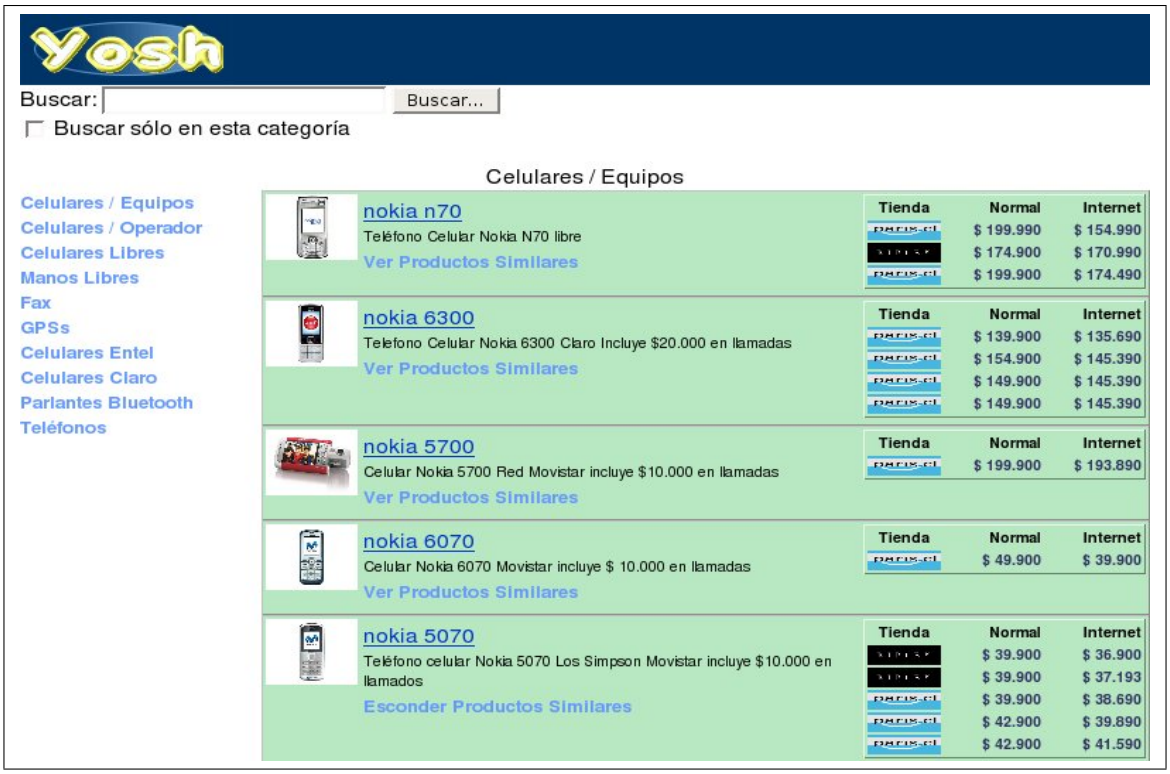

Figura 4.10: Visualización Búsqueda por Catálogo

Búsqueda por Catálogo

Consiste en la búsqueda de productos usando el árbol de categorías (catálogo) que se presenta en cada pantalla del sitio. El usuario debe ir recorriendo por conceptos según el producto que esté buscando hasta llegar a una categoría hoja que es donde se encuentran los productos. Los productos correspondientes a la categoría seleccionada se presenta en forma de lista. La visualización de esta funcionalidad se muestra en la figura 4.10

Este tipo de búsqueda es resuelta internamente por la aplicación consultando la Ontología usando el lenguaje de consultas SPARQL. La consulta SPARQL se presenta en la Tabla 4.7, la cual en palabras simples obtiene todos los recursos de tipo *bsce:Producto* que cumplen con la propiedad *bsce:contiene* para la categoría *topicResource* que es parámetro.

```
PREFIX bsce: <http://www.abb.org/bsce#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?producto
WHERE  {<topicResource> bsce:contiene ?producto .
        ?producto rdf:type bsce:Producto . }
```

Tabla 4.7: Consulta SPARQL para consulta por categoría

Figura 4.11: Visualización búsqueda por keywords en una categoría específica

Búsqueda por Keywords en una Categoría

Consiste en la búsqueda de productos usando keywords pero limitando la búsqueda sobre los productos asociados a una categoría. El usuario debe ir recorriendo por conceptos según el producto que esté buscando hasta llegar a una categoría hoja que es donde se encuentran los productos, luego debe escribir la consulta en el campo de texto superior, activar el cuadro “Buscar sólo en esta categoría” y finalmente presionar el botón “Buscar”. El resultado se despliega en forma de lista. La visualización de esta funcionalidad se muestra en la Figura 4.11

Figura 4.12: Visualización de resultados de búsquedas

Internamente esto se resuelve obteniendo los productos que cumplan con el criterio de búsqueda desde el índice Lucene usando APIs. Luego por cada producto se consulta a la Ontología por la relación entre la categoría seleccionada y el producto.

Visualización de Resultados

Todos los tipos de búsqueda en el Sistema obtienen como resultado una lista de productos que provee una forma única de visualizar los resultados al usuario. Para cada producto en la lista se muestra foto, descripción, y precios de lista e internet por cada tienda en que se venda dicho producto. Esto último se presenta como una diferencia a los estándares de este tipo de Sistemas, ya que lo más usado es mostrar el menor precio o derechamente mostrar como registros separados un mismo producto vendido por distintas tiendas. El mostrar los precios en forma de lista identificando la tienda permite que el usuario tenga más información para tomar decisión con menos clicks, tomando en cuenta variables como la fidelidad a una tienda. La visualización en detalle de esta funcionalidad se muestra en la figura 4.12

Visualización de Productos Similares

Cada producto que se presenta como resultado de una búsqueda puede estar asociado a productos similares que se pueden visualizar en la misma pantalla de resultado. El usuario debe presionar el link “Ver productos similares” ubicado debajo de la descripción del producto a comparar. El resultado se despliega en forma de lista ubicada entre el producto seleccionado y el siguiente pro-

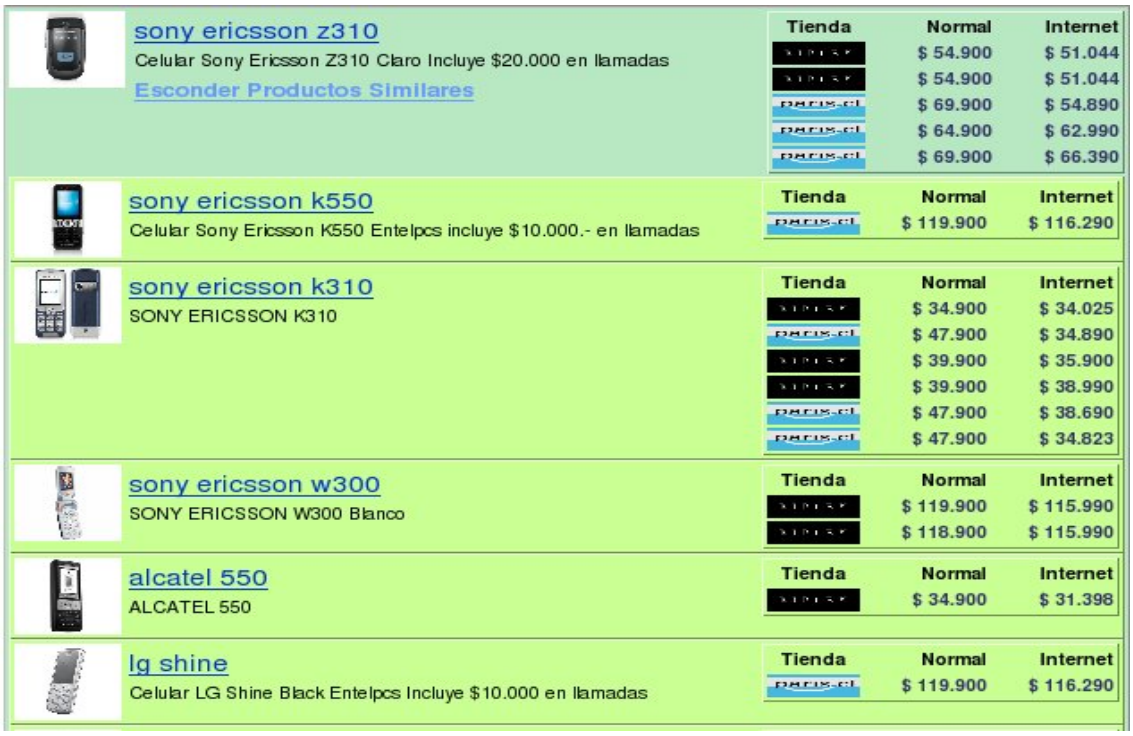

Figura 4.13: Visualización de productos similares

ducto, diferenciándose sólo por el color de fondo de los registros. El resultado de los productos similares se puede esconder mediante el link “Ocultar productos similares”.

Esta funcionalidad esta implementada con AJAX para evitar actualizar la página completa y mejorar la experiencia del usuario al usar el Sistema. La visualización de esta funcionalidad se muestra en la figura 4.13

Esto se resuelve internamente usando la relación *es_similar_a* de la Ontología asociada al producto que se está consultando.

Capítulo 5

Motor de Recomendaciones

Para mejorar la experiencia del cliente al comparar productos, se creó un proceso que evalúa cada producto y lo asocia con los productos de mayor similitud. Estos productos deben pertenecer a la misma categoría. La similitud es calculada mediante medidas de distancia, cuya elección es parte de este trabajo. Además este proceso se encarga de identificar productos iguales entre distintas tiendas, tomando en cuenta la medida de distancia, la marca y el modelo obtenidos en procesos anteriores.

5.1. Similitud Entre Productos

Para poder evaluar las medidas de distancia, se tomaron en cuenta cuatro definiciones de similitud entre productos. Las definiciones se detallan a continuación:

A - Productos iguales : Los productos tienen la misma marca y modelo. Se trata del mismo producto vendido en distintas tiendas, y por lo tanto no coinciden 100 % en su descripción, pero se trata del mismo articulo.

B - Productos comparables : Los productos no son iguales, pero comparten características de definición, por ejemplo aro en bicicletas, potencia en equipos de audio, pulgadas en televisores, etc.

C - Productos no comparables : Los productos no son iguales, comparten algunas características de definición, pero estas no tiene valores equivalentes.

D - Productos diferentes : Los productos no son iguales y no tienen ninguna característica en común.

Se puede apreciar que las definiciones B y C pueden ser bastante subjetivas al momento de comparar dos productos, es por esto que es necesario experimentar con distintas medidas de distancia para discriminar, básicamenbte entre estas dos definiciones.

5.2. Medidas de Distancia

Para el presente trabajo se definieron tres medidas de distancia entre productos, las cuales se usarán en el experimento que definirá la medida a usar por el motor de recomendaciones.

5.2.1. Distancia Coseno

Es la medida de distancia estándar en Sistemas de IR. Se trabaja cada producto como un vector de palabras TF-IDF. El espacio vectorial está dado por todas las palabras de los productos de una categoría hoja, sin considerar *stopwords* ni palabras con menos de 3 letras. La distancia entre dos productos corresponde al producto punto entre los vectores normalizados que representan ambos productos. La distancia entre un documento d_1 y d_2 esta dada por:

$$sim(d_1, d_2) = \frac{\vec{d_1} \bullet \vec{d_2}}{|\vec{d_1}| \times |\vec{d_2}|}$$

En 2.1.2 se explica en detalle el origen de esta fórmula.

5.2.2. Distancia de Características Principales

Cada producto se define a través de su descripción y su lista de características. La idea principal de esta medida de distancia es identificar cuales son las características principales en una catergoría (ej: pulgadas y contraste en televisores LCD), y comparar cada producto según la aparición de dichas características.

Para cada categoría se obtienen las palabras más recurrentes entre sus productos (sin tomar en cuenta stopwords). Dichas palabras corresponden a palabras ocurrentes en las características principales de la categoría. La distancia entre dos productos se calcula mediante el producto punto entre las características identificadas por las palabras recurrentes. La distancia entre un documento d_1 y d_2 esta dada por:

Definición: *Sea P el conjunto de las 5 palabras que más aparecen entre los productos de una una categoría y $C_{p,d}$ el conjunto de características del documento d que contienen la palabra p, donde $p \in P$. La similitud entre dos características está dada por*

$$sim_p(c_{p,d_i}, c_{p,d_j}) = \frac{\vec{c_{p,d_i}} \bullet \vec{c_{p,d_j}}}{min\ (\mid \vec{c_{p,d_i}} \mid , \mid \vec{c_{p,d_j}} \mid)}$$

Donde $\vec{c_{p,d_i}}$ y $\vec{c_{p,d_j}}$ son los vectores que representan una característica que contiene la palabra p, en los documentos d_i y d_j respectivamente. Cada dimensión de estos vectores corresponde una palabra (sin considerar p), y el número total de dimensiones corresponde al total de palabras distintas que aparecen en $c_{p,d_i} \in C_{p,d_i}$ y $c_{p,d_j} \in C_{p,d_j}$, nuevamente sin considerar p. Entonces la similitud entre un documento d_i y un documento d_j está dada por

$$sim(d_i, d_j) = \frac{\sum_{p \in P} \mid max_{i,j}\ sim_p(c_{p,d_i}\ ,\ c_{p,d_j}) \mid}{\mid P \mid}$$

Por ejemplo si tenemos los siguientes productos con sus características pertenecientes a la categoría de “Televisores LCD”.

Producto1 (Televisor LG LCD 37 pulgadas)

- Tamaño de Pantalla: 37 pulgadas
- Resolución: Full HD 1920x1080

Producto2 (Televisor LG LCD 37 pulgadas)

- Pantalla 37 pulgadas
- Resolución 1920x1080
- Resolución Full HD

Donde las palabras frecuentes de la categoría son "resolución" y "pulgadas" tenemos que el cálculo de similitud es el siguiente:

$$sim(d_i, d_j) = \frac{max\ sim_{resolucion}(c_{d_i}\ ,\ c_{d_j}) + max\ sim_{pulgadas}(c_{d_i}\ ,\ c_{d_j})}{\mid P \mid}$$

$$sim(Producto1, Producto2) = \frac{0{,}33 + max\ (0{,}22\ ,\ 0{,}33)}{2} = 0{,}33$$

5.2.3. Distancia Basada en Tags

Al igual que en la distancia anterior, la idea principal es identificar características. Pero a diferencia de la medida anterior, se usa la técnica de tags tomando como documento de entrada la unión de la información de características de todos los productos de una categoría. La distancia entre dos productos se calcula mediante el producto punto entre las características identificadas por los tags asociados a la categoría.

Definición: *Sea T el conjunto de tags más relevantes de una categoría y $C_{t,d}$ el conjunto de características del documento d que contienen el tag t, donde $t \in T$. La similitud entre dos características está dada por*

$$sim_t(c_{t,d_i}, c_{t,d_j}) = \frac{\vec{c_{t,d_i}} \bullet \vec{c_{t,d_j}}}{min\ (|\ \vec{c_{t,d_i}}\ |\ ,\ |\ \vec{c_{t,d_j}}\ |)}$$

Donde $\vec{c_{t,d_i}}$ y $\vec{c_{t,d_j}}$ son los vectores que representan una característica que contiene la palabra p, en los documentos d_i y d_j respectivamente. Cada dimensión de estos vectores corresponde una palabra (sin considerar p), y el número total de dimensiones corresponde al total de palabras distintas que aparecen en $c_{t,d_i} \in C_{t,d_i}$ y $c_{t,d_j} \in C_{t,d_j}$, nuevamente sin considerar p. Entonces la similitud entre un documento d_i y un documento d_j está dada por

$$sim(d_i, d_j) = \frac{\sum_{t \in T} |\ max\ sim_t(c_{t,d_i}\ ,\ c_{t,d_j})\ |}{|\ T\ |}$$

Por ejemplo si tenemos los siguientes productos con sus características pertenecientes a la categoría de “Televisores LCD”.

Producto1 (Televisor LG LCD 37 pulgadas)

- Tamaño de Pantalla: 37 pulgadas
- Resolución: Full HD 1920x1080

Producto2 (Televisor LG LCD 37 pulgadas)

- Pantalla 37 pulgadas
- Resolución 1920x1080
- Resolución Full HD

Donde las palabras frecuentes de la categoría son “37 pulgadas” y “full HD” tenemos que el cálculo de similitud es el siguiente:

$$sim(d_i, d_j) = \frac{max\ sim_{37pulgadas}(c_{d_i}\ ,\ c_{d_j}) + max\ sim_{fullHD}(c_{d_i}\ ,\ c_{d_j})}{|\ P\ |}$$

$$sim(Producto1, Producto2) = \frac{0{,}33 + 0{,}33}{2} = 0{,}33$$

5.3. Definición del Experimento

Para determinar la medida de distancia más adecuada para comparar productos usando su descripción, se seleccionaron 500 pares de productos del universo total de productos que incluye los obtenidos desde *falabella.cl*, *ripley.cl* y *paris.cl*.

Para la selección de los pares se usó el siguiente procedimiento:

1. Seleccionar 500 pares de productos aleatoriamente. Cada producto de un par pertenece a la misma categoría.
2. Clasificar manualmente pares dentro de cada categoría usando los criterios mencionados en el punto 5.1.
3. Seleccionar las categorías cuyos pares clasificados sean posibles de balancear
4. Balancear categorías, de tal manera que exista una cantidad equivalente de pares para cada tipo de similitud.
5. Consolidar información de pares, obteniendo el listado final de pares a usar en el experimento.

El resultado de la clasificación manual se usa como referencia para validar los distintos modelos que se probarán. Dichos modelos se construirán usando las tres medidas de distancia propuestas y distintas técnicas de clasificación como árboles J48, NaiveBayes y OneR

5.4. Resultados y Discusión

La primera iteración del experimento tuvo como objetivo identificar que técnica de clasificación da mejor resultado con cada medida de distancia propuesta. Los resultados se presentan en la

Clasificación / Distancia	OneR	NaiveBayes	J48
Coseno	52 %	56 %	**61 %**
CP	51 %	**57 %**	54 %
Tags	52 %	**59 %**	57 %
Promedio	51 %	**57 %**	**57 %**

Tabla 5.1: Tabla de precisión por medida de distancia y modelo de clasificación

Modelo / Topico	J48 Cos	NB CP	NB Tag
Deportes - Estáticas y Spinning	**55,00 %**	51,67 %	38,33 %
Electrodomésticos - Máquinas de coser	54,67 %	43,67 %	**63,67 %**
Electrodomésticos - Microondas	61,50 %	**62,00 %**	52,50 %
Electrodomésticos - Hervidores	25,83 %	**26,67 %**	22,50 %
Refrigeración - No-Frost	40,00 %	**55,00 %**	30,83 %
Celulares - Sobre $100.000	48,33 %	**61,67 %**	60,00 %
Lavado y Secado - Lavadora carga vertical	**58,33 %**	31,67 %	23,33 %
Celulares - Hasta $30.000	45,00 %	46,67 %	**51,67 %**
Audio - Home Theaters	**80,00 %**	52,00 %	75,00 %
Celulares - Hasta $99.990	55,00 %	**61,50 %**	42,00 %
Refrigeración - Side by Side	65,00 %	50,00 %	**69,17 %**

Tabla 5.2: Tabla de precisión por tópico de clasificación

Tabla 5.1, de la cual rescatamos como conclusión que el uso de árboles J48 con la distancia coseno y NaiveBayes para las otras distancias, obtuvo mejores resultados. La combinación J48 con distancia coseno fue la que mostró mejor desempeño general.

El siguiente experimento tuvo como objetivo ver en detalle el comportamiento de las medidas de distancia, usando la técnica de clasificación elegida en el punto anterior. Este detalle muestra el comportamiento por categoría de producto. El resultado se muestra en la Tabla 5.2.

Los resultados no son buenos en términos de precisión, pero muestran que una de las distancias propuestas se comporta mejor para algunas categorías que la distancia coseno.

En este punto comienza a tomar relevancia el concepto de categoría y su influencia en los resul-

Clasificación / Distancia	OneR	NaiveBayes	J48	J48+Cat
Coseno	52 %	56 %	**61 %**	**61 %**
CP	51 %	57 %	54 %	**66 %**
Tags	52 %	**59 %**	57 %	58 %

Tabla 5.3: Tabla de precisión por medida de distancia y modelo de clasificación, usando categorías

tados de cada experimento. Los productos asociados a una categoría pueden tener un vocabulario muy amplio o muy reducido para describir cada producto, lo que provoca que las dimensiones de los vectores asociados a los productos de cada categoría puedan ser muy distintos. Esto influye en los rangos que identifican la similitud entre productos, por ejemplo para una categoría el rango A puede ir entre 0.5 y 1.0, y para otra categoría puede ir entre 0.75 y 1.0. Además se debe considerar que tanto la distancia por características principales como la basada en Tags, se basan en información propia de cada categoría para el cálculo de similitud.

Por lo mencionado en el parrafo anterior se propone agregar la categoría al cálculo de medida de distancia, usando árboles de decisión. En la Tabla 5.3 se muestran los resultados de los nuevos modelos de clasificación versus la estrategia anterior.

La Tabla 5.3 muestra una mejora de la clasificación usando la categoría junto con la medida de distancia basada en características principales. Esta mejora además supera el rendimiento de la distancia coseno que no se vió afectada por el uso de la categoría en la clasificación. En la Tabla 5.4 se completa el cuadro de precisión versus estrategia de calsificación con los datos de la clasificación J48 CP+Cat por categoría.

Finalmente un parámetro importante a estudiar en detalle es la cantidad de falsos positivos que generan las clasificaciones de similitud estudiadas. Esta medición es igual de relevante que la cantidad de clasificaciones positivas ,ya que para la experiencia del usuario final que dos productos no aparezcan como similares puede ser perdonable y hasta puede no percatarse del hecho. Ocurre lo contrario cuando se presentan dos productos totalmente distintos como similares.

Modelo / Topico	J48 Cos	NB CP	NB Tag	J48 CP+Cat
Deportes - Estáticas y Spinning	**55,00 %**	51,67 %	38,33 %	25,00 %
Electrodomésticos - Máquinas de coser	54,67 %	43,67 %	63,67 %	**69,33 %**
Electrodomésticos - Microondas	61,50 %	62,00 %	52,50 %	**79,50 %**
Electrodomésticos - Hervidores	25,83 %	26,67 %	22,50 %	**31,67 %**
Refrigeración - No-Frost	40,00 %	**55,00 %**	30,83 %	36,67 %
Celulares - Sobre $100.000	48,33 %	**61,67 %**	60,00 %	45,00 %
Lavado y Secado - Lavadora carga vertical	**58,33 %**	31,67 %	23,33 %	41,67 %
Celulares - Hasta $30.000	45,00 %	46,67 %	51,67 %	**55,00 %**
Audio - Home Theaters	**80,00 %**	52,00 %	75,00 %	67,50 %
Celulares - Hasta $99.990	55,00 %	61,50 %	42,00 %	**74,00 %**
Refrigeración - Side by Side	65,00 %	50,00 %	**69,17 %**	56,67 %

Tabla 5.4: Tabla de precisión por tópico de clasificación

La Tabla 5.5 muestra la cantidad de positivos verdaderos y falsos para las clases A (productos iguales) y B (productos similares). Esta tabla muestra el árbol J48 usando la distancia de características principales y la categoría, es la que obtiene mayor cantidad de verdaderos positivos para la clase B y la menor cantidad de falsos positivos para la clase A. La distancia basada en Tags con clasificador NaiveBayes obtuvo la menor cantidad de falsos positivos para la clase B pero la relación entre falsos y verdaderos favorece a los falsos en ambas clases. Por último la distancia Coseno con clasificador J48 obtiene mayor cantidad de verdaderos positivos para la clase A, pero también obtiene una gran cantidad de falsos positivos, que es lo que se trata de evitar para esta aplicación.

Similitud	Tipo Positivos	J48 Cos	NB CP	NB Tag	J48 CP+Cat
A	Verdaderos	**63**	47	52	44
A	Falsos	54	44	66	**41**
B	Verdaderos	30	26	17	**33**
B	Falsos	40	45	**36**	45

Tabla 5.5: Tabla de comparativa de Falsos Positivos según medida de distancia y modelo de clasificación

Con todos los antecedentes mencionados anteriormente se selecciona la medida de distancia

por características principales, usando árboles J48 y la categoría, como método de clasificación de similitud entre dos productos. Esto por tener el mejor desempeõ de clasificación general y presentar bajos índices de falsos positivos.

5.5. Módulo de Procesamiento de Relaciones

Consiste en un programa que recorre las categorías hoja de la taxonomía, por cada hoja compara todos los productos de dicha categoría entre sí. Para cada producto obtiene un producto base, que es el producto que representará a un conjunto de productos iguales, y que se referenciará al realizar búsquedas y cualquier otro proceso sobre los productos. La identificación de productos iguales se realiza detectando marca y modelo usando expresiones regulares y ponderando la clasificación de similitud obtenida, la cual debe ser A para considerarse iguales.

Por cada producto base se obtiene una lista de productos similares, usando la clasificación de similitud y medida de distancia. Para esto se seleccionan las 5 mejores distancias entre el conjunto de productos (representados por el producto base) y el resto de los productos de la categoría. Finalmente la relación de similitud asocia siempre a 2 o más productos bases.

Las asociaciones se "producto" a "producto base" y de "producto base" a "producto base similar" se generan de forma explicita en la ontología mediante las propiedades *es_similar_a*, *es_base_de* y *es_hijo_de*. Estas propiedades se encuentran definidas en la Ontología explicada en el punto 4.2.

Capítulo 6

Conclusiones y Trabajo Futuro

El buscador para comercio electrónico desarrollado en esta tesis cumple con los siguientes objetivos:

- Recolección de información desde las tiendas, a través de Crawlers ad-hoc para cada tienda, basados en una implementación base, lo que permite disminuir el tiempo de desarrollo para nuevas tiendas.
- Estructurar la información obtenida desde las distintas tiendas y guardarla en un formato único. Esto se logró a través del uso de plantillas XSLT para cada tienda, y el uso de una Ontología como modelo conceptual.
- Generar un árbol de tópicos para búsqueda por navegación y clasificación de productos. Logrado tomando como base el árbol de categorías de una de las tiendas, y clasificando los productos usando un clasificador SVM.
- Desarrollar e implementar estrategias para comparación de productos. Logrado a través de la implementación de tres medidas de distancia y usando distintas estrategias de clasificación. La estrategia seleccionada para la implementación final se eligió basándose en resultados

experimentales.

- Implementar una aplicación Web que provea una interfaz para búsqueda de productos por keyword, y navegación por categorías. Logrado mediante la implementación de una aplicación WEB que usa búsqueda textual a través del motor Lucene, y obtiene relaciones desde la Ontología.

Como resultado de esta tesis, se obtuvo un buscador para comercio electrónico, que presenta ventajas tanto en la administración del Sistema, como para el usuario. En la administración la ventaja es que todos los procedimientos son automáticos, y sólo requieren de intervención humana en caso de fallas, principalmente en las tareas de clasificación.

Para el usuario la mayor ventaja que presenta este Sistema es la facilidad para comparar productos, ya que al detectar qué productos son iguales (independiente de que tienda se vendan) se presenta como resultado un sólo producto con los precios de las distintas tiendas.

Finalmente el Usuario tiene la posibilidad de obtener una lista de productos similares para cada producto obtenido como resultado de una búsqueda. Estos productos se obtienen a partir de las características de los prodcutos, por lo que se asemeja más a la búsqueda realizada por un comprador en una tienda física.

El trabajo futuro se debe focalizar en los siguientes puntos:

- Implementar Crawlers para más tiendas de comercio electrónico chilenas, de tal manera de darle al usuario mayores posibilidades de encontrar el producto que necesita y al mejor precio.

- Mejorar las técnicas de clasificación ya que es el módulo que tiene mayores falencias. Puntualmente se debe encontrar una estrategia que no use la taxonomía de una tienda ya existente, y se debe mejorar la precisión del módulo clasificador.

- Incrementar la Ontología de tal manera de agregar conceptos al dominio usado por el buscador. Los conceptos más evidentes a agregar son los relacionados con información de despacho y medios de pago. De tal manera de poder responder preguntas del estilo “cámara sony pago con VISA” o “camara sony despacho en 2 días”.

- Mejorar las técnicas recomendación. En este sentido las medidas de distancia y técnicas de clasificación estudiadas en este trabajo no presentaron un buen desempeño, pese a que se mejoró la presición de la medida de referencia (distancia coseno).

- Usar la estrategia diseñada para la distancia de características prparaincipales, aplicada a detectar atributos representativos de una característica, con el fín de generar filtros guiados por el usuario en vez de presentar productos similares.

Los primeros dos puntos de la lista anterior son necesarios para obtener una versión final del buscador y podría implantarse en un ambiente productivo. En el caso del primer punto es necesario entregar más alternativas de productos ya que la actual implementación contempla sólo tres tiendas. En cuanto al segundo punto, es necesario para evitar dependencias y para tomar en cuenta productos que no sean considerados por la tienda desde donde se está obteniendo el árbol de categorías. Este es un esfuerzo de implementación considerable ya que contempla diseño de estrategias para clasificar en una nueva taxonomía donde no se tiene conjunto de entrenamiento definido.

Los puntos mencionados anteriormente corresponden sólo a mejoras del buscador actualmente implementado.

Referencias

[1] AltaVista.

http://www.altavista.com.

[2] Bizrate.

http://www.bizrate.com.

[3] Buscapé.

http://www.buscape.com.

[4] Ciao.

http://www.ciao.co.uk.

[5] Confronte.

http://www.confronte.com.

[6] Epinions.

http://www.epinions.com.

[7] Google.

http://www.google.com.

[8] Hakia.

http://www.hakia.com.

[9] PriceGrabber.

http://www.pricegrabber.com.

[10] Pricescan.

http://www.pricescan.com.

[11] Sensebot.

http://www.sensebot.net.

[12] Tim Berners-Lee.

http://www.w3.org/People/Berners-Lee/.

[13] Yahoo!

http://www.yahoo.com.

[14] Youcef Aklouf, Guy Pierra, Yamine Aït Ameur, and Habiba Drias. Plib ontology for b2b electronic commerce. In *ISPE CE*, pages 269–278, 2003.

[15] Boanerges Aleman-Meza, Chris Halaschek, Budak I. Arpinar, and Amit Sheth. Context-aware semantic association ranking. In *Semantic Web and Databases Workshop Proceedings*, pages 33–50, Berlin, Germany, September 2003.

[16] Apache. Struts Framework.

http://struts.apache.org.

[17] Ricardo Baeza-Yates and Carlos Castillo. Crawling the infinite web. *Journal of Web Engineering*, 6(1):49–72, February 2007.

[18] Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[19] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.

[20] Dan Brickley, R.V. Guha, and Brian McBride. RDF Vocabulary Description Language 1.0: RDF Schema. January 2003.

[21] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[22] Erik Brynjolfsson and Michael D. Smith. The Great Equalizer? Consumer Choice Behavior at Internet Shopbots. *SSRN eLibrary*, 2001.

[23] Soumen Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan-Kauffman, 2002.

[24] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[25] World Internet Project Chile. WIP Chile 2008 Survey, 2008. `http://comunicaciones.uc.cl/prontus_fcom/site/artic/20080418/mmedia/MULTIMEDIA_220080418230431.pdf`.

[26] Junghoo Cho and Hector Garcia-Molina. Parallel crawlers. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 124–135, New York, NY, USA, 2002. ACM Press.

[27] Junghoo Cho, Hector Garcia-Molina, and Lawrence Page. Efficient crawling through url ordering. In *WWW7: Proceedings of the seventh international conference on World Wide Web 7*, pages 161–172, Amsterdam, The Netherlands, The Netherlands, 1998. Elsevier Science Publishers B. V.

[28] Olivier Corby, Rose Dieng-Kuntz, and Catherine Faron-Zucker. Querying the semantic web with corese search engine. In *ECAI*, pages 705–709, 2004.

[29] Michael C. Daconta, Kevin T. Smith, and Leo J. Obrst. *The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management*. John Wiley & Sons, Inc., New York, NY, USA, 2003.

[30] Centro de Estudios de la Economía Digital - Cámara de Comercio de Santiago. Economía Digital 2009.

[31] Li Ding, Tim Finin, Anupam Joshi, Rong Pan, R. Scott Cost, Yun Peng, Pavan Reddivari, Vishal Doshi, and Joel Sachs. Swoogle: a search and metadata engine for the semantic web. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 652–659, New York, NY, USA, 2004. ACM Press.

[32] H. Drucker, Donghui Wu, and V. N. Vapnik. Support vector machines for spam categorization. *Neural Networks, IEEE Transactions on*, 10(5):1048–1054, 1999.

[33] H. Drucker, Donghui Wu, and V. N. Vapnik. Support vector machines for spam categorization. *Neural Networks, IEEE Transactions on*, 10:1048–1054, 1999.

[34] Susan Dumais, John Platt, David Heckerman, and Mehran Sahami. Inductive learning algorithms and representations for text categorization. In *CIKM '98: Proceedings of the seventh international conference on Information and knowledge management*, pages 148–155, New York, NY, USA, 1998. ACM Press.

[35] Susan Dumais, John Platt, David Heckerman, and Mehran Sahami. Inductive learning algorithms and representations for text categorization. In *CIKM '98: Proceedings of the seventh international conference on Information and knowledge management*, pages 148–155, New York, NY, USA, 1998. ACM Press.

[36] Margaret H. Dunham. *Data Mining: Introductory and Advanced Topics*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2002.

[37] D. J. Foskett. Thesaurus. pages 111–134, 1997.

[38] Apache Software Foundation. Apache Software Foundation. `http://www.apache.org`.

[39] Apache Software Foundation. Lucene Open-source Search Software. `http://lucene.apache.org`.

[40] Daniela Giorgetti and Fabrizio Sebastiani. Multiclass text categorization for automated survey coding. In *SAC '03: Proceedings of the 2003 ACM symposium on Applied computing*, pages 798–802, New York, NY, USA, 2003. ACM Press.

[41] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2):199–220, 1993.

[42] R. Guha, Rob McCool, and Eric Miller. Semantic Search. In *Proceedings of the* 12^{th} *international conference on World Wide Web*, pages 700–709. ACM Press, 2003.

[43] Clement Yu Zonghuan Wu Hai He, Weiyi Meng. WISE-*i*Extractor: Extracting and Modeling Web Search Interfaces towards Web Database Integration. In *Proceedings of the* 30^{th} *VLDB Conference*, Toronto, Canada, 2004.

[44] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *Neural Networks, IEEE Transactions on*, 13(2):415–425, 2002.

[45] T. Joachims, T. Hofmann, Yisong Yue, and Chun-Nam Yu. Predicting structured objects with support vector machines. *Communications of the ACM, Research Highlight*, 52(11):97–104, November 2009.

[46] Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 137–142, Heidelberg et al., 1998. Springer.

[47] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *ICML '99: Proceedings of the Sixteenth International Conference on Machine Learning*, pages 200–209, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[48] K. Sparck Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: development and comparative experiments. *Inf. Process. Manage.*, 36(6):779–808, 2000.

[49] H. Abolhassani K. Sheykh Esmaili. A categorization scheme for semantic web search engines. In *4th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA-06)*, Sharjah, UAE, March 2006.

[50] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.

[51] S. Knerr, L. Personnaz, and G. Dreyfus. Single-layer learning revisited: A stepwise procedure for building and training a neural network. In *Neurocomputing: Algorithms, architectures and applications*, Berlin, Germany, 1990. Springer-Verlag.

[52] Ora Lassila and Ralph R. Swick. Resource Description Framework (RDF) Model and Syntax Specification, 1999.
`http://www.w3.org/TR/1999/REC-rdf-syntax-19990222`.

[53] Joon Ho Lee, Won Yong Kin, Myoung Ho Kim, and Yoon Joon Lee. On the evaluation of boolean operators in the extended boolean retrieval framework. In *SIGIR '93: Proceedings*

of the 16th annual international ACM SIGIR conference on Research and development in information retrieval, pages 291–297, New York, NY, USA, 1993. ACM Press.

[54] S. Luke and J. Heflin. SHOE: Simple HTML Ontology Extensions Commerce Ontology, Apr 2000.
http://www.cs.umd.edu/projects/plus/SHOE/.

[55] M. E. Maron and J. L. Kuhns. On relevance, probabilistic indexing and information retrieval. *J. ACM*, 7(3):216–244, 1960.

[56] Keio University Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique. JTidy, 1998-2000.
http://jtidy.sourceforge.net.

[57] B. McBride. JENA: Implementing the RDF Model and Syntax Specification. 2001.
http://www-uk.hpl.hp.com/people/bwm/papers/20001221-paper/.

[58] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler. Yale: Rapid prototyping for complex data mining tasks, 2006.

[59] Eyal Oren, Renaud Delbru, Michele Catasta, Richard Cyganiak, and Giovanni Tummarello. Sindice.com: A document-oriented lookup index for open linked data. *International Journal of Metadata, Semantics and Ontologies*, 3:2008.

[60] Brian Pinkerton. Finding what people want: Experiences with the WebCrawler. In *Proceedings of the 2nd International World Wide Web*, volume 18(6), Medford, NJ, USA, 1994. Learned Information.

[61] M J D Powell. The theory of radial basis functions approximation. In *in Advances of Numerical Analysis*, pages 105–210. Press, 1992.

[62] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.

[63] Gerard Salton and Chris Buckley. Term weighting approaches in automatic text retrieval. Technical report, Ithaca, NY, USA, 1987.

[64] Gerard Salton, Edward A. Fox, and Harry Wu. Extended boolean information retrieval. *Commun. ACM*, 26(11):1022–1036, 1983.

[65] B. Schwartz, A. Ward, J. Monterosso, S. Lyubomirsky, K. White, and D. R. Lehman. Maximizing versus satisficing: happiness is a matter of choice. *J Pers Soc Psychol*, 83(5):1178–1197, November 2002.

[66] Amit Singhal. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4):35–43, 2001.

[67] Peter Spyns, Daniel Oberle, Raphael Volz, Jijuan Zheng, Mustafa Jarrar, York Sure, Rudi Studer, and Robert Meersman. Ontoweb - a semantic web community portal. In *PAKM '02: Proceedings of the 4th International Conference on Practical Aspects of Knowledge Management*, pages 189–200, London, UK, 2002. Springer-Verlag.

[68] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res.*, 2:45–66, 2002.

[69] Giovanni Tummarello, Renaud Delbru, and Eyal Oren. Sindice.com: Weaving the open linked data. In *In Proceedings of the International Semantic Web Conference (ISWC)*, 2007.

[70] H. Turtle and W. B. Croft. Inference networks for document retrieval. In *SIGIR '90: Proceedings of the 13th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 1–24, New York, NY, USA, 1990. ACM Press.

[71] Kostas Tzeras and Stephan Hartmann. Automatic indexing based on bayesian inference networks. In *SIGIR '93: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 22–35, New York, NY, USA, 1993. ACM Press.

[72] W3C. OWL Web Ontology Language.
`http://www.w3.org/TR/owl-features/`.

[73] W3C. World Wide Web Consortium.
`http://www.w3.org/`.

[74] W3C. XML Schema.
`http://www.w3.org/XML/Schema`.

[75] W3C. XPATH.
`http://xpath.w3c.org`.

[76] Yun Wan, Satya Menon, and Arkalgud Ramaprasad. A classification of product comparison agents. In *ICEC '03: Proceedings of the 5th international conference on Electronic commerce*, pages 498–504, New York, NY, USA, 2003. ACM Press.

[77] Yun Wan, Satya Menon, and Arkalgud Ramaprasad. How it happens: a conceptual exploration of choice overload in online decision-making by individuals. In *AMSIS 2003: The 9TH Americas Conference on Information Systems*, 2003.

[78] Erik D. Wiener, Jan O. Pedersen, and Andreas S. Weigend. A neural network approach to topic spotting. In *Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval*, pages 317–332, Las Vegas, US, 1995.

[79] Yi Zhang, Wamberto Vasconcelos, and Derek Sleeman. Ontosearch: An ontology search engine. In *Proceedings The Twenty-fourth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence (AI-2004)*, Cambridge, UK, 2004.

Apéndice A

Códigos Fuentes de los Principales Programas

YOSH se compone de una aplicación Web y varios procesos que se ejecutan en forma batch, agendados para que se ejecuten cada cierto tiempo. En esta sección se presenta la URL en la que se pueden encontrar todos los fuentes asonciados a la aplicación y sus procesos.

`http://www.dcc.uchile.cl/~abilbao/tesis/fuentes`