UN AGENTE PARA CLASIFICACIÓN Y FILTRADO DE PÁGINAS WEB

Trabajo de Grado Presentado

por

SERGIO ALEJANDRO GÓMEZ

Enviado a la Facultad de Informática de la Universidad Nacional de La Plata como requerimiento parcial para obtener el título de LICENCIADO EN INFORMÁTICA.

Director: Prof. Dr. Guillermo R. Simari
Co-Director: Prof. Lic. Laura Lanzarini

Jurados:

- Prof. Lic. Patricia Bazán
- Prof. Mag. Silvia Gordillo
- Prof. Ing. Luis Marrone

La Plata, marzo de 2001.

Agradecimientos

Por su inestimable apoyo quiero agradecer a las siguientes personas:

Ante todo, debo un profundo agradecimiento a mi co-directora, Laura Lanzarini, por su guía y apoyo. Luego, también quiero agradecer a mi director, Guillermo Simari.

Muy importantemente, quiero agradecer a mi novia, María Florencia Cittadini, sin cuya motivación no habría llegado hasta aquí.

También quiero agradecer a mis padres, Carlos y Marta, por su apoyo todos estos años. Asimismo, a mis hermanos, Martín y Matías.

Además, debo mi gratitud a las siguientes personas: Claudia Banchoff, Victoria Fernández, Matías Leibovich, Fernando Lyardet, Ezequiel Marozzi, Darío Pérez, Paula Venosa.

Resumen

Esta disertación introduce un paradigma de clasificación de textos en formato HTML basado en la aplicación del modelo de redes neuronales llamado Teoría de la Resonancia Adaptativa Difusa. También, se describe la implementación concreta de un agente de filtrado de páginas HTML llamado *Querando!*, el cual aprende el perfil de gustos de información de un usuario dado.

Se realizaron diversas mediciones para determinar la representación más adecuada de los documentos HTML a procesar así como la de la topología final de la red neuronal a utilizar. Aquí, se enumeran, describen, analizan y comparan estas mediciones.

Además, para tener un sentido de completitud de los temas estudiados se analizan la literatura científica en los temas concernientes a este trabajo, los cuales se comparan con lo realizado. Dichos temas son: el área de recuperación y filtrado de la información, algoritmos de clasificación y clustering, los modelos de redes neuronales y el área de los agentes inteligentes que es donde se aplican todos estos conceptos.

También, se analiza lo concerniente a los detalles más relevantes de la implementación del agente Querando haciendo comparaciones con otras implementaciones posibles.

Índice General

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

En este capítulo discuto las motivaciones que me llevaron al abordaje del problema de la clasificación de texto HTML, los objetivos perseguidos en la realización de este trabajo de grado, asimismo se describe la estructura de esta disertación.

1.1 Motivación

Vivimos en la *era de la información*.

La cantidad de información producida por el hombre es inmensa. Primero, este punto se evidencia con la cantidad de información publicada en prensa escrita en estos últimos años, ya sea en la forma de periódicos, revistas, libros, *journals* científicos y *proceedings* de congresos. Segundo, otra evidencia es el paso de los soportes impresos a los soportes en computadora (ya sean magnéticos –como la cinta y los discos– u ópticos –como los microfilmes o los CD-ROMs); estos nuevos medios, al derribar las cotas impuestas por el aumento de espacio requerido y el costo de impresión creciente, incrementan aún más el efecto de sobrecarga de información[1]. Tercero, se suma una nueva fuente de distribución de información como *Internet*; la cantidad de texto en la forma de correo electrónico que circula por el mundo es inmensa, además de los archivos de textos accesibles en la forma de tranferencia remota por FTP[2]. Cuarto, la aparición de la *World Wide Web*, esa hija amigable de Internet aporta una cantidad inimaginable de información en la forma de hipertexto[3] que, lejos de mantenerse estable, crece día a día en forma incansable.

Desde cualquier punto de vista, la cantidad de información producida es mayor que la habilidad de los usuarios de encontrarla y analizarla. En el caso de un único documento, por ejemplo, un reporte técnico *bajable* de la Web en un área cualquiera, se puede ver que tiene un tamaño típico de uno o dos megabytes, lo cual equivale aproximadamente a mil páginas impresas. Cómo abordar entonces el acceso a una biblioteca entera en un tema dado.

Surge así la necesidad de acceder a dicha información y, en particular, a sólo la parte relevante de ésta. Es decir, el objetivo será solamente acceder a aquellos documentos relacionados con nuestra necesidad concreta de información. En el caso de las bases de datos, este problema está estandarizado (mediante consultas SQL), también en el caso

[1] *Information glut.*

[2] *File Transfer Protocol*: Protocolo de Transferencia de Archivos.

[3] Notése que se considera, por ahora, la disponibilidad de fotos, grabaciones de películas ni de sonidos.

de grandes cantidades de datos numéricos (mediante el uso de técnicas de visualización gráfica). Por otra parte, en el caso del texto en lenguaje natural la situación es más difícil. La complejidad surge de la relación difusa entre la forma de los documentos (secuencias de caracteres) y su contenido semántico.

En el caso del texto HTML, tenemos una mejora de la situación descripta ya que el texto se halla formateado. Tendremos títulos, encabezados de distintas importancias, meta información (en la forma de meta tags), párrafos, texto en itálica, etc. Esto será un aporte importante en la solución del problema ya que el texto dentro de una página se halla destacado de alguna manera de acuerdo a su importancia. También, por la naturaleza del hipertexto, tendremos documentos enlazados. Así, es de esperar que documentos relevantes estén enlazados entre sí, y lo mismo con los documentos irrelevantes.

1.2 Objetivos de este Trabajo de Grado

El objetivo de este trabajo de grado está relacionado con la gran cantidad de información existente en la World Wide Web y la dificultad de los usuarios para hallar allí información relevante a sus necesidades.

La investigación en el área de agentes de software pretende reducir la sobrecarga de información. Dichos sistemas reducen la cantidad de información que le llega a un usuario filtrando la misma.

A la hora de buscar información en la Web, lo más sofisticado que puede hacer un usuario es escribir un patrón de búsqueda en un buscador como Altavista[4] o Lycos[5] y anotar cuáles son sus páginas favoritas en un "bookmark". Aunque últimamente han aparecido metabuscadores como el *Copernic*.

El objetivo de esta tesis es estudiar los tipos de agentes en términos de artículos de investigación para describir los aspectos claves de la tecnología de agentes, así como también estudiar las técnicas de recuperación de información y la aplicación de modelos de redes neuronales. Además, este trabajo incluye el desarrollo de un agente prototipo para hacer filtrado y búsqueda *inteligente* de páginas en la World Wide Web. Dicho agente ha sido bautizado con el nombre *Querando!*.

En el prototipo construido, el trabajo del agente es aprender la manera en que el usuario elige páginas de una serie de resultados de búsqueda y adquirir un perfil de los intereses de dicho usuario. Este perfil de usuario puede usarse para sugerir qué otros enlaces deben explorarse.

1.3 Estructura de Esta Disertación

Esta disertación está estructurada con un enfoque incremental hacia los temas necesitados para comprender el trabajo en su conjunto. Pretender la escritura de un informe auto-contenido es utópico dada la cantidad de literatura existente en el tema. Sin embargo, los temas se presentan en el orden adecuado para que el lector pueda ir fundando los temas a medida que se avance en su lectura. Debido a la interdependencia de los temas,

[4] `http://www.altavista.com`.

[5] `http://www.lycos.com`.

las referencias cruzadas hacia capítulos posteriores son inevitables. Además, junto con el *background* de cada tema se detallan los detalles de implementaciones propias.

Los temas se presentan en el siguiente orden:

1. Primero, se hace una introducción al tema de agentes inteligentes.
2. Luego, se expone el background del área de recuperación y filtrado de información. Se explica también el método para la obtención de la representación de los documentos HTML a clasificar mediante la red neuronal.
3. Posteriormente, se describen los algoritmos de clasificación y *clustering* así como las mediciones realizadas para determinar el algoritmo de clasificación a usar así como las comparaciones necesarias.
4. Luego, se describen las tecnologías de red en Internet necesarias para comprender la implementación del agente *Querando!*.
5. Se describen la interacción con los motores de búsqueda de la web y los trabajos relacionados.
6. Se describe la implementación del agente *Querando!* el cual está basado en la Teoría de la Resonancia Adaptativa Difusa.
7. Finalmente, se enumeran las conclusiones de este trabajo y las posibilidades de su continuación en el futuro.

1.4 Resumen

En este capítulo se expusieron la motivación y objetivos de este trabajo de grado. También, se expuso el orden en que están estructurados los temas de esta disertación.

Capítulo 2

Agentes de la Información

El contenido de este capítulo consta de los siguientes ítems. Primero, una motivación del porqué de la necesidad del estudio y desarrollo de la tecnología de agentes; segundo, una taxonomía de aplicaciones de agentes.

2.1 Motivación del Estudio de Agentes

Las motivaciones para el estudio y desarrollo de las tecnologías de agentes son las siguientes.

Entre las motivaciones académicas halladas en la literatura, una de las más acertadas es la de los editores Michael Huhns y Munindar Singh [Huhns97a] la cual dice así:

> *La proliferación de la computadoras, las redes, el deseo de todo el mundo de estar interconectado y la necesidad de acceso a los datos en todo momento y lugar, ha hecho que los ambientes de información se volvieran grandes, complejos, abiertos y heterogéneos. Los ambientes de información están compuestos por componentes de legado, distribuidos y autónomos. Por sistemas abiertos, entenderemos que las fuentes de información son autónomas y pueden ser agregadas y removidas dinámicamente.*
>
> *El enfoque de agentes tiene como objetivo lidiar con las características descriptas anteriormente. Los agentes representarán los componentes en las interacciones, su función será la de mediar en las diferencias y de proveer un* middleware[1] *consistente, tanto sintáctica como semánticamente. Otra visión de agentes es la de interfaces activas (o asistentes personales, PDAs) para lidiar con el dinamismo y complejidad de los ambientes de información. Las aplicaciones de la tecnología de agentes serán: acceso a la información, comercio electrónico, manufactura inteligente, educación y entretenimiento. Las necesidades comunes de dichas aplicaciones son: publicitar, hallar, combinar, usar, presentar y actualizar información. Dichas funciones requieren tanto mecanismos flexibles como extensibles. Los agentes se pueden ver como una solución porque se pueden construir localmente para cada recurso y son modulares, todo esto bajo la suposición de que satisfagan un protocolo de interacción común.*

[1] *Middleware*: Software de nivel intermedio. Un ejemplo de este tipo de software es ODBC [Gulbransen98, p. 253].

Otra de las motivaciones puede hallarse en el gran número de documentos en formato electrónico y la necesidad de los usuarios de acceder a dicha información. En particular, es mucho más importante acceder sólamente a la parte relevante de dicho monto de información. Otro agravante de este problema es que la cantidad de documentos publicados crece día a día.

2.2 Definición de Agente

En esta sección se abordará el problema de definir lo que es un *agente de software*. Para comenzar, se abordará el asunto desde un punto de vista general; el diccionario Sapiens nos comunica que la etimología de la palabra agente viene del latín *"*agens, -entis*, participio activo de* agere*, hacer"*; además, entre sus acepciones podemos hallar la que dice que un agente es una *"Persona o cosa que produce un efecto"* y/o una *"Persona que obra con poder de otro"* [Sopena76, pág. 82].

En general, la comunidad de las ciencias de la computación ha adoptado estas últimas acepciones para su metáfora de agente; salvo que el sujeto de las acciones ya no será una persona, sino uno o varios programas de computadora. Sin embargo, podemos encontrar sutiles variantes en la interpretación de esta afirmación general.

Michael Genesereth [Genesereth97], en su enfoque al desarrollo de software, ve a los programas de aplicación como agentes de software, es decir, *"componentes de software que se comunican con sus pares*[2] *por medio del intercambio de mensajes en lenguaje expresivo de comunicación entre agentes"*; el cual, en su caso, serán performativas en KQML con contenido en KIF [Genesereth92, Labrow98]. También, hace una comparación del paradigma de agentes con el de la programación orientada a objetos, diciendo *"... como un* objeto*, un agente provee una interfaz basada en el envío de mensajes independiente de sus estructuras de datos y algoritmos internos. La diferencia primaria entre los dos enfoques radica en el lenguaje de la interfaz. En la programación orientada a objetos en general, el significado de un mensaje puede variar de un objeto a otro*[3]*. En una ingeniería de software basada en agentes, los agentes usan un lenguaje común con semántica independiente."*

En [Harper96], Harper, pareciéndose mucho a la definición de Sapiens, nos cuenta que la definición más simple de agente es *"una entidad de software que asiste a la gente y actúa a su favor"*. Por lo tanto, esto permite a los usuarios delegarles funciones a los agentes que, de otra manera, deberían llevar a cabo ellos mismos; a la vez, los agentes pueden automatizar, simplificar, aprender y recomendar maneras de hallar, sin complejidad, las respuestas correctas a sus problemas. Obviamente, hablamos de problemas de manejo de información; sin embargo, el interesado en la solución de problemas de índole psicológica puede leer sobre el *chatterbot* Eliza, programa que hace las veces de terapista, en [Cheong96, págs. 251–253,278] o experimentar una charla en inglés con ella usando el programa ELIZA.GS de los ejemplos del intérprete 2.30b del lenguaje de programación funcional Gofer [Jones95].

Michael Huhns y Munindar Singh, en [Huhns97a, pág. 1], sintetizan la definición de agente en *"Los agentes son componentes de software activas y persistentes que perciben, actúan y se comunican."* Además, a veces, se pide que sean autónomos, *goal-directed*,

[2] En el texto original se lee *peers*.

[3] Aquí se entiende objeto como tipo o clase.

reactivos o programados declarativamente. Hay dos escuelas extremas: la que ve a los agentes como *entidades conscientes, cognitivas, que tienen percepciones, sentimientos y emociones como los humanos* y otra que dice que son *autómatas que se comportan exactamente como fueron programados*. La primera pide demasiado y tiene el problema de que dichas teorías no sirven para aplicarse en la construcción de agentes (a pesar de que sus definiciones se asemejen a lo que entiende el vulgo por un agente, un ejemplo sería *Data*[4], el personaje de Brent Spiner en *Startrek, The Next Generation*); mientras que la segunda es muy permisiva y no representa las abstracciones que describen, analizan, entienden o explican la conducta de los agentes. En virtud de que nuestra tarea como científicos de la ciencia de la computación es construir modelos para explicar el mundo del software y de los procesos en general, la pista que nos dan estos autores es preguntarnos a nosotros mismos qué ganamos, al llamar a una entidad un *agente*, para poder aumentar su entendimiento.

Para Mark Cutkosky *et al.*, en el contexto de la descripción de su sistema PACT [Cutkosky97], un agente es un *"programa de computadora que se comunica con programas externos via un protocolo predefinido. Un agente es capaz de responder todos los mensajes definidos por el protocolo, y usa el protocolo para invocar los servicios de otros agentes."*. Este científico se halla muy influenciado por la visión de Genesereth.

Otro punto a tener en cuenta es la implementación de dichos agentes, o, dicho de otra manera, las otras dimensiones en que pueden verse. Por ejemplo, en [Huhns97b], los agentes serán implementados como sistemas expertos cuya área de experiencia[5] es el procesamiento de órdenes para telecomunicaciones.

En el caso del artículo de E. Durfee *et al.*, al describir la arquitectura de agentes de la Biblioteca Digital de la Universidad de Michigan [Durfee97], decide llamar agentes a los módulos de software que componen dicho sistema porque cada uno es capaz de operar por su cuenta y mantener sus propios objetivos y preferencias. Estos autores, a semejanza de Huhns, toman una postura conductivista donde deciden llamar agente a todo aquello que convenga llamarse así. En su caso, los agentes son implementados por procesos Unix que se comunican usando protocolos comunes de red, como por ejemplo, TCP/IP 8.3.2; además, para la comunicación entre agentes, usan CORBA [Weber97, p. 781]. Estos agentes también representan a usuarios; por ejemplo, un agente puede representar los intereses de un autor mientras que otro los de un lector.

Para Pattie Maes *et al.* [Lashkari97], ya en el ámbito de las interfaces inteligentes, un *agente de interfaz con aprendizaje*[6] es *un programa de computadora que emplea técnicas de* machine learning *para asistir a un usuario con una aplicación particular de computadora.*

Para Fah-Chun Cheong [Cheong96, pp. 5–6], los agentes son asistentes de software personales con autoridad delegada por sus usuarios, que hacen por otra persona aquello que dicha persona podría hacer por si misma. Además, cita a Nicholas Negroponte y Alan Kay acerca de usar agentes en la interfaz para delegar tareas.

Para la gente de la comunidad de sistemas operativos, la metáfora de agente es usada para describir procesos que migran a través de redes de computadoras; por ejemplo, para Dag Johansen *et al.* [Johansen98], los agentes serán *procesos que migran a través de una red*.

Un caso más bien técnico es el caso en que el término *agente de usuario* se refiere

[4] *Data* es un androide inteligente con rango de comandante en una nave interestelar.

[5] Expertise.

[6] Learning interface agent.

al programa cliente que está más cerca del usuario e inicia requerimientos a favor de un usuario [Cheong96, p. 127]; un ejemplo de este caso son los *browsers* y clientes de mail (como Eudora).

Por último, en la literatura, dependiendo de la escuela del autor, este tema se conoce también como *Agentes inteligentes* (escuelas de inteligencia artificial) o *Agentes de la Información* (escuelas de interfaces).

2.3 Taxonomías de Agentes

Según [Cheong96], las dimensiones en las que podemos estudiar a los agentes son las siguientes: delegación, coordinación, creatividad, conocimiento, emoción, programación, sociedad, y comunicación. Cheong [Cheong96] plantea una clasificación de los agentes desde el punto de vista de estas dimensiones.

Para Bigus [Bigus98], hay tres formas de clasificar a los agentes:

- En el contexto de *Agency*, Inteligencia y Movilidad;
- Por la estrategia primaria de procesamiento del agente;
- Por la función que desarrolla el agente.

Otra caracterización posible viene dada en base a si el agente trabaja solo o colaborando con otros agentes en un sistema multiagente.

2.3.1 Agencia, Inteligencia y Movilidad

Hay tres dimensiones o ejes para medir las tres capacidades de un agente: agencia, inteligencia y movilidad.

Agencia

Agency tiene que ver con el grado de autonomía que el agente de software posee para representar al usuario frente a otros agentes, aplicaciones y sistemas de computadoras. Un agente representa, ayuda y guía al usuario, y, en algunos casos, toma decisiones unilaterales a favor del mismo [Bigus98]. Para Marvin Minsky, el concepto de agencia está ligado a la capacidad de interacción (con otros agentes) para el logro de un objetivo; es decir, *agency* es equivalente a sociedad [Riecken94].

Inteligencia

La inteligencia se refiere a la habilidad del agente para capturar y aplicar conocimiento específico del dominio de aplicación como también aplicar procesamiento para resolver problemas. De esta manera, los agentes pueden ser muy simples, con lógica codificada, o complejos, usando métodos basados en IA como inferencia y aprendizaje.

Movilidad

Un agente es móvil si puede moverse entre sistemas en una red. La movilidad introduce complejidad adicional a un agente inteligente, ya que hay que considerar cuestiones de seguridad (tanto del agente mismo como del sistema target) y costo. Las intraredes (emphintranets) son un ambiente favorable para esto ya que permiten que los agentes móviles vaguen porque requiere menos seguridad que la Internet en general.

2.3.2 Por Estrategia de Procesamiento

De acuerdo a esta clasificación, hay tres tipos de agentes: reactivos, deliberativos o *goal-directed* y/o colaborativos.

Agentes Reactivos

Los más simples son los reactivos, que responden al modelo evento-condición-acción. Este tipo de agente no tiene modelo interno del mundo, responden sólamente a estímulos externos y a la información disponible del ambiente usando sensores. Como las redes neuronales, los agentes reactivos exhiben conductas emergentes que son el resultado de la interacción de agentes simples. Cuando estos agentes interactúan, intercambian datos de bajo nivel (datos de sensores físicos, por ejemplo) y no datos simbólicos de alto nivel.

Agentes Deliberativos

Los agentes deliberativos tienen conocimiento del dominio y capacidad de planeamiento necesaria para llevar a cabo una secuencia de acciones para alcanzar un objetivo específico, para lo cual pueden colaborar con otros agentes. En la práctica pueden usar cualquiera de las técnicas de IA de los últimos cuarenta años.

Agentes Colaborativos

Los agentes colaborativos trabajan juntos para resolver problemas. La comunicación entre agentes es muy importante porque mientras que cada agente individual es autónomo, la sinergía resultante de su cooperación es lo que los hace interesantes. Pueden resolver problemas que van más allá del alcance de cada agente simple y permiten un enfoque modular basado en la especialización de las funciones de los agentes o del conocimiento del dominio. Por ejemplo, agentes colaborativos pueden trabajar como asistentes de diseño en un proyecto de ingeniería. Cada agente puede verificar aspectos diferentes del diseño y usar su conocimiento experto conjunto para determinar si el diseño como un todo es consistente.

2.3.3 Por Funciones de Procesamiento

La manera más natural de pensar acerca de los diferentes tipos de agentes es basado en la función que realizan. Así tenemos: agentes de interfaz y agentes de información.

Los *agentes de interfaz* trabajan asistentes personales para ayudar a un usuario a realizar sus tareas. Los agentes de interfaz generalmente emplean aprendizaje para adaptarse a los hábitos de trabajo y preferencias del usuario.

Otra clase genérica de agentes son los *agentes de información*. Estos agentes implementan interfaces especializadas a fuentes de información hetrerogéneas, pueden también filtrar información en la web, en correo electrónico o en grupos de noticias. De cualquier manera, los agentes de información tratan de resolver el problema de conseguir la información correcta en el momento adecuado [Bigus98]. La sobrecarga de información es quizá la motivación más grande a la hora de tratar de desarrollar la tecnología de agentes.

2.3.4 Agentes Individuales versus Sistemas Multiagente

Los agentes pueden funcionar en aislamiento o no. Cuando los agentes no funcionan aislados se dice que funcionan como parte de un *sistema multiagente* [Aarsten96, Gómez99, Huhns97a, Huhns97b, Singh95]. La motivación de ello es enteramente práctica [Huhns97a].

La WWW puede verse como una red de información donde los agentes representan usuarios, servicios y recursos. Un paradima típico de uso es como sigue. El agente de recursos pone anuncios a los servicios; el agente de usuario usa los servicios para encontrar agentes de recursos y entonces consultarlos por la información que necesita.

Según [Huhns97a], un agente de la web puede hacer bien su trabajo sólo si toma ventaja no sólo de las fuentes de información de la web sino también de los otros agentes que podrían estar operando allí. Además, agentes representando a usuarios diferentes podrían colaborar encontrando y combinando información.

En el caso de *Querando!* esto se podría dar compartiendo perfiles de usuario, lo que también se podría ver como un caso particular del trabajo de [Lashkari97] (sección 9.3.1).

Otro caso que se puede dar es que los agentes compitan por bienes y recursos, por ejemplo, en el caso de agentes para hacer compras (recabar información acerca de ítems de consumo –por ejemplo, el caso del *BargainFinder* [Cheong96]) en forma automática en la web.

La comunicación entre los agentes se puede dar usando imperativas KQML con contenido representado en KIF [Genesereth97, Genesereth92, Genesereth94a, Labrow98].

2.4 Características Claves de los Agentes

En esta sección se da una taxonomía de las caracteríticas de los agentes vistos desde varios enfoques: como entidades separadas (características intrínsecas), en su relación con otros agentes (extrínsecas), del sistema, del "framework" de desarrollo y del ambiente.

Podemos hallar las siguientes clases de características [Huhns97a, p. 2–3]:

- Características Intrínsecas
  - *Lifespan*: Transient versus long-lived.
  - *Nivel de cognición*: Reactivo a deliberativo.
  - *Construcción*: Declarativo a procedural.
  - *Movilidad*: Estacionario a itinerante.
  - *Adaptabilidad*: Fija a enseñable a autodidacta.
  - *Modelado*: Del ambiente, de ellos mismos o de otros agentes.
- Características Extrínsecas

  - *Localidad*: Local a remota.
  - *Autonomía social*: Independiente a controlada.
  - *Sociabilidad*: Autista, responsable, jugador de equipo.
  - *Amigabilidad*: Cooperativo a competitivo a antagonista.
  - *Interacciones*: Via facilitadores, mediadores o no-agentes.

- Características del Sistema
  - *Unicidad*: Homogéneo a heterogéneo.
  - *Granularidad*: Grano fino o grueso.
  - *Estructura de control*: Jerarquía a democracia.
  - *Autonomía de interfaz*: Vocabulario específico, lenguaje, protocolo, ontologías, creencias.

- Características del *Framework*
  - *Autonomía de diseño*: Plataforma/lenguaje/interna. Arquitectura/protocolo de interacción.
  - *Infraestructura de comunicación*: Memoria compartida o basada en mensajes. Push o pull. Sincrónico o asincrónico.
  - *Protocolo de mensajes*: KQML, HTTP y HTML, OLE, CORBA o DCOM.
  - *Servicios de mediación*: Basados en ontología y/o transaccionales.
  - *Servicios de seguridad*: Timestamps/autenticación.
  - *Soporte de operaciones*: Archivos/redundancia/restauración, accounting.

- Características del Ambiente del Agente
  - *Conocible*: Con qué extensión el ambiente es conocido por el agente?
  - *Predecible*: Con qué extensión puede el agente predecir el ambiente?
  - *Controlable*: Con qué extensión puede el agente modificar el ambiente?
  - *Histórico*: Los estados futuros dependen de la historia completa o sólo del estado actual?
  - *Teleológico*: El ambiente posee partes útiles (es decir, hay otros agentes)?
  - *En tiempo real*: Mientras el agente está deliberando, puede cambiar el ambiente?

2.5 Arquitecturas de Agentes

Las arquitecturas proveen los *frameworks* organizacionales dentro de los cuales pueden diseñarse y construirse los agentes. Por otro lado, la infraestructura provee los servicios disponibles a medida que operan.

Podemos encontrar los siguientes tipos de arquitecturas:

- Basadas en agentes para sistemas de información (basadas en mediadores) [Aarsten96, Wiederhold97]. El enfoque de arquitecturas de agentes mediadores[7] pretende resolver el problema del acceso a fuentes de información heterógenea. De esta manera, se interpone un agente con una interfaz homogénea entre el usuario (u otro agente haciendo prospección de información) y el recurso heterogéneo [Bayardo98, Cohen96a].

- Agentes individuales (que además pueden interactuar con otros agentes). Por ejemplo, arquitecturas BDI (basadas en creencias, deseos e interracciones) [Fischer97, Bates97, Maes95].

- Agentes *broad* (agentes con capacidades integradas): Son agentes que actúan como los humanos. Pueden ser guiados por emociones (*emotion-directed control*) ó por objetivos (*goal-directed control*) [Bayardo98].

- Arquitecturas abiertas: Se buscan una estandarización y formalización de los componentes de las infraestructuras disponibles en un ambiente [Cohen96a, Bayardo98]. Por ejemplo, KIF y KQML [Genesereth97, Genesereth92, Genesereth94a, Labrow98].

2.6 Resumen

En este capítulo, se definieron a los agentes. La definición más relevante quizá es aquella que los ve como componentes de software con una función delegada por un usuario. Los agentes pueden trabajar en forma aislada o interactuando con otros agentes. Se analizaron también las características claves de los mismos y se dio una taxonomía para clasificarlos.

[7] Otros nombres para *mediadores* son *facilitadores* y/o *brokers* [Labrow98].

Capítulo 3

Recuperación de Información

En este capítulo se exponen los conceptos relacionados con el área de la Recuperación de Información[1]. Dichos conceptos servirán de fundación para entender algunas de las aplicaciones de agentes de filtrado de información estudiados junto con aspectos de implementación del agente *Querando!*.

Los puntos a tratar serán los siguientes: la definición de la disciplina, los métodos de representación de documentos y de consultas, las estructuras de datos asociadas con la disciplina, los métodos de stemming y de filtrado por listas de *stop words*. Finalmente, también se hace una comparación entre la recuperación de información y otras disciplinas relacionadas.

3.1 Definición de IR

La recuperación de información (IR) es la disciplina que estudia el almacenamiento y recuperación de documentos; su objetivo es la implementación de sistemas computacionales que permitan el almacenamiento de grandes cantidades de documentos en una base de documentos, en una manera tal que permita la recuperación de documentos relevantes a las necesidades de información de los usuarios.

De acuerdo a Mooers [Sebastiani98],

> *"La recuperación de información es el nombre del proceso o método por el cual un usuario prospectivo de información es capaz de convertir su necesidad de información en una lista real de citas a documentos útiles para él. La recuperación de información comprende los aspectos intelectuales de descripción de la información y su especificación para búsqueda, así como también cualquier sistema, técnica y máquinas que fueran usadas para llevar a cabo la operación".*

En IR, se entiende por *documento cualquier expresión en formato libre con un contenido de información*. La característica distintiva de los sistemas de IR es que proveen acceso a datos para los cuales no tenemos un modelo semántico definido. El texto es el ejemplo más notable. Sin embargo, también se han investigado sistemas de IR para imágenes, audio y video (multimedia IR)[Lewis95].

El término *grandes bases de documentos* se refiere a tecnología de CD-ROM o a sistemas distribuidos (como la World Wide Web), las cuales son capaces de almacenar del

[1] *Information Retrieval* en inglés.

orden de 10^5 a 10^7 documentos. Las bases de documentos pueden ser estáticas (CD-ROM) o dinámicas (bibliotecas digitales o la WWW), centralizadas o distribuidas. Una *necesidad de información* será un deseo (posiblemente especificado en una manera imprecisa) de información útil a la solución de un problema. Por *relevante*, entenderemos útil, de acuerdo a un criterio subjetivo del usuario.

Los sistemas modernos pueden operar tanto en forma *off-line* como en forma *on-line*:

- *Operación* off-line es cuando los documentos se adquieren de una base de documentos.
- *Operación* on-line es cuando se trabaja bajo el siguiente paradigma:
  1. Se lee un requerimiento del usuario y se le retornan aquellos documentos que el sistema estima relevantes a las necesidades de información expresadas por dicho requerimiento, en alguna de las siguientes formas:
     - un conjunto de documentos considerados relevantes;
     - una lista de documentos calificados (*ranked*) en orden decreciente.
  2. Se obtiene el *feedback* del usuario para realizar otra pasada mejorada de recuperación.

También, cabe acotar que los usuarios deberán tolerar una salida imperfecta de parte de un sistema de IR.

Además, la IR es muy difícil por la indeterminación de la relevancia debido a que: el sistema podría malinterpretar el significado de los documentos y/o del requerimiento, el usuario no podría saber con precisión qué es lo que quiere (necesidad de información vaga), ó no podría estar claro el grado de relevancia que desea el usuario (*recall-oriented versus precision-oriented*) [Sebastiani98].

3.2 Representación de Documentos y de Necesidades de Información

En IR, la búsqueda de strings, aproximada o exacta, es inapropiada porque el *matching on-line* de strings es computacionalmente caro[2]; y la ocurrencia de un string x en un documento d no es condición necesaria ni suficiente de la relevancia de x en d. Por ejemplo, un documento mencionando *radiactive dumps* (*desechos radioactivos*) probablemente sea relevante al requerimiento *nuclear waste disposal* (*basura nuclear*), por otro lado, un documento mencionando *lead* (plomo) no sea relevante al requerimiento *lead* (liderar) [Sebastiani98].

3.2.1 Indexación

Entonces, para hacer más eficiente la recuperación de documentos relevantes es usual producir *representaciones internas* (IREP) de los documentos –en forma *off-line*– y de

[2]La búsqueda de strings es la forma más trivial de recuperación de información. Dado un conjunto de documentos de texto "chatos" y una palabra que supuestamente representa nuestra necesidad de información, la recuperación consiste en encontrar aquellos documentos que contienen dicha palabra.

los requerimientos – en forma *on-line*–, y determinar su *match* (*parecido*) en tiempo de recuperación. El proceso de producir IREPs de los documentos y los requerimientos se llama *indexación*. La IREP de un requerimiento se llama *query* (*consulta*), y la indexación de un requerimiento se llama *adquisición de un query*.

El proceso de indexación típicamente genera un conjunto de términos de indexación (características ó *features*), posiblemente pesados, como las IREPs de los documentos. La hipótesis subyacente es que el significado combinado de este conjunto aproxime el significado del documento o del requerimiento. En IR de texto, los términos de índice pueden ser:

- palabras de un vocabulario controlado (por ej. *categorization*);
- palabras (por ej. *classification*) automáticamente extraídas del documento;
- raíces (*stems*) de palabras (ej.: *class-*) automáticamente extraídas del documento. Es la opción más frecuente y más efectiva;
- proposiciones subordinadas sustantivas (por ej.: *classification of industrial processes*) extraídas automáticamente del documento. Extrañamente, esta técnica no ha dado mejores resultados que las dos anteriores;
- metadatos (*metadata*), por ej., título, autor, etc.

Los conjuntos de características o IREPS pueden adolecer de los siguientes problemas que pueden hacer difícil a un clasificador por *machine learning* que aprenda correctamente el conjunto de patrones [Lewis92]:

- el conjunto de características no distingue apropiadamente las instancias;
- el conjunto de características resulta de gran dimensionalidad;
- el conjunto de características puede contener ruido
- el conjunto de características puede contener redundancia.

Existen varios métodos para mejorar dichos problemas [Lewis92]:

- *Selección de características*: Dado un conjunto de D características iniciales, la técnica consiste en elegir $d < D$ nuevas características. Es de notar que el proceso de hallar las óptimas d nuevas características implica recorrer un espacio de búsqueda de tamaño $\frac{D!}{(D-d)!d!}$.
- *Extracción de características*: La extracción de características sirve para eliminar características de baja calidad.

La investigación [Frakes92a, Sebastiani98] ha mostrado que el método de indexación usado es, por lejos, más importante que la técnica de *matching* usada. Varios puntos deben tenerse en consideración: Primero, el uso de la misma técnica para documentos y requerimientos tiende a garantizar un proceso correcto de *matching*. Segundo, hay indeterminación en el proceso de indexación, ya que indexadores diferentes (humanos o automáticos) en general no producen la misma IREP para el mismo documento. Tercero,

	d_1	...	...	d_j	...	d_m
t_1	w_{11}	...	...	w_{1j}	...	w_{1m}
...	...	...	...	...	...	...
t_i	w_{i1}	...	...	w_{ij}	...	w_{im}
...	...	...	...	...	...	...
t_n	w_{n1}	...	...	w_{nj}	...	w_{nm}

Figura 3.1: Resultado de la indexación

a diferencia de lo que ocurre en sistemas de recuperación referencial, la disponibilidad *on-line* de los documentos permite el uso del documento entero no sólo para presentación del mismo sino también para indexamiento. Salton[3] dice

> *"la evidencia experimental acumulada en los últimos veinte años indica que los sistemas de indexamiento de texto basados en la asignación de términos simples pesados producen resultados de recuperación superiores que aquellos obtenibles con representaciones del texto más elaboradas. Estos resultados dependen crucialmente en la elección del sistema que asigna los pesos a los términos".*

El resultado de la indexación puede ser visto como una *matriz de incidencia* (figura 3.2.1 (3.1)), donde los elementos de la matriz se llaman *pesos*, y pueden ser *binarios*, es decir $w_{ij} \in \{0, 1\}$, ó, *no-binarios*, es decir $w_{ij}\epsilon[0, 1]$.

Los pesos, para un documento o un requerimiento, pueden ser asignados de dos formas. Por un lado, *manualmente*, por un indexador humano familiar con la disciplina con la que tienen que ver los documentos, la técnica de indexación (por ejemplo, el número óptimo de términos para la IREP, el vocabulario controlado, etc.) y los contenidos de la colección (por ejemplo, la distribución de los tópicos). La otra forma es, *automáticamente*, por procesos de indexación basados en análisis estadísticos de ocurrencia de palabras en los documentos, en los requerimientos y en la colección; con respecto a este punto ver el análisis de los trabajos de Balabanovic y de Pazzani en las secciones 9.3.8 y 9.3.13, respectivamente.

El enfoque automático es el más usado hoy en día en IR de texto porque es barato y más efectivo, mientras que el enfoque manual se usa en IR multimedia, debido a la falta de técnicas automáticas de indexación efectivas para medios no textuales [Sebastiani98].

3.2.2 Medidas de Similitud entre Documentos

Una vez que tenemos la IREP de un documento y la IREP correspondiente a la consulta, es necesario dar una medida del parecido o *match* entre ambas. En la literatura de IR podemos encontrar cuatro medida de similitud entre documentos [Rasmussen92, Sebastiani98]: el producto interno, los coeficientes de Dice, de Jaccard y del coseno. Rasmussen [Rasmussen92] también provee de citas a G. Salton[4] y Anderberg[5] para profundizar en el tema.

[3]Salton, G. *The SMART Retrieval System.* Englewood Cliffs, NJ. Prentice Hall. 1971.
[4]Salton, G. *Automatic Text Processing.* Reading, Mass.: Addison-Wesley. 1989.
[5]Anderberg, M. R. *Cluster Analisys for Applications.* 1973. New York: Academic.

Producto Interno

El *produto interno* para calcular la similitud entre dos documentos D_i y D_j está definido por la ecuación [Sebastiani98]:

$$S_{D_i,D_j} = \sum_{k=1}^{L} (peso_{ik} \cdot peso_{jk}) \tag{3.1}$$

En el caso binario (ver *modelo booleano* en la sección 3.7.2),

$$S_{D_i,D_j} = |D_i \cap D_j| \tag{3.2}$$

El Coeficiente de Dice

El *coeficiente de Dice* para calcular la similitud entre dos documentos D_i y D_j está definido por la ecuación [Rasmussen92]:

$$S_{D_i,D_j} = \frac{2 \sum_{k=1}^{L} (peso_{ik} peso_{jk})}{\sum_{k=1}^{L} peso_{ik}^2 + \sum_{k=1}^{L} peso_{jk}^2} \tag{3.3}$$

Si se usan pesos de términos binarios, el coeficiente de Dice se reduce a:

$$S_{D_i,D_j} = \frac{2C}{A+B} \tag{3.4}$$

donde C es el número de términos que tienen en común D_i y D_j, y A y B son el número de términos en D_i y D_j respectivamente.

El Coeficiente de Jaccard

El *coeficiente de Jaccard* para calcular la similitud entre dos documentos D_i y D_j está definido por la ecuación [Rasmussen92]:

$$S_{D_i,D_j} = \frac{\sum_{k=1}^{L} (peso_{ik} peso_{jk})}{\sum_{k=1}^{L} peso_{ik}^2 + \sum_{k=1}^{L} peso_{jk}^2 - \sum_{k=1}^{L} (peso_{ik} peso_{jk})} \tag{3.5}$$

El Coeficiente del Coseno

El *coeficiente del coseno* para calcular la similitud entre dos documentos D_i y D_j está definido por la ecuación [Rasmussen92]:

$$S_{D_i,D_j} = \frac{\sum_{k=1}^{L} (peso_{ik} peso_{jk})}{\sqrt{\sum_{k=1}^{L} peso_{ik}^2 \sum_{k=1}^{L} peso_{jk}^2}} \tag{3.6}$$

Este coeficiente da una medida del coseno del ángulo que separa a los dos vectores (cuanto más pequeño el ángulo, mayor el coseno).

Comparación entre Coeficientes

Según [Sebastiani98], los coeficientes de Dice, Jaccard y Coseno son versiones normalizadas del producto interno; esto previene que documentos con mayores IREP sean sobre valorados. Además, estas medidas están basadas en consideraciones ad-hoc y no está claro cuál es el mejor.

3.2.3 La Matriz de Similitud

Dada una base de n documentos, la matriz de similitud es una matriz cuadrada S de $n \times n$ donde $S_{i,j} = S_{D_i,D_j}$ [Rasmussen92].

Generalmente, como la relación de medida entre dos documentos es simétrica, alcanza con calcular la matriz triangular inferior de S.

3.3 Relevancia: Parcialidad y Probabilidad

A diferencia de la satisfacción de *queries* en el área de bases de datos, la relación de relevancia R entre documentos D y necesidad de información N no está formalmente definida, pero es *subjetiva* (y por lo tanto inefable), es decir, determinada por el usuario. A diferencia de las bases de datos, y en virtud de la mayor expresividad y ambigüedad del lenguaje natural con respecto a un lenguaje formalizado (por ejemplo, SQL [Korth86]), la efectividad es un asunto muy importante.

3.3.1 Modos de Definición de la Relevancia

La relevancia puede ser definida de varias maneras:

1. Con valores booleanos:

   $R : D \times N \mapsto \{0, 1\}$

2. Con valores finitos:

   $R : D \times N \mapsto \{0, \frac{1}{s}, \frac{2}{s}, \ldots, \frac{s-1}{s}, 1\}$

3. Con valores infinitos:

   $R : D \times N \mapsto Q \cap [0, 1]$

El enfoque 3 es el más plausible, pero no hay acuerdo sobre cual es el más efectivo en costo. El enfoque 1 es el más usado [Sebastiani98]. La relación de relevancia no es conocida por el sistema antes del juicio del usuario: basada en las IREPs de los documentos y de los requerimientos, el sistema sólo puede *"adivinar"*, es decir, calcular el llamado *Retrieve Status Value*[6], $RSV(d_i, r_j)$ que representa:

- la relación de satisfacción $d \models r$ entre la IREP del documento d y el *request* r, por ejemplo en el modelo booleano, $RSV(d_i, r_j) \in \{0, 1\}$;
- la similitud entre la IREP de un documento y un *request*, por ejemplo en el modelo espacial vectorial, $RSV(d_i, r_j) \in [0, 1]$;
- la probabilidad (subjetiva) de la relevancia del documento a la necesidad de información basada en una muestra de juicios de relevancia provista por el usuario; $RSV(d_i, r_j) \in [0, 1]$.

Según [Sebastiani98], los documentos con RSV más alto no son necesariamente los más relevantes.

[6] Valor del Estado de la Recuperación.

3.3.2 Factores que Influencian la Relevancia

La relevancia es una noción inefable, en particular, es [Sebastiani98]:

- *Subjetiva*, ya que dos usuarios pueden hacer el mismo *request* y dar distintos juicios de relevancia sobre un mismo documento recuperado.
- *Dinámica*, ya que puede juzgar relevante a un documento dado en una pasada de recuperación, y después, en otra pasada de recuperación, juzgarlo irrelevante, o viceversa. Además, los documentos recuperados y mostrados al usuario pueden influenciar su juicio de relevancia sobre los documentos que se le fueran a mostrar luego.
- *Multifacética*, ya que no está sólo determinada por su topicalidad (es decir, su afinidad temática), sino también por su credibilidad, especificidad, exhaustividad, precisión, ser o no reciente, claridad, etc.
- *Dependiente del* surrogate[7] *del documento*, es decir de la porción (o representación) del documento en la cual el usuario juzga la relevancia del mismo.

3.4 Evaluación y Experimentación

La evaluación de una técnica de IR, o de un sistema de IR, en general es llevada a cabo experimentalmente, en vez de analíticamente. Los criterios posibles de evaluación son:

- *Efectividad*: la habilidad de separar documentos relevantes de irrelevantes (para sistemas de recuperación binarios), o "rankear" documentos correctamente con respecto a su grado de relevancia (en sistemas de recuperación con *ranking*).
- *Eficiencia*: el tiempo promedio requerido para procesar el query y retornar los documentos juzgados relevantes al usuario.
- *Utilidad*: la efectividad (y/o eficiencia) con respecto al costo (de diseño, desarrollo, mantenimiento, actualización, uso, etc.) pagado por las partes involucradas (por ejemplo, desarrolladores, usuarios, etc.).

Nota: Cabe notar que la primer opción es la más usada, especialmente por la naturaleza poco cuantificable de las otras.

3.5 Efectividad: Precisión y *Recall*

La relevancia se asume binaria, la efectividad es usualmente tomada como una combinación de [Frakes92a, Girardi98a, Sebastiani98]:

[7] *Surrogate*: *Adj.* Una persona o cosa usada o actuando en lugar de otra [Quirk90]. *"Sustituto"* tal vez sea la mejor palabra castellana para su traducción.

- *Precisión*[8]: la probabilidad de que, si un documento es recuperado, sea relevante, es decir:

$$Pr \equiv P(Rel|Rec) \equiv \frac{|Rel \cap Rec|}{|Rec|} \tag{3.7}$$

- *Recall*: la probabilidad de que, si un documento es relevante, sea recuperado, es decir:

$$Re \equiv P(Rec|Rel) \equiv \frac{|Rel \cap Rec|}{|Rel|} \tag{3.8}$$

La intuición de la ecuación 3.7 es que la *precisión* nos dice cuántos documentos son relevantes de todos aquellos que se recuperaron. Por otro lado, el *recall* en la ecuación 3.8 nos dice cuántos documentos se recuperaron de todos aquellos que eran relevantes.

Dado un requerimiento r y un sistema de IR s que opera sobre una colección, entonces $D \equiv Rel \cup \overline{Rel}$:

Si s es un sistema de recuperación binario, la precisión y el *recall* son valores absolutos; y si s es un sistema de recuperación "rankeado", la precisión y el *recall* son valores relativos uno del otro.

Usualmente, la función $Pr = Pr(Re)$ se computa para niveles determinados de *recall* (por ejemplo, $Re = 0.1, 0.2, \ldots, 1.0$) y los valores resultantes se interpolan; esto da un gráfico de precisión *vs. recall* [Frakes92a, pp. 10–11].

La efectividad de un sistema es típicamente evaluada promediando sobre distintos requerimientos, pero como dice Sebastiani [Sebastiani98]: *"cada tipo de promedio, sin embargo, sólo da una vista parcial del problema..."*. Un gráfico típico de precisión versus *recall* tiene las siguientes características: es monótonamente decreciente (a veces hiperbólico); para $Re = 1$, es tangente la línea $Pr = g$, donde $g \equiv \frac{|Rel|}{|D|}$ es la generalidad del *request*, y, un sistema típico de IR no va más allá de precisión 0.4 para un *recall* de 0.4.

3.5.1 Colecciones para *Testing* de la Efectividad

Para comparar la efectividad de diferentes técnicas de IR, se usan colecciones standard *benchmark* de documentos[9], las cuales consisten de conjuntos de documentos o *abstracts* de a lo sumo 10^6 elementos, un conjunto de *requests*, y una matriz de relevancia compilada por expertos (en general los escritores de los *requests* o expertos del dominio) [Sebastiani98].

Cuando se usa una colección de *testing* para comparar diferentes sistemas de IR, compararemos los gráficos obtenidos considerando un sistema s dado yendo a través de los siguientes pasos:

1. Para cada *request* r_i
   - correr s en r_i, (produciendo una lista de documentos rankeados $s(r_i) = \langle d_{i1}, \ldots, d_{im} \rangle$);
   - usar los juicios de relevancia para calcular la precisión $Pr_{s(r_i)}(Re)$ para valores predeterminados de *recall* (*e.g.*, $Re = 0.0, 0.1, \ldots, 0.9, 1.0$).
2. Para cada valor predeterminado de *recall* Re_x, calcular el promedio $\overline{Pr_s(Re_x)} = \frac{\sum_{j=1}^{l} Pr_{s(r_j)}(Re_x)}{l}$ con respecto a los requerimientos $Req = \{r_1, \ldots, r_l\}$.

[8] La ecuación $P(x|y)$ se lee como *la probabilidad de que ocurra el evento x dado que haya ocurrido el evento y* [Devore95].

[9] Por ejemplo, las colecciones TREC.

3. Interpolar los puntos así obtenidos, obteniendo el gráfico $\overline{Pr_s(Re_x)}$ que representa la efectividad promedio del sistema s en la colección dada.

Debe notarse que el problema de la experimentación por medio de colecciones de tamaño realista no es su uso sino su construcción [Sebastiani98].

En las pruebas de *Querando!* no se usaron colecciones standard, simplemente se usaron páginas de la web (capítulos 4 y 6).

3.6 Feedback de Relevancia y Expansión/Reformulación de Consultas

En *feedback* de relevancia (RF), el usuario expresa un juicio binario de relevancia en uno o más documentos recuperados con anterioridad y las características de los documentos que el usuario juzga relevantes son usadas por el sistema para modificar la partición relevante/irrelevante (en recuperación binaria) ó para modificar el *ranking* de documentos (en recuperación con *ranking*). De esta manera, el *feedback* de relevancia es usado para modificar los pesos de los términos de indexación y se ha probado que puede incrementar la precisión, para un nivel fijo de *recall*, hasta en un 50% [Sebastiani98]. Aunque, según [Harman92a], esta técnica tiene como objetivo aumentar el nivel de recall, que en el caso de ser resuelto manualmente, sólo es lograble debilitando la consulta (devolviendo de esta forma más documentos irrelevantes).

Un método para recalcular los pesos de los términos de indexación es la fórmula de Rocchio, la que transforma la consulta original en otra mediante la ecuación [Harman92a]:

$$Q_1 = Q_0 + \frac{1}{n_1}\sum_{i=1}^{n_1} R_i - \frac{1}{n_2}\sum_{i=1}^{n2} S_i \tag{3.9}$$

donde

1. Q_0 = el vector para la consulta original;
2. R_i = el vector para documento relevante i;
3. S_i = el vector para documento irrrelevante i;
4. n_1 = el número de documentos relevantes, y,
5. n_2 = el número de documentos irrelevantes.

Hoy en día, ninguno de los buscadores de la Web soporta feedback de relevancia [Cohen96b].

3.7 Modelos de IR

Un modelo es la descripción de un sistema centrada en sus características arquitectónicas y que abstrae los detalles que no son fundamentales para su entendimiento. La construcción de modelos es una actividad de abstracción, lo que a su vez produce generalización, es decir, una descripción abstracta (o parcial) es equivalente a la clase de las descripciones concretas (o totales) compatibles con ella [Sebastiani98].

La siguiente es una taxonomía de los modelos de IR, la cual está de acuerdo con [Frakes92a, Sebastiani98]:

1. modelos matemáticos;
2. modelos de *match* exacto;
3. modelo booleano;
4. modelos de *match* aproximado;
5. modelos bien fundados;
6. modelo del espacio vectorial;
7. modelo probabilístico;
8. modelo de lógica difusa, y,
9. modelo booleano extendido.

Los modelos de IR pertenecen a dos clases fundamentales: matemáticos y cognitivos. Los primeros, matemáticos, describen los (micro) sistemas de IR (modelos reduccionistas). Típicamente se concentran en las modalidades para la representación de documentos y requerimientos y, sobre todo, en la función de *matching*. Los modelos matemáticos son definidos por medio de herramientas matemáticas, por ejemplo, álgebra lineal, cálculo de probabilidades, lógica, etc. Los segundos, los modelos cognitivos, describen los (macro) sistemas que consisten del sistema de IR y el usuario (modelos holísticos). Ellos ven el micro sistema de IR a un nivel mayor de abstracción y típicamente se enfocan en las modalidades de interacción y de la conducta del usuario; se definen por medio de herramientas de la psicología, por ejemplo, psicología experimental, ciencias cognitivas [Sebastiani98].

3.7.1 Modelos Matemáticos de IR

Especificar un modelo matemático de IR consiste en especificar el formato de las IREPs de los requerimientos y los documentos así como especificar la función de *match*. Los modelos matemáticos de IR pueden dividirse en modelos de *match* exacto y modelos de *match* aproximado.

- *Modelos de match exacto*: Se retorna un conjunto de documentos (recuperación binaria). El ejemplo principal es el modelo booleano.
- *Modelos de match aproximado*: Se retorna una lista rankeada de documentos (recuperación rankeada). Estos modelos tienen en cuenta el hecho de que un sistema nunca puede estar seguro sobre si un cierto documento es o no relevante a una necesidad de información dada. Estos pueden dividirse en modelos *ad hoc* y modelos bien fundados. Entre los modelos *ad hoc* podemos encontrar a los modelos de Dice y de Jaccard (sección 3.2.2); que están basados en coeficientes de similitud entre IREPs. Los modelos bien fundados están firmemente basados en intuiciones o metáforas de alguna de las ramas de la matemática [Sebastiani98].

3.7.2 El Modelo Booleano

El modelo booleano data del año 1955, es el primer modelo de IR, el más criticado y el más usado comercialmente. En el modelo booleano, un documento se representa por la conjunción (*and*) de términos índices, típicamente perteneciendo a un vocabulario restringido, por ejemplo:

$$D_1 = \text{(and thriatlon Kona Tinley)} \quad (3.10)$$

$$D_2 = \text{(and duathlon “San Diego” Souza)} \quad (3.11)$$

Estos documentos denotan:

- La ecuación 3.10 denota que D_1 es un documento sobre un *triatlón* corrido en *Kona* (Hawai) donde corrió *Scott Tinley*.
- La ecuación 3.11 denota que D_2 es un documento sobre *duatlón* corrido en *San Diego* donde corrió *Kennie Souza*.

Un requerimiento, que usualmente coincide con la consulta, se representa con una combinación, obtenida a través de *and, or, not* (y, o, no) –ó *and, not, and-not* (y, no y sin)– de términos índices, típicamente pertenecientes a un vocabulario controlado, por ejemplo,

$$R_1 = \text{(and (or thriatlon duathlon) Kona (or Tinley Souza))} \quad (3.12)$$

La función de *matching* recupera un documento D siguiendo un requerimiento R si y sólo si R es una consecuencia lógica de D ($D \models R$). Se debe notar que hay una hipótesis de mundo cerrado ímplicita [Lloyd87] pues la ausencia de un término T en la representación de D es sólo una abreviación notacional para la presencia de *(not T) -no T-* en la misma representación.

Posibles extensiones a dichas funcionalidades básicas son:

1. Truncar términos de consultas, por ejemplo:

   $$R_2 = \text{(and thriat* Kona (or Tinley Souza))} \quad (3.13)$$

   Esto permite que las palabras *thriatlon* y *thriatlete*, hagan match con la consulta.

2. Información sobre adyacencia, distancia entre las palabras (en el caso en que las IREPs de los documentos resultaran de indexación automática), por ejemplo:

   $$R_3 = \text{(and (thriatlon NEAR Timex) Kona Tinley)} \quad (3.14)$$

   Esto permite denotar que se quieren los documentos donde la palabra *thriatlon* aparezca cercana a la empresa relojera, organizadora de eventos, *Timex*.

3. Posibilidad de restringir la búsqueda a campos determinados (título, autor, *abstract*, fecha de la publicación, etc.).

El buscador de la web *Altavista*, el cual puede considerarse un sistema de recuperación comercial, implementa el modelo booleano en el modo de búsqueda avanzado[Alta98]. Además, implementa las extensiones 2 y 3.

En Altavista, la búsqueda por cercanía se hace con el operador `near`, por ejemplo, la consulta `Mary NEAR lamb` buscará las apariciones de `Mary` a menos de 10 palabras de `lamb`. Además usando la sección *Rango de fechas* es posible especificar el rango de fechas en que se quiere que se restrinja la búsqueda de documentos.

También en modo standard, Altavista permite especificar criterios de búsqueda por los atributos de una página [Alta98] (veáse tabla 3.1).

Ventajas del Modelo Booleano

Entre las ventajas del modelo booleano podemos enumerar las siguientes [Sebastiani98]:

1. Eficiencia, obtenida a través del uso de archivos invertidos [Baeza92, Harman92a] (sección 3.11.5).

2. La posibilidad de formular consultas estructuradas, por ejemplo distinguir sinonimia *(or* T_1 T_2*)* de frases sustantivas *(and* T_1 *near* T_2*)*.

3. Intuitividad, sólo en el caso de usuarios expertos, a aquellos usuarios familiarizados con el uso de la lógica Booleana, es inmediatamente claro el porqué un documento ha sido o no recuperado para una consulta dada.

Desventajas del Modelo Booleano

Por otro lado, entre las desventajas del modelo booleano, podemos nombrar las siguientes [Sebastiani98]:

1. Es intimidante ya que en general no hay un método de adquisición automática de la consulta; los que propusieron el modelo booleano eran matemáticos; pero los usuarios legos encuentran el lenguaje del modelo booleano no natural. De hecho, en lenguaje natural, la diferencia entre *"and"* y *"or"* es difusa, y *"apples and oranges"* denota un conjunto mayor a *"apples"*. Una vez que se ha aprendido el lenguaje, el usuario debe dominar los siguientes aspectos: modalidades de combinación, balance de paréntesis, alcance de operadores y prioridad de operadores.

2. El lenguaje es más complejo que lo que las necesidades reales de los usuarios justificaría. Los usuarios expertos tienden a usar sólo requerimientos facetados, es decir, disyunciones de cuasi-sinónimos (facetas) unidos a través de *and*, por ejemplo:

$$\begin{array}{ll} (and & (or\ duathlon\ triahtlon\ marathon) \\ & (or\ Hawai*\ Kona\ Maui) \\ & (or\ \text{“}DaveScott''\ \text{“}ScottTinley''\ \text{“}MarkAllen'')) \end{array} \tag{3.15}$$

   Sin embargo, la complejidad del lenguaje pone a los usuarios inexpertos en problemas.

3. Falta de control de la magnitud de la salida para un requerimiento dado, a menos que el usuario conozca bien la distribución de tópicos en una colección dada. Hay dos problemas asociados a ello. Por un lado, la posibilidad de una salida subdimensionada, muy regularmente, el resultado de la primera formulación del requerimiento

- *Anchor:texto*: Encuentra las páginas que contengan la palabra o frase especificadas en el texto de un hipervínculo. `anchor:"Click here to visit Altavista"` hallaría páginas con *"Click here to visit Altavista"* como un enlace.
- *applet:clase*: Encuentra páginas que contienen un applet de Java específico. Usar `applet:morph` para encontrar páginas usando applets llamados morph.
- *domain:nombreDeDominio*: Encuentra páginas con el dominio especificado. Use `domain:de` para hallar páginas de Alemania, o use `domain:org` para hallar páginas de organizaciones.
- *host:nombre*: Encuentra páginas de una computadora específica. La búsqueda `host:dilbert.unitedmedia.com` hallaría las páginas en la computadora llamada dilbert en unitedmedia.com
- *image:nombreDeArchivo*: Encuentra páginas conteniendo un nombre de archivo específico. Use `image:elvis` para hallar páginas con imágenes llamadas elvis.
- *link:textoURL*: Encuentra páginas con un link a una página con el tecto de URL especificado. Use `link:altavista.digital.com` para encontrar todas las páginas que tengan enlaces a AltaVista.
- *text:text*: Encuentra páginas que contengan el texto especificado en cualquier parte de la página que no sea una etiqueta de una imagen (image tag), un enlace o una URL. La búsqueda `text:cow9` encontraría todas las páginas con el término cow9 en ellas.
- *title:texto*: Encuentra páginas que contengan la palabra o frase especificada en el título de la página (que aparece en la barra de título de los browsers). La búsqueda `title:Elvis` encontraría todas las páginas con Elvis en el título.
- *url:texto*: Encuentra todas las páginas con una palabra o frase específicas en la URL. Use `url:altavista` para hallar todas las páginas en todos los servidores que tuvieran la palabra altavista en el nombre del host, path o nombre de archivo, es decir, la URL completa.

Tabla 3.1: Resumen de criterios de búsqueda de Altavista.

(por ejemplo, de 3 o 4 facetas) es vacío; el usuario debe entonces remover una o más facetas, este proceso le cuesta al usuario tiempo, frustración y una disminución en la precisión de la representación del requerimiento. Por otro lado, la posibilidad de una salida sobredimensionada, por ejemplo, si muchas facetas se remueven, existe la posibilidad de que un conjunto enorme de documentos sea retornado. Para aliviar estos dos problemas, los sistemas booleanos generalmente requieren indicaciones del número de documentos indexados por cada uno de los términos de indexación individuales llamadas postings.

4. La salida obtenida como resultado de un requerimiento dado no es rankeada con respecto al grado estimado o probabilidad de relevancia. Especialmente en el tercer caso, sería necesario contar con una indicación del sistema de cuáles documentos son los más probables de ser relevantes.

5. Existe también la imposibilidad de asignar factores de importancia a los términos de indexación en las IREPs de los documentos y los requerimientos.

6. La obtención de resultados *achatados*[10], y por lo tanto, poco intuitivos y con falta de discriminación de la salida. Por ejemplo, bajo el siguiente requerimiento *(and thriatlon Kona Tinley)*, los documentos *(and thriatlon Kona Tinley)* y *(and banana coconut)* no son igualmente recuperados, cuando el primer documento debería haberlo sido más probablemente. También, siguiendo el requerimiento *(or thriatlon Kona Tinley)*, los documentos *(and thriatlon "New Zealand" "Ken Glah")* y *(and thriatlon Kona Scott)* van a ser ambos recuperados.

Los sistemas booleanos han sido principalmente usados por intermediarios expertos y se considera que son poco aptos para las necesidades de los buscadores modernos de información. Una de las causas del recientemente roto monopolio de los sistemas booleanos dentro del mundo comercial ha sido la falta de consenso de cuál de los modelos alternativos es el mejor [Sebastiani98].

3.7.3 El Modelo de Lógica Difusa

El modelo de la lógica difusa es una extensión en una dirección cuantitativa del modelo booleano; data del año 1976 y el autor es Tahani [Sebastiani98]. En este modelo, un documento se representa con una conjunción pesada (*and*) de términos de indexación, por ejemplo:

$$d1 = (and <thriatlon,\ 0.4> <Kona,\ 0.3> <Tinley,\ 0.5>) \tag{3.16}$$

Los requerimientos se representan como en el modelo booleano. La función de *matching* computa el grado en el cual un documento d satisface un requerimiento r (RSV):

- $max\{w_{1i}, w_{2i}\}$ para la consulta $r = (or\ t_1\ t_2)$
- $min\{w_{1i}, w_{2i}\}$ para la consulta $r = (and\ t_1\ t_2)$
- $1 - w_{1i}$ para la consulta $r = (not\ t1)$

[10] Achatados: *flattened*.

- $min\{w_{1i}, 1 - w_{2i}\}$ para la consulta r = *(and-not* t_1 t_2*)*.

Aquí también tenemos una hipótesis implícita de mundo cerrado, la ausencia del término t en la representación de d es sólo una abreviación notacional para la presencia de <t,0> en la misma representación.

Ventajas del Modelo de Lógica Difusa

Entre las ventajas del modelo de lógica difusa podemos nombrar a las siguientes [Sebastiani98]:

1. Para un usuario familiar con los sistemas booleanos, un sistema difuso es diferente sólo en el hecho en que retorna salidas rankeadas.

2. Si los pesos w_{ij} son todos 0 ó 1, el modelo de lógica difusa se comporta como el booleano; un sistema de IR basado en este modelo es compatible con documentos indexados usando producidos por sistemas booleanos.

3. Existe control en la magnitud de la salida. Por lo tanto, no hay más salida subdimensionada, como los términos que representan a un documento serán muchos más que en una representación booleana, aún si tuvieran pesos menores a 1. Como consecuencia de esto, casi siempre al menos un documento satisface una consulta con grado mayor a 0. Además, tampoco habrá salidas sobredimensionadas; gracias a la recuperación rankeada, el usuario puede recorrer la lista rankeada empezando desde el principio, porque sabe que los documentos más promisorios estarán allí, y detenerse cuando su necesidad de información haya sido satisfecha o la precisión se haya vuelto muy baja.

Desventajas del Modelo de Lógica Difusa

Entre las desventajas del modelo de lógica difusa podemos citar las siguientes [Sebastiani98]:

1. Como en el modelo booleano, no hay adquisición automática de consultas.

2. También como en el modelo booleano, no hay posibilidad de asociar pesos a los términos de indexación en las IREPs de los requerimientos.

3. De vuelta, como en el modelo booleano, hay falta de discriminación en la salida, tenemos resultados *achatados*, y por lo tanto, poco intuitivos. El problema deriva del hecho de que los documentos son rankeados basándose sólo en los valores los términos más pequeños de la consulta. Por ejemplo:

   (a) siguiendo el requerimiento *(and thriatlon Kona Tinley)*, se le da el mismo *RSV* de 0.1 a los documentos *(and <thriatlon, 0.1> <Kona, 0.1> <Tinley, 0.1>)* y al documento *(and <thriatlon, 0.1> <Kona, 0.9> <Tinley, 0.9>)*;

   (b) siguiendo el requerimiento *(or thriatlon Kona Tinley)*, se le da el mismo *RSV* de 0.8 a los documentos `(and <thriatlon, 0.1> <Kona, 0.8> <Scott, 0.1>)` y al documento `(and <thriatlon, 0.8> <Maui, 0.9> <Allen, 0.9>)`.

3.7.4 Modelo de Espacio Vectorial

El modelo del espacio vectorial es de Salton[11] y data de 1965 (ver nota a pie de página 30). En este modelo, un documento d_i y un requerimiento r_j son listas de términos de indexación (usualmente pesados), vistos como vectores de un espacio n-dimensional, donde n es el número de términos de indexación que aparecen al menos una vez en la IREP de al menos un documento [Sebastiani98].

La medida de similitud usada en este modelo es la del coseno (sección 3.2.2). De esta manera, los documentos y los requermientos yacen en la superficie de una n-esfera de radio 1.

Las consecuencias de esta normalización son [Sebastiani98]:

1. El producto escalar entre dos vectores, el coseno del ángulo que los separa y la inversa de la distancia euclídea entre ellos, son *funciones monótonamente crecientes entre sí*, y, por lo tanto, equivalentes desde un punto de vista de la IR;

2. todos los documentos se consideran igualmente informativos; la diferencia entre documentos es así *cualitativa* (qué tipo de información contienen) y no *cuantitativa* (cuánta información contienen). Si bien es una posición filosófica debatible, la normalización ha probado dar buenos resultados.

Ventajas del Modelo de Espacio Vectorial

Las ventajas del modelo de espacio vectorial son las siguientes [Sebastiani98]:

1. Flexibilidad: La interpretación geométrica se puede aplicar en diferentes contextos:

   (a) *Categorización automática de documentos*: Una categoría se representa como un vector de términos pesados de indexación (computada como el centroide de los documentos de un conjunto de entrenamiento); todos los documentos que están a una distancia predeterminada de este vector pertenecen a esta categoría.

   (b) *Filtrado automático de documentos*: Un filtro es un conjunto de categorías; todos los documentos que están a una distancia predeterminada de una de las categorías pasan el filtro.

   (c) *Clustering de documentos*: La n-esfera se puede particionar en clases (ver capítulo 5).

2. Indexación manual y automática: La posibilidad de asignar pesos a los términos permite estas dos variantes.

3. Adquisición automática de consultas.

4. No hay achatamiento de la salida, todos los términos contribuyen de acuerdo a su peso.

5. Mejor efectividad que en el modelo booleano y en el difuso.

[11] Este modelo fue implementado en el sistema SMART [Girardi98a, Sebastiani98].
`ftp://ftp.cs.cornell.edu/pub/smart/smart.11.0.tar.Z`

Desventajas del Modelo de Espacio Vectorial

Las desventajas del modelo de espacio vectorial es la siguiente [Sebastiani98]: Imposibilidad de formular requerimientos estructurados, ya que no hay operadores (*and*, *or*, *not*, *near*).

3.7.5 Modelo Probabilístico

El modelo probabilístico (de Robertson y Spark Jones, 1976) está basado en la hipótesis de que la distribución de los términos en los documentos relevantes es distinta a la distribución de los términos en los documentos irrelevantes [Sebastiani98].

Aquí, los documentos d_i y los requerimientos r_j son vectores binarios y la función de matching es:

$$S_{d_i,r_j} = P(Rel(d_i, r_j)) \tag{3.17}$$

$$= \sum_{k=1}^{n} w_{ki} \cdot w_{kj} \cdot \log \frac{p_k(1-q_k)}{q_k(1-p_k)} \tag{3.18}$$

donde $p_k = P(w_{kx} = 1|Rel(x, r_j))$ y $q_k = P(w_{kx} = 1|\overline{Rel}(x, r_j))$.

La fórmula 3.18 se deriva del teorema de Bayes (ver sección 5.2.1) y puede interpretarse de la siguiente manera: el matching se realiza de acuerdo al producto interno pero:

- se le da una importancia mayor a los términos que ocurren en muchos documentos irrelevantes y están ausentes en muchos documentos irrelevantes;
- se le da una importancia menor a los términos que ocurren en muchos documentos irrelevantes y están ausentes en muchos documentos relevantes.

Este modelo se usa en los clasificadores y *Lira* de Balabanovic (sección 9.3.8) y *Syskill & Webert* de Pazzani (secció 9.3.13).

3.7.6 Otros Modelos de IR

En esta sección se detallan otros modelos de IR. Estos modelos se detallan sólamente por una cuestión de completitud. Los modelos que faltan son:

1. *Modelo Booleano Extendido*: Es una generalización del modelo booleano, del de lógica difusa y del de espacio vectorial [FoxE92, Sebastiani98].
2. *Modelo Lógico*: El matching consiste en el cálculo del valor v para el cual la fórmula $P(d \rightarrow r) = v$ es válida, donde d y r son fórmulas que representan a un documento y un requerimiento respectivamente, v es el RSV, P es un operador de probabilidad (subjectiva), y $\rightarrow$ es el conectivo condicional de la lógica probabilística. Este modelo es de van Rijsbergen, 1986 [Sebastiani98].
3. *Modelo de Red de Inferencia*: Los documentos y los requerimientos se representan como fragmentos de una red Bayesiana de inferencia [], en la cual los nodos representan objetos (términos de indexación, documentos, requerimientos, etc.) y las aristas representan relaciones de dependencia condicional entre los nodos. Según [Sebastiani98], este modelo está emergiendo fuertemente. Este modelo es de Turtle y Croft, 1990.

3.8 Stop Lists

El agente construido clasifica documentos HTML. Éstos, antes de clasificarse, documentos son procesados mediante las técnicas de eliminación de *stop list* y aplicación de *stemming*. La técnica de eliminación de stop list consiste en eliminar todas aquellas palabras superfluas de un documento y la de stemming en quedarse sólo con la raíz de la palabra.

Dado un idioma predeterminado, existe un conjunto de palabras que aparecen en casi cualquier documento. En el área de la recuperación de información, icho conjunto de palabras se llama *stop list* [FoxC92].

La poca información que aportan las palabras de la stop list sobre la relevancia o irrelevancia de un documento para una consulta viene dada por el hecho de que como dichas palabras aparecen en todos (o casi todos) los documentos de la base de datos, su uso como índice de búsqueda nos devolvería todos (o casi todos) los documentos de la misma.

Si bien existe, para cada idioma, una lista conocida de *stop words*, también cabe concebir una para cada tema. Por ejemplo, en una búsqueda en una base de datos restringida a temas de computación, la aparición de la palabra *computadora* no aportaría mayor información para decidir la relevancia de un documento.

3.9 Algoritmos de Stemming

En *Querando!*, una vez que se aplicó elimación de stop list a los términos del documento a clasificar, se aplica *stemming* a los términos restantes. En esta sección se explica en qué consiste la técnica y se dan detalles de implementación.

3.9.1 Definición

Los algoritmos de *stemming* son programas que relacionan términos de búsqueda e indexación similares [Frakes92b]. El stemming se usa para mejorar la efectividad de la recuperación y para disminuir el tamaño de los índices. Intuitivamente, el stemming consiste de tomar una palabra y *achicarla* hasta su raíz y entonces buscar otras palabras que sean iguales a dicha raíz [Dwight97, pág. 278]; el proceso de igualación se conoce en la literatura como *conflation* [Frakes92b]. Por ejemplo, la palabra *engineering*[12] tiene como raíz a *engineer*[13] [Frakes92b].

3.9.2 Clasificación de los Algoritmos de Stemming

Básicamente, los métodos stemming se pueden dividir en manuales y automáticos. El *método manual* consiste en usar a una persona para hacer conflación. Por otro lado, hay cuatro clases de *métodos automáticos*: por *look-up*, por variedad de sucesores, de n-gramas, y por remoción de afijos [Frakes92b].

[12]Engineering: Ingeniería.
[13]Engineer: Ingeniero.

Stemmers por Look-Up

La idea del stemming por look-up es, por cada palabra del idioma, tener el stem asociado. En teoría, funcionaría eficientemente si usáramos B-Trees o hashing [Aho83b, Helman86, Weiss92]. El problema es que, en la práctica, dicha tabla no existe.

Stemmers por Variedad de Sucesores

En términos poco formales, la *variedad de sucesores* de una cadena de caracteres es el número de caracteres diferentes que lo siguen en palabras de algún cuerpo de texto. Por ejemplo, de acuerdo a [Frakes92b]:

> Si tenemos las palabras *able*, *axle*, *accident*, *ape* y *about*[14]. Para determinar la variedad de sucesores de *apple*, se usa el siguiente proceso. La primer letra es *a*. *a* está seguida por los cuatro caracteres: *b*, *x*, *c* y *p*. Así, la variedad de sucesores de *a* es cuatro. La siguiente variedad de sucesores para *apple* sería uno, ya que sólo *e* sigue *ap* en el cuerpo de texto, y así sucesivamente.

Una vez que la información de sucesores se ha derivado para una palabra, esta información puede usarse para segmentar la palabra. En [Frakes92b], se describen varios métodos para hacerlo. El enfoque general es que la variedad de sucesores decrecerá a medida que el prefijo crece, luego setearemos un valor de corte al que usaremos para decidir cuándo tenemos el stem.

Este método parece ser muy caro, ya que para obtener el stem de una única palabra, es menester analizar todo el texto varias veces.

Stemmers de n-gramas

Los digramas que conforman la palabra *elefante* son:

$$el, le, ef, fa, an, nt, te. \tag{3.19}$$

En cambio, los trigramas que componen la misma palabra son:

$$ele, lef, efa, fan, ant, nte. \tag{3.20}$$

El método de los *digramas* o *trigramas*, dependiendo de si n vale 2 o 3, no produce stems en el sentido propio de la palabra sino que calcula la cantidad de digramas (trigramas) de las dos palabras a igualar y determina su grado de similitud usando el coeficiente de Dice [Frakes92b] (ver sección 3.2.2), que mide la similitud entre dos palabras w_i y w_j:

$$S(w_i, w_j) = \frac{2C}{A + B} \tag{3.21}$$

donde

- A es la cantidad de digramas (trigramas) de w_i
- B es la cantidad de digramas (trigramas) de w_j
- C es la cantidad de digramas (trigramas) en común de A y B.

[14] *Able*, *axle*, *accident*, *ape* y *about* significan *capaz*, *eje*, *accidente*, *mono* y *acerca* respectivamente.

Stemmers por Remoción de Afijos

Dada una palabra w, los *stemmers por remoción de afijos* trabajan eliminando prefijos y sufijos de w, generando una secuencia

$$w = w_0 \Rightarrow w_1 \Rightarrow \ldots \Rightarrow w_{i-1} \Rightarrow w_i \ldots \Rightarrow w_n = w' \tag{3.22}$$

donde w_i se deriva de w_{i-1} sacando un prefijo o un sufijo de w_{i-1}; y, w' es el stem de w.

Cada paso de la ecuación 3.22, es generado por la aplicación de una regla de reescritura del estilo [Frakes92b]

$$SufijoViejo \Rightarrow SufijoNuevo. \tag{3.23}$$

Un ejemplo de reglas para un stemmer de este tipo se puede hallar en [Frakes92b]:

> Si una palabra termina en *ies* pero no en *eies* o *aies* Entonces $ies \Rightarrow y$.
>
> Si una palabra termina en *es* pero no en *aes*, *ees* o *oes* Entonces $es \Rightarrow e$.
>
> Si una palabra termina en *s* pero no en *us* o *ss* Entonces $s \Rightarrow NULL$.

En el algoritmo citado por [Frakes92b] sólo se aplica la primer regla que hace *match*. Supongo que los criterios de la teoría de sistemas de producción [Rich94, Winston94] son aplicables aquí y se podrían usar sus técnicas; sin embargo, éstas no se usaron en la implementación del agente *Querando!*.

En general, estos algoritmos trabajan hasta que no se pueden aplicar más reglas. El problema que puede surgir es el de excesiva reducción del término que está siendo "stemmizado"[15]. Para ver un ejemplo de esto, consideremos la "stemmización" de *ski* y *sky*[16]; el resultado en ambos casos sería *sk*, lo que haría que el algoritmo igualara ambos términos cuando en realidad no tienen nada que ver (sección 3.9.3). Para solucionar este problema, existen técnicas de recodificación y *matching* parcial.

En [Frakes92b], se muestra una implementación en lenguaje C [Kernigham85] del algoritmo de stemming de Porter.

3.9.3 Desempeño de los Algoritmos de Stemming

Hay varios criterios para juzgar el desempeño de los algoritmos de stemming: correctitud, efectividad de recuperación y performance de compresión [Frakes92b].

Correctitud

Hay dos maneras en las que el stemming puede ser incorrecto:

- *Overstemming*: En este caso, se quitan demasiados caracteres de los términos y se igualan términos no relacionados.
- *Understemming*: En este caso, por otro lado, se quitan menos caracteres de lo necesario y no se igualan términos relacionados.

[15] Perdón por mi *spanglish*.

[16] Esquí y Cielo, respectivamente.

Efectividad de la Recuperación

La efectividad de la recuperación se mide en términos de precisión y recall (ver sección 3.5). Estudios reportados por [Frakes92b] muestran que, donde se han hallado efectos del stemming, éstos han sido positivos. Además, no hay indicios de que el stemming degrade la efectividad de recuperación. Por otro lado, el stemming es tan efectivo como la conflación manual.

Performance de la Compresión

Con respecto a la performance de compresión lograda por los métodos de stemming, los datos aportados por W. Frakes [Frakes92b] son inconsistentes, primero afirma que se puede lograr hasta un 50% de ahorro de almacenamiento y, luego afirma que las reducciones van del 13.5 al 20%. Sin embargo, cita trabajos donde se logra un 40% de compresión al usar stemming.

3.10 Thesauri

Una manera de "ensanchar" el alcance de una búsqueda es usar el un thesaurus, un archivo separado que enlaza garandes números de palabras con listas de sus equivalentes más comunes (sinónimos) [Dwight97]. Algunos thesauri nuevos pueden automáticamente agregar y correlacionar todas las nuevas palabras qu ocurren en los documentos que leen. Un thesaurus puede ser de ayuda especialmente para usuarios que no son familiares con una cierta terminología. Mantener manualmente un gran thesaurus es tan dificultoso como mantener cualquier otro trabajo de referencias. Es así como los thesauri de los nuevos motores de búsqueda estadísticamente rastrean los referencias cruzadas más comunes de cada palabra, así las más comunes se pueden agregar a una consulta.

3.11 Estructuras de Datos para Recuperación de Información

Las estructuras de datos para recuperación de información son [Baeza92]: vectores, árboles de búsqueda, hashing, árboles digitales, índices lexicográficos (ó índices ordenados).

3.11.1 Vectores

Los *vectores* sólo sirven para implementar búsquedas en colecciones estáticas y son sólo una opción eficiente cuando los datos están ordenados para permitir búsquedas dicotómicas [Aho83b, Baeza92, Weiss92].

3.11.2 Árboles de Búsqueda

Entre los árboles de búsqueda se hayan los tradicionales árboles binarios de búsqueda, Avl, 2-3, B, B+, B* (éstos dos últimos son importantes en el caso de usar estructuras en disco). En general, el tiempo de acceso será logarítmico en la cantidad de ítems de la colección [Aho83b, Baeza92, Helman86, Weiss92, Wiederhold85].

3.11.3 Hashing

Una *función de hashing* $h(x)$ mapea una clave x a un entero en un rango dado. Las funciones de hashing están diseñadas para producir valores uniformemente distribuidos en el rango dado. El valor de hashing también se conoce como *signature* [Baeza92].

Una función de hashing se usa para mapear un conjunto de claves a slots en una tabla de hashing. Si la tabla de hashing da el mismo slot para dos claves diferentes, se dice que tenemos una *colisión*. Las técnicas de hashing sólo difieren en la forma en que se manejan las colisiones [Baeza92]. Hay dos clases de esquemas de resolución de colisiones: direccionamiento abierto y direccionamiento por overflow.

Direccionamiento Abierto

En direccionamiento abierto, la clave colisionada es "rehasheada" en la tabla, computando un nuevo valor para el índice. La técnica más usada es *hashing doble*, que usa una segunda función de hashing [Baeza92].

Direccionamiento con Overflow

En direccionamiento con overflow[17], la clave colisionada es almacenada en un área de derrama, tal que los valores con el mismo valor de hash se enlistan juntos [Aho83b, Baeza92]. El problema de este enfoque es que para sinónimos (claves con el mismo valor de hash), la búsqueda degenera en lineal. Sin embargo, para tablas casi vacías (con densidad baja [Wiederhold85]), el tiempo de búsqueda es constante.

Otra variante es *hashing extensible* [Baeza92, Wiederhold85] que usa hashing en dos niveles, uno para un directorio y otro para nivel de *bucket* (conjunto de sinónimos).

La desventaja del hashing es que, como las claves están desparramadas por la estructura de datos, no es posible hacer un listado ordenado por clave.

Desorden Organizado

Una combinación entre B-trees y tablas de hashing es el *desorden organizado* en el que los buckets se organizan en B-trees [Baeza92].

3.11.4 Árboles Digitales

Tries

La búsqueda eficiente de prefijos puede ser hecha usando índices. Uno de los mejores índices para búsqueda de prefijos es el *árbol digital* o *trie binario* construido a partir de un conjunto de substrings de un texto [Baeza92]. Si el alfabeto es ordenado, tenemos un árbol ordenado lexicográficamente. La raíz del trie usa el primer caracter, los hijos de la raíz usan el segundo caracter, y así sucesivamente. Si el subtrie restante contiene sólo un string, dicho string se guarda en una hoja.

[17] Direccionamiento con área de derrama.

Patricia trees

Un *Patricia tree* es un trie con la restricción adicional de que los nodos con un solo descendiente se eliminan [Baeza92, Gonnet92].

3.11.5 Archivos Invertidos

El concepto de archivo invertido es como sigue. Asuma un conjunto de documentos. A cada documento se le asigna un lista de claves o atributos, con pesos de relevancia opcionales asociados a cada clave (atributo). Un *archivo invertido* es entonces la lista ordenada (o índice) de claves (atributos), con cada clave teniendo asociados el conjunto de documentos donde aparece [Harman92a]. La ventaja de este tipo de estructura es la eficiencia para hallar claves en documentos, mientras que la desventaja es que el tamaño del índice es del orden del 10 al 100% o más del tamaño del texto mismo [Harman92a]; Donna Harman afirma que el tamaño de los archivos invertidos va del 50 al 300% [Harman92a].

Usualmente, hay algunas restricciones impuestas en estos índices y consecuentemente en las búsquedas posteriores [Harman92a]:

- un *vocabulario controlado* que es la colección de claves que serán indexadas. Las palabras que no estén en el vocabulario no serán indexadas, y, luego, no buscables.
- una lista de *stop words* (veáse sección 3.8).
- un conjunto de reglas, el comienzo de una palabra o un texto que será indexable. Estas reglas lidian con el tratamiento de espacios, signos de puntuación, o prefijos standard, y pueden tener impacto en qué términos serán indexados.
- una lista de strings a ser (o no) indexados. En grandes bases de datos textuales, no todos los caracteres son indexados (por ejemplo, secuencias numéricas).

Entre las estructuras de datos usadas para implementar archivos invertidos podemos citar a: arreglos ordenados, B-trees, tries, tablas de hashing o una combinación de ellos.

Construcción de un Archivo Invertido

La construcción de un archivo invertido consiste básicamente de tres pasos [Harman92a]: primero, el texto de entrada debe ser parseado en una lista de palabras junto con su ubicación en el texto (tal vez eliminando *stop words* y aplicando *stemming* (ver secciones 3.8 y 3.9). Segundo, esta lista debe ser llevada de una lista de términos con su ubicación a una lista de términos ordenada para permitir la búsqueda (en el caso de una implementación con arreglos ordenados esto es así, mientras que en el caso de usar un árbol-B o una tabla de hashing, estos dos pasos se harían simultáneamente. Tercero, un paso adicional consiste en postprocesar el archivo invertido para agregar pesos a los términos (sección Modelo del Espacio Vectorial 3.7.4) o para comprimir los archivos.

3.11.6 Estructuras de Datos de IR en *Querando!*

La construcción del agente *Querando!* implicó la implementación de varias estructuras de datos básicas como ser colas [Aho83b, Weiss92], árboles de búsqueda balanceados (AVL)

[Aho83b, Weiss92], tablas de hashing abierto [Weiss92, Wiederhold85], vectores de booleanos empaquetados (debido al gran tamaño de las estructuras de datos manejadas, esta técnica proporciona un ahorro de memoria considerable), vectores dispersos [Frakes92a].

La construcción se hizo usando programación orientada a objetos y usando clases *container* genéricas en C++ siguiendo lineamientos propuestos por [Brokken95].

3.12 Sistemas de Filtrado de Información

Los *sistemas de filtrado de información* (SFI) son similares a los sistemas de IR (SRI) convencionales porque ayudan en la selección de documentos que satisfagan las necesidades de información de los usuarios. Sin embargo, existen diferencias entre los SFI y los SRI [Mostafa97]:

1. Los SRI usualmente se diseñan para facilitar la recuperación rápida de unidades de información para satisfacer necesidades de relativamente corto plazo de una población diversa de usuarios. En contraste, los SFI comunmente se personalizan para soportar las necesidades de información de largo plazo de un usuario particular o un grupo de usuarios con necesidades similares. Así, cumplen el objetivo de la personalización adquiriendo información directa o indirectamente del usuario. En los SFI, estas necesidades de información se representan como perfiles de interés, los cuales se usan posteriormente con propósitos de *matching* o *ranking*.

2. Los perfiles de interés se mantienen más allá de una simple sesión y pueden modificarse basados en *feedback* del usuario.

3. Otra diferencia tiene que ver con la fuente de los documentos. Los SRI usualmente operan en un conjunto relativamente estático de documentos mientras que los SFI usualmente identifican documentos relevantes de un flujo de documentos que cambia continuamente.

El filtrado de documentos es difícil por las siguientes razones [Mostafa97]:

1. *Dificultad de la representación*: Para operar eficientemente, los SFI deben adquirir y mantener conocimiento preciso tanto de documentos como de usuarios. La naturaleza dinámica de los intereses del usuario hace muy difícil esta tarea.

2. *Estocacidad del feedback*: El feedback de relevancia del usuario puede a veces parecer aleatorio desde el punto de vista del SFI. Esto puede ocurrir debido a varias razones. Primero, el usuario particular interactuando con el sistema puede tener necesidades inciertas o puede no ser muy discriminante de sus necesidades. También, en ciertas ocasiones, el feedback del usuario puede estar motivado por características particulares en los documentos que no son parte del esquema de representación subyacente. Luego, el feedback generado basado en esas "características faltantes" podría parecer aleatorio al SFI y éste sería incapaz de determinar que causó tal feedback.

3. *Cambio de intereses del usuario*: Debido a razones profesionales o personales, los intereses de un usuario pueden evolucionar o reemplazarse por otros. Estos cambios pueden ocurrir a los largo de un período de corta duración o uno de larga duración.

4. *Cambios en el flujo de documentos*: En cualquier momento, se pueden introducir nuevos temas en el flujo de documentos y los intereses del usuario pueden cambiar en base a ellos. Además, puede no hacer suficientes documentos representativos disponibles para la etapa de entrenamiento.

3.13 Comparación entre Disciplinas

La siguiente es una comparación entre diversas disciplinas que están relacionadas y está basada en [Sebastiani98]:

- *Data retrieval*: Como en bases de datos. Los datos se recuperan en un formato predefinido; y la satisfacción de un query más que la de un usuario es el objetivo.
- *Knowledge retrieval*: Como en inteligencia artificial, donde un hecho A se infiere de una base de conocimiento R de hechos expresados en un formalismo lógico L.
- *Conceptual information retrieval*: Como en *datamining*. Un requerimiento se contesta no sólo con un conjunto de documentos o un ranking de documentos, sino con una respuesta generada a partir de un análisis semántico de los documentos. Se aplica a dominios muy acotados y se requiere mucho conocimiento de los mismos.
- *Information browsing*: Como en hipermedia. Los documentos relevantes se encuentran a partir de una intervención activa del usuario y no por una rutina de búsqueda.
- *Information Filtering Systems*: Primero, el filtrado de información a través del uso de perfiles representan intereses a largo plazo, mientras que la satisfacción de un query en IR tipicamente pueden satisfacerse recuperando un conjunto particular de documentos. Segundo, las aplicaciones de IR asumen que el cuerpo de documentos no cambia a menudo, mientras que el filtrado de información asume un flujo constante de documentos sensibles al tiempo. Por último, el filtrado es el acto de remover ítems irrelevantes de este flujo, mientras que IR es el acto de encontrar ítems relevantes en la base de datos [Balabanovic98].
- *Assisted Browsing Systems*: En vez de proveer al usuario con páginas selectas, algunos sistemas asisten al usuario en su "browseo". El sistema WebWatcher[18] requiere que el usuario establezca un objetivo particular y, entonces, interactivamente ofrece consejo sobre qué enlace seguir. Así, el sistema aprende a determinar si su consejo es o no seguido, y también pregunta al usuario que indique si tuvo éxito o fracasó cuando su búsqueda terminó.

3.14 Resumen

En este capítulo, se expusieron los conceptos fundamentales de la disciplina de la recuperación de información relacionados con los agentes de filtrado de información estudiados y con la implementación del agente *Querando!*.

[18] `http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-6/web-agent/www/project-home.html`

Capítulo 4

Representación de Documentos en *Querando!*

En este capítulo, se describen las técnicas de representación de documentos usadas en el agente *Querando!*. El capítulo está estructurado de la siguiente manera: primero, se describe brevemente el lenguaje HTML; segundo, se especifica cómo se hizo el "parseo" de las páginas HTML; tercero, el procesamiento de los stop lists; cuarto, el algoritmo de stemming; quinto, el uso de la información de formato para mejorar la representación de los documentos; sexto, las mediciones preliminares probando la representación de los documentos.

4.1 El Lenguaje HTML

Por ahora, veremos a la WWW como un digrafo de documentos HTML relacionados por hipervínculos (para una definición más rigurosa veáse la sección 8.2). En esta sección, se describe el formato básico de los documentos HTML para poder abordar el problema de la representación de los mismos. Por lo tanto, se obviarán cuestiones concernientes a requerimientos a servidores, remotos, URIs y URLs, etc., dejándolos para una discusión posterior.

4.1.1 Definición

Para publicar información en forma global, se necesita un lenguaje común, que entiendan todas las computadoras. El lenguaje de publicación de la WWW es el HTML (por *HyperText Markup Language*, o lenguaje de marcado de hipertexto) [Raggett98, Scharf96].

HTML le da a los autores la posibilidad de:

- Publicar documentos en línea en la forma de encabezados, texto, tablas, listas, fotos, etc.
- Recuperar información en línea a través del click de un botón.
- Diseñar formularios para conducir transacciones con servicios remotos, para buscar información, hacer reservaciones, comprar productos, etc.
- Incluir hojas de cálculo, videos y otras aplicaciones dentro de los documentos.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html40/strict.dtd">
<HTML>
<HEAD>
<TITLE>Mi primer documento HTML</TITLE>
</HEAD>
<BODY>
<P>Hola mundo!
</BODY>
</HTML>
```

Figura 4.1: Un ejemplo de un documento HTML simple.

4.1.2 Estructura de un Documento HTML

Un documento HTML 4.0 está compuesto de tres partes [Raggett98]:

1. una línea conteniendo información de la versión HTML;
2. una sección declarativa de encabezado (delimitada por el tag `HEAD`);
3. un cuerpo, el cual contiene el contenido propio del documento. El cuerpo puede estar implementado por el tag `BODY` o el tag `FRAMESET`.

Espacios en blanco (como espacios, enters, tabs y comentarios) pueden aparecer antes o después de cada sección. Las secciones 2 y 3 deberían estar delimitadas por el tag `HTML`.

En la figura 4.1 (4.1) se puede ver un ejemplo de un documento HTML simple. El tag `<!DOCTYPE...>` contiene información de versión.

El Encabezado del Documento

Como parte del encabezado de un documento podemos encontrar los siguientes elementos:

1. El elemento `HEAD`: El elemento HEAD contiene información acerca del documento corriente, como su título, claves que pueden ser útiles a los motores de búsqueda y otros datos que no son considerados como parte del contenido del documento.
2. El elemento `TITLE`: El elemento TITLE especifica el título del documento; el dar títulos significativos es responsabilidad de los autores. Los títulos pueden contener entidades de caracteres[1] pero no puede contener otros tags.
3. Metadatos: HTML le permite a los autores especificar metadatos –información acerca de un documento en vez de contenido– en una variedad de formas.

   Por ejemplo, para especificar el autor de un documento, se puede usar el tag `META` así:

[1]Las entidades de caractes son aquellos caracteres que no aparecen en el código Ascii de 7 bits; por ejemplo, la `á` se simbolizará ` ` dado que 160 corresponde al ascii de la 'a' acentuada.

```
<HTML>
<HEAD>
      <-- For speakers of US English -->
      <META name="keywords" lang="en-us"
            content="vacation, Greece, sunshine">
      <-- For speakers of British English -->
      <META name="keywords" lang="en"
            content="holiday, Greece, sunshine">
      <-- For speakers of French -->
      <META name="keywords" lang="fr"
            content="vacances, Gr&egrave;ce, soleil">
</HEAD>
<BODY>
....
</BODY>
</HTML>
```

Figura 4.2: El uso del tag `META` para dar información sobre claves de búsqueda.

`<META name="Author" content="Sergio Gomez»`

El tag `META` especifica una propiedad ("Author") y le asigna un valor ("Sergio Gomez").

El lenguaje HTML contiene un tag especial llamado `META` que sirve para que los programadores HTML o diseñadores de páginas web agreguen explícitamente información para los buscadores de la web como el *Altavista*, *Lycos* y *Yahoo!* entre otros tantos. El tag `META` permite definir el idioma en el que está escrita la página y definir explícitamente claves de búsqueda [Raggett98, p. 57]. En el caso de la metainformación destinada a los buscadores, existe una sintaxis estandarizada para darle información sobre las claves de búsqueda con debe indexarse el documento. Esto se aprecia en la figura 4.1.2 (4.2).

El Cuerpo del Documento

El cuerpo del documento HTML se encuentra delimitado por los tags `<BODY>` y `</BODY>`. Allí, se especificará el contenido de la página HTML.

Los elementos que se pueden especificar son los siguientes:

1. parráfos, destacar oraciones en negrita, cursiva, etc;
2. tablas;
3. fotos;
4. frames (es decir porciones de la página cuyo contenido está definido en otra documento HTML);

5. porciones de código ejecutable en la forma de guiones JavaScript [JS99, Gulbransen98], los cuales van entre los tags `<SCRIPT...>` y `</SCRIPT>`;

6. programas en la forma de *Java applets* [Weber97];

7. elementos de estilo en la forma de hojas de estilo en cascada [Bert98, Gulbransen98].

4.2 Parsing de Páginas HTML

La obtención de los vectores características a partir de los documentos HTML requirió la construcción de un parser de las mismas. Si bien esta labor se podría haber hecho usando alguna herramienta de diseño de compiladores como LEX o YACC [Aho83a], las rutinas necesarias se implementaron completamente[2].

El procesamiento de las páginas HTML es el siguiente:

1. Dada una página HTML p (figura 4.3), obtener su versión decorada p_2 (figura 4.4). p_2 es una lista donde cada item es un tag HTML o es texto o es código de hojas de estilo en cascada –el tag anterior será un `STYLE` y el siguiente un `/STYLE`– o es una porción de código de script –entre los tags `SCRIPT` y `/SCRIPT`.[3]

2. Una vez que se tiene la página decorada p_2, los tags corresponden a aquellos items que comienzan con `<` y el texto que forma parte del contenido de la página es el resto de los items (salvo que sea un *script* o un *style*).

3. A partir del contenido de la página se obtienen las palabras que forman parte de éste, se eliminan las stop words y luego a los términos que quedan se les aplica stemming (ver más adelante secciones 4.3 y 4.4).

4.3 Stop Lists en *Querando!*

Los documentos son procesados, antes de ser clasificados por el agente, mediante las técnicas de eliminación de *stop list* y aplicación de *stemming*. La técnica de eliminación de stop list consiste en eliminar todas aquellas palabras superfluas de un documento (sección 3.8). La implementación de esta técnica se describe en esta sección mientras que la de stemming se describe en la sección siguiente.

[2]LEX sirve para construir analizadores léxicos o *scanners* y YACC para construir *parsers*, ambos toman como entrada un archivo de texto correspondiente a la gramática BNF del lenguaje al que pertenecen los documentos a analizar y devuelven un módulo en C que realiza el análisis [Aho83a].

[3]Las porciones de código correspondientes a guiones como PHP se consideran tags. Los guiones PHP van en un tag que empieza con *'?php'*; por ejemplo: `<?php echo "Hola, soy un script PHP!"; ?>` se ejecutará en el servidor y su resultado es que el string *"Hola, soy un script PHP!"* será el contenido de la página en el lugar donde figuraba el tag de marras [Bakken99].

```
<HTML>
<HEAD><TITLE>Este es el titulo</TITLE></HEAD>
<BODY>
<H1>Titulo gran titulo gran</H1>
<P>Primer parrafo de texto.
<P>Segundo parrafo de texto.
</BODY>
</HTML>
```

Figura 4.3: Un ejemplo de una página HTML a parsear.

```
<HTML>
<HEAD>
<TITLE>
Este es el titulo
</TITLE>
</HEAD>
<BODY>
<H1>
Titulo gran titulo gran
</H1>
<P>
Primer parrafo de texto.
<P>
Segundo parrafo de texto.
</BODY>
</HTML>
```

Figura 4.4: Un ejemplo de una página HTML "decorada".

4.3.1 La Lista de Palabras Stop

La lista de stop words usadas en la implementación del generador de características usado en *Querando!* aparece en las tablas A.1, A.2 y A.3. Esta lista está basada en la de los trabajos de Fox [FoxC92] y Lewis [Lewis92].

En *Querando!*, toda palabra que no pertenezca a la stop list se considera como aportante de información para determinar la relevancia o irrelevancia de un documento.

Un enfoque que no fue aprovechado en la implementación actual es el siguiente: cuando el usuario obtiene los documentos a filtrar de parte de un buscador (como Altavista), el clustering sobre dichos documentos se podría hacer considerando a los términos de la consulta como stop words, ya que como aparecerán en todos los documentos no contendrán mayor información. Sin embargo, si, en una sesión previa, el usuario hubiera usado (implícitamente) esos términos para notar la relevancia de un documento, ésta se perdería. Por lo tanto, este enfoque se dejó de lado deliberadamente.

4.3.2 Detalles de Implementación

La forma en que está implementado el chequeo de stop words en el agente es el siguiente:

Dado el documento decorado hd, se genera un nuevo documento decorado hd_2 donde por cada renglón que no es un tag HTML, se extraen todas aquellas palabras que estén en la lista de stop words. Existen dos casos especiales donde el texto que está entre tags no es analizado y son bien puntuales:

1. Las porciones de código correspondientes a *scripts*, las que aparecen entre los tags `<SCRIPT>` y `</SCRIPT>`.

2. El código correspondiente a hojas de estilo en cascada, el que aparece entre los tags `<STYLE>` y `</STYLE>`.

Por cada palabra del documento, se determina si está contenida en una tabla que contiene la stop list de las tablas A.1, A.2 y A.3. Dicha tabla está implementada con un árbol AVL para tener búsquedas en tiempo logarítmico. Aunque se sabe que la mejor opción era tener una tabla de hash (para tener acceso en orden 1), el reuso de estructuras de datos impidió la construcción de funciones de hash especializadas a cada conjunto de datos[4].

Esta parte está implementada por la clase *GeneradorDeFeatures* la cual utiliza otras clases como la *Tabla* genérica, el *DecoradorDeHTML*, etc.

4.4 El Algoritmo de Stemming de *Querando!*

Cuando la necesidad de información se halla asociada a un término de búsqueda, en general los documentos conteniendo términos en la misma familia de palabras serán relevantes a la necesidad de información y querrán recuperarse también. Una manera de lograr esto es reduciendo todos los términos de una familia de palabras a una expresión común. Esta expresión común está dada por la raiz de todos los términos de una misma familia de

[4] Además, como se estaba construyendo un prototipo, los lujos de eficiencia generalmente se dejan para el final.

palabras. El método por el cual se obtiene la raiz de una palabra se llama un *algoritmo de stemming* (ver sección 3.9).

En esta sección, se describe el algoritmo de stemming usado en *Querando!*.

4.4.1 Algoritmo Implementado

Si bien, como ya se citó en secciones anteriores, se poseía una implementación de un algoritmo de stemming, se decidió implementar uno *ad-hoc*.

Como se había dicho antes, dada una página HTML h, primero sufre la eliminación de tags HTML, luego de stop words yf a los términos restantes se les aplica stemming.

Si bien se estudiaron los tipos de "stemmers" hallados en la literatura (stemmers por look-up, por variedad de sucesores, de n-gramas y por remoción de afijos [Frakes92b]), se implementó uno nuevo. El algoritmo implementado trabaja por remoción de sufijos.

Dada una palabra inglesa P:

1. Determina cuál es el sufijo más largo que puede eliminar (digamos S).

2. Si no encuentra ningún sufijo, entonces el algoritmo termina y P es la raiz buscada. Por otro lado, si encuentra dicho sufijo, obtiene $P := P - S$ (la operación de quitar S del final de P) y se va al paso 1.

La eliminación del sufijo más largo ayuda a la correctitud del algoritmo. Por ejemplo, los verbos en tiempo presente en la tercera persona del singular se forman agregando s o es al infinitivo. Así, un verbo terminado en es es plausible de que se elime sólo la s o el es completo. Al eliminar el es completo, nos aseguramos que se forme correctamente la raiz del verbo; así, otras formas del verbo, como el caso de las formas ing, darán lugar a la misma raiz. Un ejemplo de esto es *use, uses, using*, las que darán lugar a *us*.

Otro ejemplo es la palabra *usefulness*, como ésta termina en s, si se decidiera a eliminar dicha s, nos quedaría la palabra $usefulnes$, la eliminación posterior de es, haría que nos quedara $useful n$, lo que, definitivamente, no es el escenario adecuado. Por otro lado, la eliminación del sufijo más largo $ness$, hace que nos quede $useful$, luego use, y luego us; dando así, el resultado esperado.

4.4.2 Lista de Stems

La lista de *stems* o declinaciones de palabras está basada en la lista de sufijos del diccionario de inglés Longman [Quirk90, p. B10–B14]. La lista está compuesta por las siguientes partículas de la tabla A.4.

En la lista original propuesta por el diccionario figuraba la palabra *phone*[5], la cual fue excluída de la lista usada en el programa debido a que podía eliminar términos importantes en la búsqueda de páginas relacionadas con la telefonía.

4.4.3 Correctitud del Algoritmo de Stemming

Con respecto a la correctitud del algoritmo de *stemming*, no se realizaron experimentos formales para determinar si el algoritmo de "stemming" implementado realiza *overstemming* o *understemming*. Por los resultados de las corridas en tiempo de depuración, el

[5]El stem *phone* aparece en palabras tales como *Francophone*.

algoritmo hace "overstemming". Por ejemplo, si tomamos en cuenta la palabra *usefulness*, el algoritmo de stemming, primero eliminará 'ness', luego 'ful', por último, eliminará la última 'e' quedando 'us' (quizá en forma exagerada). Sin embargo, *user* se resuelve a la misma raiz, lo que hace parecer correcto al algoritmo. Sin embargo, una referencia a Estados Unidos (*'US'*), se reducirá trivialmente a la misma raiz (resultando en *overstemming* en un número limitado de casos).

Para evitar el overstemming, se le agregó una condición de terminación al algoritmo. La misma consiste en terminar el algoritmo cuando el largo del stem es menor al valor de la variable `LARGO_MINIMO_STEM` (actualmente en 4). Además, esto impide que algunos términos *desaparezcan*. Tal era el caso de la palabra *skinless*, la cual era separada (incorrectamente) por el algoritmo como *s-kin-less*; con el parche, la palabra se reduce (correctamente) a *skin*.

4.5 La Representación de Trigramas

En la implementación del agente *Querando!* se quiere probar la aplicación de las redes neuronales a la clasificación de documentos. En todo problema encarado mediante redes neuronales un paso importantísimo es el de elegir la representación de los datos de entrada. En esta sección se explica como se forman los vectores de características [6] para representar a los documentos a clasificar.

4.5.1 Formación de los Trigramas

En la IR tradicional, las representaciones de los documentos generalmete están compuestas por un conjunto de términos con un peso asociado [Frakes92a, FoxC92, Frakes92b, Girardi98a, Harman92a, Lewis92, Sebastiani98]. Esta representación no es adecuada para su uso con redes neuronales por los siguientes motivos:

1. A diferencia, la lista de términos que representa a una página no tiene asociada una consulta.

2. El uso de una red neuronal para clasificar documentos requiere que el vector de características del documento tenga tamaño constante; si bien alguna de las opciones mencionadas anteriormente cumplían ese requisito, entre dos vectores distintos (en el caso de [Balabanovic98]) no había relación ya que las claves con que estaba formado eran potencialmente distintas.

Por lo anterior, la representación que decidimos usar es la de trigramas; que está basada en [Hyötyniemi96] pero con modificaciones propias. Dado un documento D, su vector de características se obtiene mediante el siguiente algoritmo:

1. Eliminar todos los tags HTML de D para obtener D_1.

2. Eliminar todas las "stop words" de D_1 obteniendo D_2. Para realizar esto se utilizó la listas de la sección 4.3 (ver sección 3.8 para el tema *Stop Lists*).

[6] Conocidos como *feature vectors* en la literatura de IR.

3. Aplicar "stemming" a cada token de D_3 obteniendo D_4. Para este punto se utilizó la lista de sufijos de la sección 4.4 y se utilizó el algoritmo descripto en la misma sección (ver la sección 3.9 para el tema *Stemming*).

4. A partir de D_4, obtener el conjunto de contadores de trigramas que representan al documento a partir de la técnica siguiente: Sea $D_4 = a_1 a_2 \dots a_N$. Para cada tripla $a_i a_{i+1} a_{i+2}$ (con $i = 1, \dots, N-2$), obtener el valor $a = \rho(a_i) \cdot 27^2 + \rho(a_{i+1}) \cdot 27 + \rho(a_{i+2})$ e incrementar el contador a-ésimo, donde $\rho(\text{' '}) = 0$, $\rho(\text{'A'}) = 1$, $\dots$, $\rho(\text{'Z'}) = 26$.

   Nota: Hay dos variantes aquí, usando el *blanco* como parte de la representación y sin usarlo.

5. Para lograr el escalado de los valores del vector, se probaron dos variantes:

   (a) Escalar el vector de contadores dividiendo cada entrada por el valor de la entrada máxima; de esta manera, todos los valores quedan entre 0 y 1.

   (b) Escalar los elementos del vector usando una función sigmoidea.

6. Si el documento original D posee un tag `META` con información dependiente del lenguaje, obtener el vector de trigramas correspondientes a los trigramas de dichas palabras y setear en el vector de contadores normalizado dichas entradas a 1.

Otra representación posible se obtiene teniendo un vector de bits empaquetado donde si aparece el trigrama correspondiente en el documento, dicho bit se encuentra a 1 y si éste no aparece, este bit tiene el valor 0. De esta manera, esta representación permite saber que trigramas están presentes en el documento y, así, se tiene otra IREP.

En todos los puntos de la estrategia anterior se utilizaron las técnicas de autómatas finitos [Aho83a].

Es de notarse, que al usar un alfabeto de 26 símbolos (si no tenemos en cuenta al blanco) y de 27 símbolos (si tenemos en cuenta al blanco), el vector de características contiene codificadas todas las combinaciones de tres símbolos (trigramas) del alfabeto. La cantidad de combinaciones de tres símbolos del alfabeto es igual al cubo de la cantidad de símbolos del mismo; así, en ambos casos la cantidad de trigramas posibles ascenderá a $26^3 = 17576$ y $27^3 = 19683$, respectivamente.

El método de los trigramas para representar documentos puede verse como un caso particular de la representación de patrones temporales [Skapura96], en el que dos patrones de dos eventos que ocurren secuencialmente se concatenan en un patrón mayor. Así, un trigrama *xyz* puede leerse como: *existe una letra 'z' que aparece luego de una letra 'y' que aparece luego de una 'x'.*

4.5.2 Uso de la Información de Formato

El hecho de que las páginas poseen tags para formatear la información se usa en la generación de características de dos maneras: a través del uso de metadatos y del uso de títulos de páginas y de encabezados.

Uso de Metadatos

En *Querando!*, aquellos trigramas que compongan los términos especificados en el tag `META` (sección 4.1.2) correspondiente a la propiedad *keywords* en inglés, recibirán peso igual a 1.0, haciéndolos de esta manera, los trigramas más relevantes de la página.

Uso de Títulos y Encabezados

El buscador *Google* utiliza sólo los títulos y encabezados para hallar los términos de indexación de una página. También, utiliza los términos que aparezcan en el ancla de otro documento referenciando a la página que se está indexando [Brin2000].

En *Querando!*, se hicieron mediciones considerando agregar esta característica sin mayores resultados (ver mediciones con clustering en capítulo 6).

4.5.3 Uso de la Estructura de Hipertexto

Como el HTML permite encadenar en forma de digrafo a los documentos mediante el uso del tag `<A HREF='otroDoc.htm'>`, el agente permite especificar una profundidad de búsqueda en el digrafo para generar los trigramas correspondientes al documento raiz del mismo. Esto se hace en la creencia de que documentos relacionados concordarán en su grado de relevancia; es decir, se espera que documentos relevantes estén encadenados entre sí, y lo mismo para los irrelevantes.

4.6 Mediciones Preliminares

En esta sección se describen las mediciones iniciales para probar las virtudes y defectos de la representación de documentos mediante el uso de contadores de trigramas.

Estas mediciones tuvieron el objeto de determinar la matriz de similitud entre documentos de una base dada (para *matriz de similaridad*, ver sección 3.2.3) y diversas estadísticas sobre las IREPs propuestas para los documentos.

4.6.1 Medida de la Similitud Entre Documentos Usada

Al momento de decidir la representación se analizaron las siguientes opciones (ver sección 4.5):

1. El vector de contadores de trigramas escalado mediante la división de sus entradas por el máximo de los valores.

2. El vector de bits donde cada bit a 1 corresponde a la aparición de dicho trigrama en el documento representado.

Las medidas de similitud entre documentos usadas fueron la distancia euclídea en el primer caso y la distancia de Hamming en el segundo. Recordemos que la *distancia euclídea* entre dos vectores u y v se define como:

$$d_E(u,v) = \sqrt{\sum_{i=1}^{n} (u_i - v_i)^2} \tag{4.1}$$

Figura 4.5: Una vista del libro *Database Developer's Guide with Visual C++ 4, Second Edition.*

Y que la *distancia de Hamming* entre dos vectores binarios es igual a la cantidad de posiciones en las que difieren los valores de los bits; por ejemplo:

$$\begin{aligned} v_1 &= 00010010011101 \\ v_2 &= 00110001011101 \\ d_H(v_1, v_2) &= 3 \end{aligned} \tag{4.2}$$

4.6.2 Un Ejemplo del Programa de Mediciones

Las mediciones de esta sección se hicieron con el programa, de implementación propia, llamado *AnalizadorDeFeatures*. Aquí, se reproducirá un ejemplo de la salida del programa. Por una cuestión de extención, para el resto de los análisis sólo mostraré los resultados finales; los archivos completos de las mediciones se hallan en el CD-ROM en el directorio *Resultados*.

Los Documentos Usados

En las figuras 4.5 y 4.6 se puede ver parte dos documentos clasificados:

1. *vcg05.htm*: Este documento corresponde a un capítulo del libro *on-line* de bases de datos en C++ *Database Developer's Guide with Visual C++ 4, Second Edition.* [Jennings99].

2. *ch01.htm*: Este documento corresponde a un capítulo del libro *Using Java 1.1* [Weber97]

La Salida del Programa para Un Documento

Aquí, veremos la salida del programa para uno de los dos documentos del ejemplo. En la misma se pueden apreciar cuáles son los trigramas del mismo cuyas apariciones son no nulas (figura 4.6.2).

En la figura 4.6.2 (4.7), se pueden apreciar los siguientes datos:

Figura 4.6: Una vista del libro *Using Java 1.1.*

1. Que la cantidad de trigramas sea 19683, quiere decir que el *blanco* es parte del alfabeto.

2. El nombre del archivo procesado: *ch01.htm*.

3. El porcentaje de trigramas con cantidad de apariciones no nulas (11 %). El trigrama correspondiente a tres blancos, se anula a mano ya que siempre resulta el de más apariciones.

4. Luego, aparecen los trigramas con cantidad de apariciones no nulas (la tabla se muestra incompleta por cuestiones de espacio).

Matriz de Similitud de Documentos

Por otro lado, en la figura 4.8, se tienen unas mediciones correspondiendo a:

1. La distancia euclídea entre las IREPs normalizados de los documentos (0.431118).

2. El producto escalar de los vectores IREPs normalizados (0.907069) Este valor debe oscilar entre -1 y 1, donde el 0 corresponde a vectores ortogonales, el 1 a vectores paralelos con la misma dirección y el -1 a vectores paralelos con orientación opuesta. Aquí, se pretendía estudiar cuán cerca o lejos estaba un documento de otro, en la esperanza de que documentos conceptualmente parecidos estuvieran cerca y viceversa los conceptualmente distintos.

3. La distancia de Hamming entre IREPs (1351) con su porcentaje equivalente (6.8 %).

En la figura 4.8, se puede observar la salida para la matriz de similitud entre los documentos *ch01.htm* y *vcg05.htm*. Allí, se aprecian:

1. El indicador *(0,1)* indica que estamos en presencia de la entrada (0,1) de la matriz de similitud de los documentos[7].

[7]La diagonal de la matriz no se detalla, y sólo se muestra la matriz triangular superior, ya que la relación es simétrica.

```
ANALISIS DE DATOS
-----------------
Analisis de los datos del Archivo: c:/resultados2/xxx/p2sin.dat

Metodo: trigramas sin usar stop list ni stemming.

Datos Generales
---------------
La cantidad de trigramas es: 19683
La cantidad de bits de los trigramas en binario es: 629856
La cantidad de bits del vector de apariciones es: 19683

Datos de cada archivo individual:
---------------------------------


Nombre del Archivo: c:/resultados2/xxx/paginas2/CH01.HTM.
Fraccion de Trigramas Distintos de Cero: 0.110095.
Porcentaje de Bits de los Trigramas Distintos de Cero: 0.006367
Porcentaje de Bits de las apariciones de Trigramas Distinto de Cero : 0.110095
Conjunto de Trigramas con Cantidad de Apariciones No Nula
---------------------------------------------------------
   :    2029   a:     239   b:     183   c:      44   d:       8
  e:       8   f:      45   g:      10   h:     110   i:      98
  j:      41   k:       2   l:     127   m:      17   n:       7
  o:      12   p:     152   r:      36   s:      59   t:     441
  u:      16   v:      81   w:      57   y:      17  a :     232
 ab:      13  ac:       7  ad:       4  ag:       1  al:     115
 am:       2  an:     107  ap:     163  aq:       6  ar:      54
 as:      19  at:      29  au:       2  av:       7  aw:      11
 b :     122  ba:       3  be:      39  bi:      15  bl:       2
 bo:      17  br:      73  bu:      17  by:      36  c :       9
 ca:      38  ce:       5  ch:      19  cl:      50  co:     120
 cr:       2  cu:      11  cy:       3  da:       8  de:      37
 di:      22  do:      27  dr:      10  dt:       1  dy:       8
 ea:       8  ed:       8  eg:       2  ei:       1  em:       1
 en:      16  eq:       1  es:       2  ev:      10  ex:      53
 fa:       9  fe:      11  ff:       5  fi:      77  fo:      61
 fr:      14  ft:       1  fu:       6  ga:       6  gb:       1
 ge:       5  gi:       9  go:       7  gr:      10  gu:      19
 gw:       1  h :      36  ha:      14  he:      53  hi:       5
 ho:      17  hr:      58  hs:       2  ht:      67  hy:       4
 i :      67  id:       1  if:       9  ig:       2  il:       6
```

Figura 4.7: Salida de *AnalizadorDeFeatures* para *Ch01.htm* (sólo una parte).

```
Matriz de similitud
-------------------

-----------------------------------------------------
Procesando el par (0,1),
Archivo1: c:/resultados2/xxx/paginas2/CH01.HTM,
Archivo2: c:/resultados2/xxx/paginas2/vcg05.htm.
Distancia Euclidea Normalizada      : 0.431118
Producto Escalar Normalizado        : 0.907069
Distancia de Hamming de Contadores  : 7003 (0.011118)
Distancia de Hamming de Apariciones : 1351 (0.068638)

-----------------------------------------------------
```

Figura 4.8: Matriz de similitud para los *Ch01.htm* y *Vcg05.htm*.

2. La distancia euclídea entre los vectores normalizados de ambos documentos.

3. El producto escalar entre los vectores normalizados de ambos documentos.

4. La distancia de Hamming entre los vectores de apariciones de trigramas de los documentos.

4.6.3 Resultados de las Mediciones

En esta subsección se muestran los resultados de las mediciones iniciales para testear las bondades y/o defectos de la representación de los documentos usada.

Se hicieron varias mediciones. La primera medición corresponde a un conjunto de documentos relacionados entre sí. La segunda medición corresponde a documentos que no están relacionados entre sí. Por último, encontramos una medición en la que los documentos pertenecen a varias clases conceptuales y determinaremos si existen parámetros para poder diferenciar entre ellas a partir de los números obtenidos.

Medición 1: Una Medición con Una Única Clase Conceptual

En esta sección, se detalla una medición donde todos los documentos se hallan relacionados. Los documentos fueron tomados todos del mismo sitio de internet correspondiente a los PDA *Palm Pilot*[8]. Los documentos se hallan el directorio *resultados2/xxx/palm*. Las bitácoras completas de las mediciones se hallan en los archivos *resultados2/xxx/palm-sin.txt* y *resultados2/xxx/palm-con.txt*.

Los documentos de esta medición, con sus respectivas proporciones de trigramas no nulos para ambas mediciones aparecen en las tablas:

1. *Sin sacar stop words y sin stemming:* En la tabla 4.1, con $\overline{x} = 0.0678$, $\sigma_n = 0.00831$ y $\sigma_{n-1} = 0.0091$[9].

[8] http://www.palm.com/

[9] La expresión $\overline{x}$ corresponde a la media de la tabla mientras que σ_n y σ_{n-1} a los estimadores de la varianza[Devore95, Meyer].

Nro.	*Archivo*	*Trigramas distintos de 0*
0	palm6.htm	0.052
1	palm2.htm	0.063
2	palm3.htm	0.078
3	palm4.htm	0.071
4	palm5.htm	0.072
5	palm1.htm	0.071

Tabla 4.1: Documentos procesados sin stop list ni stemming del PDA *Palm Pilot* (medición 1).

2. *Sacando stop words y con stemming:* En la tabla 4.2, con $\overline{x} = 0.015$, $\sigma_n = 0.0039$ y $\sigma_{n-1} = 0.00428$.

Las matrices de similitud entre documentos diferenciadas para los diversos métodos de similitud se hallan en las tablas siguientes:

1. *Sin stop list ni stemming*:
   (a) Distancia euclídea entre vectores: tabla 4.3, con $\overline{x} = 0.13$, $\sigma_n = 0.0375$ y $\sigma_{n-1} = 0.0389$.
   (b) Producto escalar entre vectores: tabla 4.4; con $\overline{x} = 0.9857$, $\sigma_n = 0.006$ y $\sigma_{n-1} = 0.006$.
   (c) Distancia de Hamming entre vectores de apariciones: tabla 4.5, con $\overline{x} = 0.027$, $\sigma_n = 0.0057$ y $\sigma_{n-1} = 0.0059$.
2. *Con stop list y stemming*:
   (a) Distancia euclídea entre vectores: tabla 4.6, con $\overline{x} = 0.866$, $\sigma_n = 0.0939$ y $\sigma_{n-1} = 0.0972$.
   (b) Producto escalar entre vectores: tabla 4.7, con $\overline{x} = 0.6106$, $\sigma_n = 0.0840$ y $\sigma_{n-1} = 0.0869$.
   (c) Distancia de Hamming entre vectores de apariciones: tabla 4.8, con $\overline{x} = 0.6106$, $\sigma_n = 0.0840$ y $\sigma_{n-1} = 0.0869$.

Analizando los resultados obtenidos se puede observar lo siguiente:

1. La cantidad de trigramas no nulos pasa de un 6.7% a un 1.5% al pasar del método sin filtrado de stop words y stemming a aplicarlo.
2. En general, las tres medidas de distancias entre documentos aumentan al pasar de no usar stop list ni stemming a usarlos.
3. En el caso del producto escalar usando stop list y stemming siempre da mayor a 0.42.

Nro.	*Archivo*	*Trigramas distintos de 0*
0	palm6.htm	0.007
1	palm2.htm	0.014
2	palm3.htm	0.018
3	palm4.htm	0.015
4	palm5.htm	0.018
5	palm1.htm	0.018

Tabla 4.2: Documentos procesados con stop list y stemming del PDA *Palm Pilot* (medición 1).

	0	1	2	3	4	5
0		0.08	0.19	0.13	0.12	0.11
1			0.19	0.14	0.10	0.10
2				0.20	0.15	0.13
3					0.12	0.12
4						0.07
5						

Tabla 4.3: Matriz de similitud con distancia euclídea para las páginas de Palm Pilot (sin stop list ni stemming) (medición 1).

	0	1	2	3	4	5
0		0.99	0.98	0.99	0.99	0.99
1			0.98	0.98	0.99	0.99
2				0.97	0.98	0.99
3					0.99	0.99
4						0.99
5						

Tabla 4.4: Matriz de similitud con producto escalar para las páginas de Palm Pilot (sin stop list ni stemming) (medición 1).

	0	1	2	3	4	5
0		0.018	0.031	0.025	0.025	0.024
1			0.034	0.026	0.022	0.023
2				0.035	0.038	0.037
3					0.029	0.028
4						0.023
5						

Tabla 4.5: Matriz de similitud con distancia de Hamming para las páginas de Palm Pilot (sin stop list ni stemming) (medición 1).

	0	1	2	3	4	5
0		0.94	0.91	0.78	0.81	0.79
1			0.90	1.06	0.92	0.88
2				1.00	0.90	0.83
3					0.82	0.79
4						0.67
5						

Tabla 4.6: Matriz de similitud (con stop list y stemming) de Palm con distancia euclídea entre vectores (medición 1).

	0	1	2	3	4	5
0		0.55	0.58	0.69	0.66	0.68
1			0.59	0.42	0.56	0.60
2				0.49	0.59	0.65
3					0.65	0.68
4						0.77
5						

Tabla 4.7: Matriz de similitud (con stop list y stemming) de Palm con producto escalar entre vectores (medición 1).

	0	1	2	3	4	5
0		0.009	0.013	0.011	0.012	0.012
1			0.015	0.013	0.011	0.012
2				0.015	0.016	0.017
3					0.016	0.015
4						0.013
5						

Tabla 4.8: Matriz de similitud (con stop list y stemming) de Palm con distancia de Hamming entre vectores (medición 1).

4.6.4 Medición 2: Una Medición con Documentos no Relacionados

En esta subsección, mostraré los resultados correspondientes a una medida donde los documentos involucrados no tienen ninguna relación conceptual entre ellos.

La lista de documentos usados para esta medición son los siguientes:

- *palm1.htm*: Página sobre el PDA *Palm Pilot*.
- *CH01.HTM*: Capítulo de libro de *Java 1.1*.
- *vcg03.htm*: Capítulo de libro de bases de datos en C++.
- *high4.htm*: Página sobre la serie televisiva *Highlander*.
- *Breitling Produits7.htm*: Página sobre los relojes *Breitling*.

La proporción de trigramas no nulos, para los dos métodos de obtención de características, aparecen en las siguientes tablas:

1. *sin stop list ni stemming*: en la tabla 4.9, con $\overline{x} = 0.0864$, $\sigma_n = 0.037$ y $\sigma_{n-1} = 0.0414$;
2. *con stop list y stemming*: en la tabla 4.10, con $\overline{x} = 0.0334$, $\sigma_n = 0.0278$ y $\sigma_{n-1} = 0.0312$.

Las matrices de similitud correspondientes se encuentran en las tablas:

1. *sin stop list ni stemming:*
   (a) distancia euclídea entre vectores: en la tabla 4.11, con $\overline{x} = 0.367$, $\sigma_n = 0.1015$ y $\sigma_{n-1} = 0.1069$;
   (b) producto escalar entre vectores: en la tabla 4.12, con $\overline{x} = 0.92$, $\sigma_n = 0.0384$ y $\sigma_{n-1} = 0.0405$;
   (c) distancia de Hamming entre vectores: en la tabla 4.13, con $\overline{x} = 0.0777$, $\sigma_n = 0.0205$ y $\sigma_{n-1} = 0.0216$;
2. *con stop list y stemming:*
   (a) distancia euclídea entre vectores: en la tabla 4.14, con $\overline{x} = 1.321$, $\sigma_n = 0.0465$ y $\sigma_{n-1} = 0.04909$;
   (b) producto escalar entre vectores: en la tabla 4.15, con $\overline{x} = 0.114$, $\sigma_n = 0.0601$ y $\sigma_{n-1} = 0.0634$;
   (c) distancia de Hamming entre vectores: en la tabla 4.16, con $\overline{x} = 0.0486$, $\sigma_n = 0.02218$ y $\sigma_{n-1} = 0.0233$;

Analizando estos resultados vemos lo siguiente:

1. Sin stop list ni stemming, la distancia euclídea entre documentos tiene un mínimo de 0.20, lo cual hace suponer que cualquier documento con un valor mayor a ese, se podría considerar no relacionado.

Nro.	*Archivo*	*Trigramas distintos de 0*
0	palm1.htm	0.071
1	ch01.htm	0.110
2	vcg03.htm	0.147
3	high4.htm	0.048
4	Breitling Produits7.htm	0.056

Tabla 4.9: Documentos no relacionados procesados sin stop list ni stemming (medición 2).

Nro.	*Archivo*	*Trigramas distintos de 0*
0	palm1.htm	0.018
1	ch01.htm	0.049
2	vcg03.htm	0.081
3	high4.htm	0.009
4	Breitling Produits7.htm	0.010

Tabla 4.10: Documentos no relacionados procesados con stop list y stemming (medición 2).

	0	1	2	3	4
0		0.50	0.32	0.31	0.20
1			0.48	0.44	0.51
2				0.30	0.33
3					0.28
4					

Tabla 4.11: Matriz de similitud entre los documentos no relacionados sin stop list ni stemming usando distancia euclídea (medición 2).

	0	1	2	3	4
0		0.87	0.94	0.94	0.97
1			0.88	0.89	0.86
2				0.95	0.94
3					0.96
4					

Tabla 4.12: Matriz de similitud entre los documentos no relacionados sin stop list ni stemming usando producto escalar (medición 2).

	0	1	2	3	4
0		0.069	0.093	0.053	0.053
1			0.079	0.082	0.079
2				0.110	0.108
3					0.051
4					

Tabla 4.13: Matriz de similitud entre los documentos no relacionados sin stop list ni stemming usando distancia de Hamming (medición 2).

	0	1	2	3	4
0		1.25	1.31	1.33	1.36
1			1.22	1.35	1.35
2				1.32	1.36
3					1.36
4					

Tabla 4.14: Matriz de similitud de documentos no relacionados usando stop list y stemming con distancia euclídea (medición 2).

	0	1	2	3	4
0		0.21	0.13	0.11	0.06
1			0.24	0.07	0.08
2				0.11	0.06
3					0.07
4					

Tabla 4.15: Matriz de similitud de documentos no relacionados usando stop list y stemming con producto escalar (medición 2).

	0	1	2	3	4
0		0.042	0.070	0.022	0.021
1			0.065	0.049	0.047
2				0.078	0.076
3					0.016
4					

Tabla 4.16: Matriz de similitud de documentos no relacionados usando stop list y stemming con distancia de Hamming (medición 2).

2. Con stop list y stemming, la distancia euclídea, tiene un mínimo de 1.25 (mismas consideraciones).

3. Sin stop list ni stemming, el producto escalar sigue dando alto, es decir, existe gran parecido entre todos los documentos. Esta medida no sirve ya que hay entradas de la matriz con valores de 0.97, lo cual invalida cualquier criterio basado en esta medida bajo las condiciones establecidas.

4. Al usar producto escalar con stop list y stemming, obtenemos valores bajos (entre 0.06 y 0.24), lo cual es bueno, porque hace que esta medida sirva para discriminar documentos no relacionados.

5. En ambos casos, la distancia de Hamming no sirve para hacer discriminaciones.

Medición 3: Una Medición con Varias Clases Conceptuales

En esta medida, se quiere determinar si la representación es capaz de proporcionar algunos valores de entre los parámetros que se puedan usar como umbral para determinar cuándo dos documentos están relacionados conceptualmente, o, en otras palabras, los vectores que comprenden su representación pertenecen al mismo cúmulo (o clase o *cluster* –ver capítulo 5 sobre *métodos de* clustering de vectores).

Con respecto a cada documento en particular, también se midió la cantidad de trigramas con apariciones no nulas.

Con respecto a la matriz de similitud entre documentos, en esta medida, también se usaron las tres métricas usadas en las mediciones anteriores: la distancia euclídea entre documentos normalizados, el producto escalar entre documentos normalizados y la distancia de Hamming entre los vectores de apariciones de los trigramas.

La lista de documentos usados en esta medida aparece en la tabla 4.17. El detalle de la cantidad de trigramas con apariciones no nulas, para ambas métodos de obtención de la representación, se halla en las tablas 4.18 y 4.22.

Las categorías de documentos (subjetivas) son las siguientes:

1. *clase 1*: documentos 0, 1, 2, y 3;

2. *clase 2*: documentos 4 y 5;

3. *clase 3*: documentos 6, 7 y 8;

4. *clase 4*: documentos 9, 10 y 11.

Las tablas correspondientes a las matrices de similitud entre documentos se hallan en las tablas:

1. *sin stop list ni stemming:*

   (a) con distancia euclıdea: tabla 4.19;

   (b) con producto escalar: tabla 4.20;

   (c) con distancia de Hamming: tabla 4.21;

2. *con stop list y stemming:*

Nro.	*Archivo*	*Documento*	*Tema*
0	high4.htm	TV : The Sci-Fi Channel : Highlander: ...	La última película de *Highlander*.
1	peter2.htm	methosluvr's Methos page	Un personaje de la serie *Highlander*.
2	peter.htm	Peter Wingfiled Interview	Una entrevista a uno de los protagonistas de *Highlander*.
3	Marina's Highlander Page.htm	Marina's Highlander Page	Fanship de *Highlander*.
4	ch02.htm	Chapter 2	Capítulo de libro de *Java 1.1*.
5	ch01.htm	Chapter 3	Capítulo de libro de *Java 1.1*.
6	vcg03.htm	vcg03.htm	Capítulo de libro de bases de datos en C++.
7	vcg04.htm	vcg04.htm	Capítulo de libro de bases de datos en C++.
8	vcg05.htm	vcg05.htm	Capítulo de libro de bases de datos en C++.
9	palm2.htm	3Com/Palm Computing - E-mail Conduit Update	El PDA *Palm Pilot*.
10	palm3.htm	3Com/Palm Computing - Palm III Support	El PDA *Palm Pilot*.
11	palm1.htm	3Com/Palm Computing - Palm Desktop 3.0.1 Software	El PDA *Palm Pilot*.

Tabla 4.17: Lista de documentos de la medición 3.

(a) con distancia euclıdea: tabla 4.23;

(b) con producto escalar: tabla 4.24, y,

(c) con distancia de Hamming: tabla 4.25.

En esta medición, vemos que:

1. Las distancias euclídeas entre vectores aumentan al usar stemming mientras que disminuyen al no usarlo.

2. El producto escalar entre vectores da más alto al *no* usar stop list ni stemming que al usarlo.

3. La distancia de Hamming disminuye al usar stop list y stemming que cuando no los usamos. La explicación reside en el hecho de que la técnica del filtrado por stop list y stemming, al eliminar las palabras y terminaciones redundates, reduce la cantidad de trigramas con apariciones no nulas.

4. Con respecto a las matrices de similitud, se puede decir:

   (a) *Sin stop list ni stemming*:

      i. *Distancia euclídea*: El ruido aportado por los trigramas de las stop words no eliminadas se hace sentir en las entradas (6,7) y (7,8), las que, a pesar

Nro.	*Archivo*	*Trigramas distintos de 0*
0	high4.htm	0.048062
1	peter2.htm	0.069095
2	peter.htm	0.096835
3	Marina's Highlander Page.htm	0.049484
4	ch02.htm	0.136361
5	ch01.htm	0.110095
6	vcg03.htm	0.147183
7	vcg04.htm	0.149571
8	vcg05.htm	0.143576
9	palm2.htm	0.063253
10	palm3.htm	0.078088
11	palm1.htm	0.070823

Tabla 4.18: Estadísticas de la medición 3 (sin stop list ni stemming).

	0	1	2	3	4	5	6	7	8	9	10	11
0		0.36	0.73	0.30	0.58	0.50	0.30	0.59	0.32	0.34	0.33	0.32
1			0.67	0.32	0.59	0.47	0.37	0.57	0.36	0.32	0.37	0.32
2				0.78	0.68	0.70	0.70	0.62	0.61	0.78	0.82	0.78
3					0.63	0.51	0.33	0.61	0.36	0.24	0.20	0.20
4						0.34	0.59	0.40	0.52	0.61	0.67	0.62
5							0.49	0.46	0.43	0.50	0.55	0.50
6								0.55	0.16	0.35	0.35	0.33
7									0.44	0.60	0.66	0.61
8										0.37	0.39	0.36
9											0.19	0.11
10												0.13
11												

Tabla 4.19: Matriz de similitud de la medición 3 (sin stop list ni stemming). Las entradas corresponden a la distancia euclídea entre los vectores normalizados.

	0	1	2	3	4	5	6	7	8	9	10	11
0		0.94	0.73	0.95	0.83	0.90	0.95	0.83	0.95	0.94	0.94	0.95
1			0.78	0.95	0.82	0.89	0.93	0.84	0.93	0.95	0.93	0.95
2				0.69	0.76	0.76	0.78	0.81	0.82	0.70	0.67	0.70
3					0.80	0.87	0.95	0.81	0.94	0.97	0.98	0.98
4						0.94	0.82	0.92	0.86	0.81	0.78	0.81
5							0.88	0.89	0.90	0.87	0.85	0.87
6								0.85	0.99	0.94	0.94	0.95
7									0.90	0.82	0.78	0.81
8										0.93	0.92	0.94
9											0.98	0.99
10												0.99
11												

Tabla 4.20: Matriz de similitud de la medición 3 (sin stop list ni stemming). Las entradas corresponden al producto escalar de los vectores normalizados.

	0	1	2	3	4	5	6	7	8	9	10	11
0		0.05	0.07	0.05	0.10	0.08	0.11	0.11	0.11	0.05	0.06	0.05
1			0.06	0.05	0.09	0.07	0.10	0.10	0.09	0.06	0.06	0.06
2				0.07	0.07	0.06	0.08	0.08	0.08	0.07	0.08	0.07
3					0.10	0.08	0.11	0.11	0.11	0.05	0.06	0.06
4						0.06	0.07	0.06	0.06	0.09	0.09	0.09
5							0.08	0.07	0.07	0.07	0.08	0.07
6								0.06	0.06	0.10	0.10	0.09
7									0.04	0.10	0.10	0.09
8										0.09	0.10	0.09
9											0.03	0.02
10												0.04
11												

Tabla 4.21: Matriz de similitud de la medición 3 (sin stop list ni stemming). Las entradas corresponden a la distancia de Hamming (en proporción) entre los vectores de apariciones de trigramas.

Nro.	*Archivo*	*Trigramas distintos de 0*
0	high4.htm	0.009274
1	peter2.htm	0.015248
2	peter.htm	0.038405
3	Marina's Highlander Page.htm	0.010014
4	ch02.htm	0.071802
5	ch01.htm	0.049272
6	vcg03.htm	0.080564
7	vcg04.htm	0.075444
8	vcg05.htm	0.072713
9	palm2.htm	0.014224
10	palm3.htm	0.018207
11	palm1.htm	0.017695

Tabla 4.22: Estadísticas de la medición 3 (con stop list y stemming).

	0	1	2	3	4	5	6	7	8	9	10	11
0		1.32	1.20	0.93	1.33	1.35	1.32	1.33	1.33	1.31	1.31	1.33
1			1.11	1.30	1.23	1.27	1.30	1.21	1.19	1.23	1.21	1.27
2				1.16	1.26	1.29	1.31	1.25	1.22	1.26	1.25	1.30
3					1.35	1.34	1.35	1.36	1.34	1.35	1.36	1.36
4						0.82	1.23	1.14	1.10	1.19	1.24	1.22
5							1.22	1.17	1.10	1.27	1.27	1.25
6								1.13	0.90	1.28	1.31	1.31
7									0.79	1.20	1.25	1.25
8										1.19	1.21	1.22
9											0.90	0.87
10												0.84
11												

Tabla 4.23: Matriz de similitud de la medición 3 (con stop list y stemming). Las entradas corresponden a la distancia euclídea entre los vectores normalizados.

	0	1	2	3	4	5	6	7	8	9	10	11
0		0.12	0.28	0.56	0.11	0.08	0.12	0.11	0.11	0.14	0.14	0.11
1			0.38	0.14	0.24	0.19	0.15	0.26	0.29	0.23	0.27	0.19
2				0.32	0.20	0.15	0.13	0.21	0.25	0.20	0.21	0.15
3					0.07	0.10	0.08	0.07	0.10	0.08	0.07	0.06
4						0.65	0.23	0.34	0.39	0.29	0.22	0.25
5							0.25	0.31	0.39	0.19	0.19	0.21
6								0.36	0.59	0.17	0.14	0.13
7									0.69	0.27	0.21	0.21
8										0.29	0.26	0.25
9											0.59	0.61
10												0.64
11												

Tabla 4.24: Matriz de similitud de la medición 3 (con stop list y stemming). Las entradas corresponden al producto escalar de los vectores normalizados.

	0	1	2	3	4	5	6	7	8	9	10	11
0		0.020	0.038	0.014	0.068	0.049	0.078	0.072	0.070	0.020	0.022	0.022
1			0.036	0.018	0.063	0.046	0.074	0.067	0.065	0.021	0.023	0.022
2				0.036	0.056	0.049	0.070	0.058	0.058	0.037	0.037	0.038
3					0.067	0.048	0.078	0.071	0.069	0.019	0.022	0.021
4						0.049	0.061	0.046	0.048	0.061	0.061	0.059
5							0.065	0.054	0.053	0.043	0.045	0.042
6								0.056	0.054	0.072	0.071	0.070
7									0.038	0.065	0.063	0.062
8										0.062	0.062	0.061
9											0.015	0.012
10												0.017
11												

Tabla 4.25: Matriz de similitud de la medición 3 (con stop list y stemming). Las entradas corresponden a la distancia de hamming entre los vectores de apariciones.

de pertenecer a documentos en la misma clase, tienen valores más altos que los de algunas entradas de pares de documentos no relacionados. Los documentos 9, 10 y 11 están relacionados con valores bajos, lo cual es correcto ya que están en la misma clase. El 4 y el 5 también tienen valores bajos. Los documentos 0, 1, 2 y 3 tienen una relación muy débil a pesar de ser compañeros de clase. Existen pares como el (3,6) que dan con valores muy bajos, a pesar de no aparear a documentos de una misma clase.

ii. *Producto escalar*: Los documentos 9, 10 y 11 están bien apareados con valores cercanos a 1, lo mismo que el 4 y el 5 y los compañeros 6, 7 y 8. Los documentos de la primer clase (0, 1, 2 y 3) también tienen buenos valores. El problema aparece entre el documento 0 con todos los demás, quien muestra valores muy altos; también existen muchas entradas que relacionan –a través de valores cercanos a 1– a documentos que no están relacionados.

iii. *Distancia de Hamming*: En este caso, la distancia de Hamming devuelve valores por debajo de 0.07 en las entradas correspondientes a documentos relacionados, y valores más altos en los no relacionados salvo en la submatriz con filas 0 y 1 y columnas 9 a 11. Por lo tanto, esta medida no sirve.

(b) *Con stop list y stemming*:

i. *Distancia euclídea*: Salvo la entrada (0,1) que dio 1.32 y la (1,3) con 1.30, todos los pares de documentos relacionados tienen valores menores a 1.20 y los no relacionados, valores mayores a 1.20 (salvo el caso de la entrada (1,8) (donde el valor de la entrada es 1.19 y el error puede deverse al redondeo). Por lo tanto, esta medida parece correcta.

ii. *Producto escalar*: En este caso, los valores de las entradas dan mucho más bajos que el caso sin stop list ni stemming (la diferenciación entre vectores la aporta la eliminación del ruido). Los documentos 9, 10 y 11 tienen valores altos; el (6,7) y (7,8) también dan bien, asimismo el (4,5) –todos estos aparean a documentos en mismas clases–. Sin embargo, hay anomalías entre los primeros documentos y con el par (6,7). Por otro lado, para los pares de documentos no relacionados, vemos que las entradas son muy menores a 0.30 (valor que se podría tomar como umbral).

iii. *Distancia de Hamming*: La distancia de Hamming no es concluyente, ya que documentos no relacionados dan valores menores a 0.05 mientras que documentosrelacionados tienen valores mayores (lo opuesto a lo que debería pasar).

4.6.5 Discusión de los Resultados

Con respecto a los documentos vistos individualmente, se corroboran los resultados obtenidos por [Hyötyniemi96] (sección 5.6.8), quien dice que alrededor de sólo 5% de los trigramas de los documentos son no nulos. En la investigación realizada, se obtuvieron diferentes medias; sin embargo, el valor dado por [Hyötyniemi96] se verifica para los vectores de características de trigramas obtenidos sin aplicar stop list ni stemming, mientras que los valores para los vectores que usan stop list y stemming son mucho menores.

Con respecto a la similitud entre documentos, en el caso del análisis sin usar stop list ni stemming (a), los valores del producto escalar entre los vectores dan mayores a los valores de los productos obtenidos con los vectores generados usando stop list y stemming (b).

Además, en el caso de los vectores (a), los valores no permiten discriminar entre documentos relacionados y documentos no relacionados. Por otro lado, en el caso de las características obtenidas usando (b), el producto escalar entre vectores de documentos relacionados da mayor a 0.3, mientras que en el caso de vectores no relacionados da menor 0.24.

Estos hechos están causados por la presencia de los trigramas correspondientes a las stop words (en (a)), que además de agregar ruido a la representación, proporcionan la mayoría de los trigramas con apariciones no nulas, haciendo que todos los vectores se parezcan entre sí.

4.7 Resumen

Es este capítulo, se expusieron el método de obtención de características (o representación) de los documentos HTML en el agente *Querando!*, las mediciones preliminares hechas para determinar las bondades y/o defectos de dicha representación y se analizaron los resultados obtenidos.

Capítulo 5

Métodos de Clasificación

En este capítulo se detalla la teoría sobre los distintos métodos de clasificación de vectores que se pueden hallar en la literatura. Se describen el clasificador bayesiano, los algoritmos de clustering tradicionales, el razonamiento basado en memoria y el paradigma de redes neuronales. Finalmente, se discuten sus ventajas y desventajas con una orientación hacia su aplicación al filtrado y clasificación de páginas HTML.

5.1 Definición

Cuando se enfrenta un problema de clasificación, el primer paso consiste en encontrar una forma de representar los datos del mismo en la forma de vectores de características. Una vez que se tiene la representación de las instancias del problema, una buena heurística consiste en determinar una separación en clases de las mismas para poder determinar cuáles se parecen entre sí y hallar la topología del problema [Freeman93, Maravall94, Rao95, Skapura96].

Los algoritmos de clasificación realizan justamente la tarea de tomar un conjunto de vectores de características y particionarlos en clases.

Un algoritmo de clasificación puede ser *supervisado* o *no supervisado*. El aprendizaje es supervisado si se usa un criterio externo para comparar la salida del algoritmo. Si no se usa ningún criterio externo, se dice que el algoritmo es no supervisado [Rao95].

En este capítulo se explicarán varios métodos de clasificación de patrones basados en el paradigma conocido como *machine learning*. Éstos, a diferencia de los paradigmas de inteligencia artificial simbólica, como los sistemas de producción, los sistemas expertos, etc. [Ford93, Rich94, Winston94], no requieren una descripción axiomática de los datos de entrada por parte del usuario para lograr aprender cómo clasificarlos.

5.2 Clasificador Bayesiano

El *clasificador bayesiano* aparece como algoritmo de clasificación en varios agentes de filtrado de páginas Web [Balabanovic98, Billsus97, Pazzani97a, Pazzani97b] (secciones 9.3.8 y 9.3.13). Es un método de clasificación supervisado, ya que el aprendizaje se realiza con vectores de entrenamiento donde el usuario sabe a priori cómo están (o al menos desea) particionados. El modelo de recuperación de información probabilístico también está basado en el teorema de Bayes, lo que lo hace relevante a los contenidos de este trabajo de

grado.

5.2.1 Teorema de Bayes

El teorema de Bayes puede expresarse de la siguiente manera:

$$P(\alpha_i|X = x) = \frac{P(X = x|\alpha_i) \cdot P(\alpha_i)}{P(X = x)} \tag{5.1}$$

En donde cada miembro de la igualdad tiene el siguiente significado:

- $P(\alpha_i|X = x)$: Probabilidad de que un vector de características X pertenezca a la clase α_i.
- $P(X = x)$: Probabilidad de que dada α_i, el valor de la variable aleatoria X sea x. En otras palabras, es la función de densidad de probabilidad (fdp) de la clase α_i considerada como variable aleatoria.
- $P(\alpha_i)$: Probabilidad *a priori* de que se presente un elemento de la clase α_i.
- $P(X = x)$: Probabilidad *a priori* de que se presente un objeto a clasificar con un vector de características igual a X (considerado como un vector numérico concreto).

Según [Maravall94], este último factor puede despreciarse, ya que presenta el mismo valor para un conjunto de N clases, $\alpha_1, \alpha_2, \ldots, \alpha_N$, compitiendo por el vector X a clasificar.

Para más referencias sobre el significado de probabilidades simple y condicional se puede consultar [Devore95, Meyer].

El primer término de la ecuación 5.1 aporta la solución al problema de la clasificación: Dado un vector X, pertenecerá a la clase α_j si la probabilidad de que X pertenezca a dicha clase es maximal, es decir:

$$(\forall i, i = 1, 2, \ldots, N : i \neq j)((P(\alpha_j|X = x) > P(\alpha_i|X = x)) \Rightarrow (X = x \in \alpha_j)) \tag{5.2}$$

Basándose en el segundo miembro de la ecuación 5.1 (habiendo eliminado el factor de escala $p(X)$) se tiene una forma alternativa de clasificar un vector X:

$$(\forall j, j = 1, 2, \ldots, N : j \neq i)((P(X = x|\alpha_j) \cdot P(\alpha_j) > P(X = x|\alpha_i) \cdot P(\alpha_i)) \Rightarrow (X = x \in \alpha_j)) \tag{5.3}$$

5.2.2 Consideraciones sobre el Clasificador Bayesiano

El principal problema para el diseño del clasificador bayesiano es la estimación estadística de las funciones de densidad de probabilidad de las clases $P(X|\alpha_1), \ldots, P(X|\alpha_N)$ a partir de un conjunto de muestras de las respectivas clases $\alpha_1, \ldots, \alpha_N$ consideradas como variables aleatorias [Maravall94].

5.2.3 Clasificador Bayesiano bajo Distribución Normal

En el caso de clases con distribución Gaussiana o normal y vectores de características n-dimensionales, la fdp sigue la siguiente ley [Maravall94]:

$$p(X|\alpha_i) = \frac{1}{(2\pi)^{n/2}|C_i|^{1/2}} \cdot e^{\frac{1}{2}(X-m_i)^t C_i^{-1}(X-m_i)} \tag{5.4}$$

En la ecuación 5.4 m_i y σ_i son, respectivamente, el vector media o esperanza matemática y la matriz de covarianza de la clase α_i, considerada como una variable aleatoria. Por $|C_i|$ se denota el determinante de la matriz C_i.

Por definición,

$$m_i = E(X) = E(X_1 X_2 \cdots X_n)^t = (m_{i1} m_{i2} \cdots m_{in})^t \tag{5.5}$$

y,

$$C = E(X - m_i)(X - m_i)^t = E \begin{pmatrix} X_1 - m_{i1} \\ X_2 - m_{i2} \\ \vdots \\ X_n - m_{in} \end{pmatrix} |(X_1 - m_{i1})(X_2 - m_{i2}) \cdots (X_n - m_{in})| \tag{5.6}$$

Los elementos de la diagonal principal son las denominadas varianzas de los elementos del vector de características.

$$\sigma_i^2 = E(X_i - m_i)^2; i = 1, 2, \ldots, n \tag{5.7}$$

Los demás elementos de C, que es una matriz simétrica respecto de la diagonal principal, se denominan covarianzas.

El coeficiente de correlación r_{ij} entre dos elementos cualesquiera X_i y X_j del vector de características se define como la razón:

$$r_{ij} = \frac{C_{ij}}{\sigma_i \sigma_j} \tag{5.8}$$

siendo C_{ij} un elemento genérico de la matriz de covarianza y las desviaciones típicas de las características X_i y X_j respectivamente. Si el coeficiente de correlación r_{ij} es cero, se dice que las características X_i y X_j están incorreladas, lo cual siempre es deseable; es decir, interesa que las características se elijan de tal manera que la matriz de covarianza sea diagonal pura.

Por lo anterior usaremos como discriminantes las siguientes expresiones:

$$p(\alpha_i)p(X|\alpha_i) = p(\alpha_i)\frac{1}{(2\pi)^{n/2}|C_i|^{1/2}} \cdot e^{(-\frac{1}{2})(X-m_i)^t C_i^{-1}(X-m_i)} \tag{5.9}$$

para $i = 1, \ldots, N$.

Se puede demostrar [Maravall94] que, cuando las matrices de covarianza son iguales –una situación poco común pero que simplifica los algoritmos de aprendizaje tanto como la carga computacional–, las funciones estadísticas son lineales contra el caso más general en el que aparecen formas cuadráticas.

5.2.4 Reconocimiento con Aprendizaje en Condiciones Estadísticas

El enfoque anterior se basaba en que se abordaba la estimación de las probabilidades *a priori* $p(X|\alpha_i)$ y $p(\alpha_i)$, ahora vamos a atacar el problema de reconocimiento automático a través de la estimación de las denominadas probabilidades *a posteriori* $p(\alpha_i|X)$.

Partiremos de un conjunto de un conjunto de muestras de entrenamiento o aprendizaje por cada clase:

$$\{X_{11}, X_{12}, \ldots, X_{1P}\}, \ldots, \{X_{N1}, X_{N2}, \ldots, X_{NP}\} \tag{5.10}$$

cuya pertenencia es conocida, por lo tanto, se trata de un aprendizaje supervisado.

El aspecto clave en el diseño de los reconocedores estadísticos con aprendizaje estriba en partir de una expresión concreta para las probabilidades *a posteriori* $p(\alpha_i|X)$ que se van a estimar (o aprender) basándose en las muestras de entrenamiento (ecuación 5.10).

Por simplicidad, consideraremos aproximaciones lineales a las funciones de densidad de probabilidad *a posteriori* de $p(\alpha_i|X)$, es decir,

$$p(\alpha_i|X) \cong W_i^t \cdot X = W_1 X_1 + \cdots + W_N X_N + W_{N+1} \tag{5.11}$$

El siguiente paso es establecer un índice funcional de rendimiento del aprendizaje, es decir, un índice del error a minimizar. Para tal fin, el objetivo del proceso de aprendizaje será obtener unas funciones discriminantes lineales:

$$fd_i(X) = W_i^t \cdot X \cong p(\alpha_i|X) \tag{5.12}$$

tales que:

$$(fd_i(X) = 1 \Leftrightarrow X \in \alpha_i) \wedge (fd_i(X) = 0 \Leftrightarrow X \notin \alpha_i) \tag{5.13}$$

que son las probabilidades ideales $p(\alpha_i|X)$, puesto que esta probabilidad *a posteriori* será 1 cuando $X \in \alpha_i$ y será 0 cuando $X \notin \alpha_i$.

El *maestro* o supervisor del aprendizaje se va a modelar como una variable aleatoria Γ tal que:

$$(\Gamma_i(X) = 1 \Leftrightarrow X \in \alpha_i) \wedge (\Gamma_i(X) = 0 \Leftrightarrow X \notin \alpha_i); i = 1, \ldots, N \tag{5.14}$$

Obsérvese que existirá una variable aleatoria para cada clase.

En virtud del planteo anterior, se podrán definir varios índices del error. Por ejemplo, el índice del error en módulo:

$$J_i = E\{|\Gamma_i(X) - W_i^t \cdot X|\}; i = \{1, \ldots, N\} \tag{5.15}$$

o el índice del error cuadrático:

$$J_i = E\{|\Gamma_i(X) - W_i^t \cdot X|^2\}; i = \{1, \ldots, N\} \tag{5.16}$$

Para la minimización de los índices de error, se aplica una actualización recursiva de los coeficientes de las funciones discriminantes basadas en el gradiente de error:

$$W_i(k+1) = W_i(k) - \mu(k) \nabla_W J(W)|_{W(k)} \tag{5.17}$$

en donde $\mu(k)$ ahora exige unas propiedades más restringidas:

- $\lim_{k\to\infty} \mu(k) = 0$
- $\sum_{k=1}^{\infty} \mu(k) = \infty$
- $\sum_{k=1}^{\infty} \mu^2(k) < \infty$

Una función muy simple y utilizada que cumple con estas tres restricciones es:

$$\mu(k) = 1/k \tag{5.18}$$

El factor $\mu(k)$ sigue teniendo un papel fundamental en el rendimiento del algoritmo de la ecuación 5.17. Unos pequeños valores de μ producen una gran lentitud del proceso de aprendizaje, pero éste es más fiable. Inversamente, valores elevados de μ aceleran el proceso de cambio de los coeficientes de W, pero en detrimento de una mayor inseguridad en la convergencia hacia el mínimo [Maravall94].

El aprendizaje recursivo tomará la forma:

$$W_i(k+1) = \begin{cases} W_i(k) + \mu(k)X(k), & \text{si } W_i^t(k) < \Gamma_i(X(k)) \\ W_i(k) - \mu(k)X(k), & \text{si } W_i^t(k) > \Gamma_i(X(k)) \end{cases} \tag{5.19}$$

El desarrollo de cómo se llega a este resultado y su interpretación se encuentra en [Maravall94].

Comentarios sobre el Método de Aprendizaje

Se deben considerar tres aspectos prácticos [Maravall94]:

1. En primer lugar, se trata de un proceso de aprendizaje multiclase, es decir, se actualizan simultáneamente todas las funciones discriminantes $W_1, W_2, \ldots, W_N$ para cada muestra de entrenamiento.

2. Otro aspecto práctico es la elección y el manejo del conjunto de muestras de entrenamiento. Para garantizar una convergencia correcta del correspondiente algoritmo de aprendizaje, el número de muestras de entrenamiento debe estar repartido equitativamente entre clases. En cuanto a la secuencia de presentación de las muestras, lo mejor es hacerlo de manera intercalada y secuencial, es decir, ir presentando correlativamente una muestra de cada clase.

3. Un tercer problema práctico es la inicialización del vector de coeficientes $W_i(0)$ de cada clase. En el caso estadístico, no se dispone de una inicialización privilegiada, por lo que es más aconsejable comenzar por la llamada situación de indiferencia: $W_i(0) = (0 \cdots 0)^t$.

5.3 Algoritmos de *Clustering*

Los algoritmos de *clustering*[1] tienen la capacidad de creación de clases.

La única información que requieren los algoritmos de *clustering* es la definición previa del vector de características. Algunos de estos algoritmos (el k-medias por ej.), a lo sumo, necesitan conocer también el número de clases.

[1] Clustering: agrupación

Una vez establecido el vector de características, estos algoritmos reciben como entrada los objetos a clasificar (convertidos en vectores numéricos), de modo que, sin supervisión alguna, agrupan estos vectores en clases (o *clusters* o nubes). Por esta razón, se los denomina también algoritmos de clasificación autoorganizada.

Las técnicas de agrupación se utilizan cuando no existe un conocimiento suficiente de las clases en las que se pueden distribuir los objetos de interés. Según [Maravall94], se pueden aplicar cuando sí se conocen las clases, pero como herramienta de verificación de un sistema supervisado. En particular, si se introdujeran como datos de entrada del algoritmo de agrupación los vectores del conjunto de muestras que se emplearon para el diseño del correspondiente clasificador supervisado, entonces los resultados que se obtuvieran como salida del algoritmo de agrupación indicarán la calidad del vector de características. Si la agrupación efectuada por aquel coincidiera con las clases reales, entonces el vector estaría bien escogido; en caso contrario, solapamientos o agrupaciones incorrectas, se trataría de un vector de características con propiedades indeseables.

Según [Maravall94] citando a Momenan[2]:

> *"...estos métodos (no supervisados) se han usado como alternativa completa al diseño supervisado, aunque en general no dan lugar a mejores resultados, sino todo lo contrario."*

Los algoritmos de agrupación varían entre sí por el mayor o menor grado de reglas heurísticas que utilizan y por el nivel de procedimientos formales involucrados. Todos ellos se basan en el empleo sistemático de las distancias entre los vectores así como entre los clusters o grupos que se van haciendo y deshaciendo en el proceso de ejecución del algoritmo concreto. Habitualmente, se emplea la distancia euclídea entre vectores, es decir,

$$d_E(X_i, X_j) = \sqrt{\sum_{i=1}^{n}(X_{ik} - X_{jk})^2} \tag{5.20}$$

De menor a mayor complejidad, los algoritmos que vamos a analizar en esta sección son:

- distancias encadenadas (chain map);
- max-min;
- K-medias, e,
- ISODATA.

5.3.1 Algoritmo de las Distancias Encadenadas (*chain map*)

Definición

Es un algoritmo que no requiere ningún tipo de información *a priori* respecto a la distribución por clases de los objetos a clasificar. Aunque los resultados pueden no ser los

[2]Momenan, R. *et al. Characterization of Tissue from Ultrasound Images.* IEEE Control Systems Magazine, 1988.

óptimos es recomendable como procedimiento inicial para tantear la agrupación de los objetos.

Dados los vectores a clasificar $X_1, X_2, \ldots, X_P$, se escoge uno de ellos al azar, digamos el X_i, y se ordenan según la sucesión:

$$X_i(0), X_i(1), X_i(2), \ldots, X_i(k), X_i(k+1), \ldots, X_i(P-1) \tag{5.21}$$

en donde esta sucesión se ha formado de tal manera que el siguiente vector de la cadena es el más próximo al anterior. Es decir, $X_i(1)$ será el vector de la cadena más próximo al $X_i(0)$, el $X_i(2)$ será el más cercano a $X_i(1)$ y así sucesivamente. Nótese que la sucesión formada depende del elemento inicial.

A continuación, se calculan las distancias euclídeas relativas:

$$d_1, d_2, \ldots, d_k, \ldots, d_{P-1} \; donde \; d_k = \|X_i(k) - X_i(k-1)\|; k = \{1, \ldots, P-1\} \tag{5.22}$$

A partir de su representación gráfica (histograma), se pueden detectar fácilmente los clusters o clases. Una clase será detectada cuando se produzca un salto significativo en el valor de la correspondiente distancia euclídea.

Consideraciones sobre el Desempeño del Algoritmo

Según [Maravall94], el parámetro más delicado del algoritmo es el umbral de detección de una nueva clase. Un umbral excesivamente bajo dará lugar a clases ficticias, mientras que un umbral alto tenderá a agrupar en una misma clase a objetos de distintas clases.

En cuanto a la elección del primer elemento de la cadena, el algoritmo no puede ser considerado excesivamente sensible, salvo para distribuciones muy caprichosas.

5.3.2 Algoritmo Max-Min

Definición

Es un algoritmo heurístico que emplea como único elemento formal la distancia euclídea. Tampoco requiere ninguna información *a priori* respecto al número de clases existentes.

Partiendo del conjunto de muestras o vectores a agrupar, los pasos del algoritmo son los siguientes:

1. Se escoge al azar un elemento individual de los P disponibles, digamos el X_i, y se crea una primera clase α_1, luego se supondrá que $X_i \in \alpha_1$.

2. Se calculan las distancias euclídeas de los $P-1$ vectores no agrupados y se toma la máxima distancia, de tal manera que el elemento implicado produzca una segunda clase, es decir, X_j tal que $d_E(X_i, X_j)$ es máxima, luego $X_j \in \alpha_2$.

3. Se dispone ya de dos clases (α_1 y α_2) con sus correspondientes prototipos (X_1 y X_2). Ahora es necesario agrupar los restantes $P-2$ vectores, para ello se realizan dos operaciones a) y b):

   - *a)* Para cada vector X no agrupado se obtiene la pareja de distancias euclídeas $d_E(X, Z_1)$ y $d_E(X, Z_2)$, en donde se ha representado por Z_1 y Z_2 los prototipos de las clases α_1 y α_2 respectivamente. De cada $P-2$ parejas de distancias euclídeas previamente calculadas se toma la mínima distancia.

- *b)* Del conjunto de las $P-2$ distancias mínimas obtenidas en el paso anterior se toma la distancia máxima, digamos d_{MAX}. Si esta distancia es superior a una determinada fracción ϵ de la distancia $d_E(Z_1, Z_2)$ entre los prototipos de las dos clases previamente formadas, entonces se crea una tercera clase. Es decir, si $d_{MAX} > \epsilon \cdot d_E(Z_1, Z_2)$, entonces se crea una clase α_3; donde $0 < \epsilon < 1$. El prototipo (y único elemento hasta ahora) de α_3 es el elemento correspondiente a la distancia máxima d_{MAX}.

4. Quedan $P-3$ elementos por clasificar. El proceso ahora es similar; es decir, se calculan las tres distancias euclídeas de cada uno de los $P-3$ elementos X sin clasificar a los prototipos o centroides de α_1, α_2 y α_3 y se toma únicamente la mínima distancia de cada terna. A continuación, de entre las $P-3$ distancias mínimas se toma la máxima, de manera que si supera una fracción de la distancia media entre los prototipos de las clases formadas, se crea una nueva clase, α_4. Es decir:

   - $d^k_{MIN} = min\{d_E(X, Z_1), d_E(X, Z_2), d_E(X, Z_3)\}$ para k entre los $P-1$ elementos por clasificar.
   - $d_{MAX} = max\{d^1_{MIN}, d^2_{MIN}, \ldots, d^{P-3}_{MIN}\}$.
   - Si $d_{MAX} > \frac{1}{3}\epsilon[d_E(Z_1, Z_2) + d_E(Z_1, Z_3) + d_E(Z_2, Z_3)]$, entonces se crea α_4; donde $0 < \epsilon < 1$.

5. Se continúa este proceso hasta que no se cumpla la condición:

$$d_{MAX} > \epsilon \cdot distanciaMediaEntreClases; \tag{5.23}$$

   momento a partir del cual ya no es posible crear más clases.

6. Los elementos que queden por agrupar se asignan a la clase cuyo prototipo esté más cerca.

La explicación intuitiva de cómo opera este algoritmo se basa en la combinación heurística de distancias euclídeas mínimas y máximas. En efecto, como el algoritmo está orientado a crear (o no) una nueva clase en cada paso iterativo, se calculan las distancias mínimas a las clases ya existentes de tal manera que la máxima distancia entre ellas (que corresponde al elemento sin agrupar que está más alejado de una clase concreta, que a su vez es la clase más cerca a él) se compara con la distancia media entre las clases ya formadas para testear si es posible crear una nueva clase. En definitiva, se está comprobando la viabilidad de formar una nueva clase con un elemento lo suficientemente separado de las clases ya existentes.

En [Maravall94], se muestra un ejercicio numérico donde se desarrolla la ejecución de este algoritmo.

Consideraciones sobre el Desempeño del Algoritmo

Según [Maravall94], este algoritmo tiene el inconveniente de ser muy sensible al coeficiente ϵ. Una buena elección de ϵ es fundamental para el correcto funcionamiento de este procedimiento. No obstante, es posible ensayar sucesivos valores hasta que se obtenga una agrupación final correcta.

5.3.3 Algoritmo K-Medias

Definición

El nombre de este algoritmo hace referencia a que existen K clases o patrones, siendo necesario conocer el a priori el número de clases existentes.

Partiendo de un conjunto de objetos a clasificar $X_1, X_2, \ldots, X_P$, el algoritmo de las K-medias realiza las siguientes operaciones:

1. Establecido previamente el número de clases existentes, digamos K, se escogen al azar entre los elementos a agrupar K vectores, de forma que van a constituir los centroides (al ser los únicos elementos) de las K clases. Es decir:

$$\alpha_1 = \{Z_1(1)\}; \alpha_2 = \{Z_2(1)\}; \ldots; \alpha_K = \{Z_K(1)\} \tag{5.24}$$

   en donde el paréntesis indica el número de iteración del algoritmo.

2. Como se trata de un proceso recursivo con un contador n, en la iteración genérica n se distribuyen todas las muestras $\{X\}_{1 \leq j \leq P}$ entre las K clases, de acuerdo a:

$$(\forall i, j = 1, 2, \ldots, K : i \neq j)(X \in \alpha_j(n) \Leftrightarrow \|X - Z_j(n)\| < \|X - Z_i(n)\|) \tag{5.25}$$

   en donde se han indexado las clases (que son dinámicas) y sus correspondientes centroides.

3. Una vez distribuidos los elementos a agrupar entre las diferentes clases, es preciso recalcular o actualizar los centroides de las clases. El objetivo en el cálculo de los nuevos centroides es minimizar el índice de rendimiento siguiente:

$$Ji = \sum_{X \in \alpha_i(n)} \|X - Z_i(n)\|^2; i = 1, 2, \ldots, K. \tag{5.26}$$

   Este índice se minimiza utilizando la media muestral o aritmética de $\alpha_i(n)$:

$$Z_i(n+1) = \frac{1}{N_i(n)} \sum_{X \in \alpha_i(n)} X; i = 1, 2, \ldots, K \tag{5.27}$$

   siendo $N_i(n)$ el número de elementos de la clase α_i en la iteración n.

4. Se comprueba si el algoritmo ha alcanzado una posición estable. Es decir, si se cumple:

$$Z_i(n+1) = Z_i(n); i = 1, 2, \ldots, K \tag{5.28}$$

   Si se cumple, el algoritmo termina. En caso contrario, ir al paso 2.

En [Maravall94], se puede hallar una descripción con diagramas de flujo de este algoritmo.

Consideraciones con Respecto al Desempeño del Algoritmo

Según [Maravall94], el algoritmo de las K-medias es simple y extraordinariamente eficiente si el número de clases se conoce *a priori* con exactitud. Es decir, es muy sensible al parámetro K. Un valor de K superior al número real de clases dará lugar a clases ficticias, mientras que un valor de K inferior producirá menos clases de las reales.

Una forma de detectar una posible mala elección del parámetro K es analizar las dispersiones estadísticas de las clases formadas. Cuando estas dispersiones sean sensiblemente diferentes entre sí, siendo alguna de ellas muy elevadas con respecto a las demás, se puede sospechar que se ha manejado un valor de K bajo. Análogamente, si algunas distancias interclases (separación entre los centroides de dos clases) son muy pequeñas respecto a las demás distancias interclases, entonces es válido plantearse que el parámetro K introducido es alto.

Desgraciadamente, las dos situaciones anteriores pueden deberse exclusivamente a la naturaleza de las variables características escogidas.

5.3.4 Algoritmo ISODATA

Definición

ISODATA es el acrónimo de la definición en inglés: *Iterative Self-organizing Data Analysis Techniques*, al que se le ha agregado la letra A para conseguir una palabra fácilmente pronunciable [Maravall94].

Este algoritmo es una especialización del algoritmo de las K-medias. Otra característica es que su naturaleza interactiva obliga a que su implementación disponga de una interfaz compleja, ya que el usuario del programa debe tener una participación activa en su ejecución.

Los parámetros que maneja el algoritmo ISODATA son los siguientes:

- N_C: es el número actual (*i.e.*, el correspondiente a una iteración genérica del algoritmo) de *clusters* o grupos o clases que se han formado.
- K: es el número estimado de clases *a priori*.
- θ_N: número mínimo de miembros o elementos de una clase para constituirse como tal.
- θ_S: desviación típica máxima. Servirá para aplicar un criterio de división de un grupo o clase en dos, cuando la desviación típica del grupo sea superior a θ_S.
- θ_C: es un parámetro de unión de dos *clusters*. Se utilizará para comprobar si la distancia euclídea entre dos clusters es menor que θ_C, en cuyo caso serán dos grupos a fusionar.
- L: cuando en una iteración genérica del algoritmo haya más de una pareja de clases susceptibles de ser unidas, L limita el número de fusiones que pueden llevarse a cabo en esa iteración.
- I: número máximo de iteraciones que puede ejecutar el algoritmo.

Los pasos del algoritmo son los siguientes:

1. *Inicialización*: Se recomienda hacer $N_C = K$. En esta fase de inicialización se escogen aleatoriamente K elementos entre los P a clasificar, $X_1, X_2, \ldots, X_P$, formándose con cada uno de ellos un *cluster* inicial. Se tendrán entonces los $K = N_C$ primeros centroides: $Z_1, Z_2, \ldots, Z_{N_C}$.

2. *Distribución de los elementos entre las diferentes clases*: Los elementos o muestras, $X_1, X_2, \ldots, X_P$, se agrupan entre los N_C grupos ya formados según el principio de la mínima distancia euclídea, *i.e.*,

$$(\forall j, j = 1, 2, \ldots, P)(\forall i, i = 1, 2, \ldots, N_C)(X_j \in \alpha_i \Leftrightarrow \|X_j - Z_i\| \; es \; m\acute{i}nima) \quad (5.29)$$

3. *Eliminación de clases con un número insuficiente de miembros*: Se eliminan los *clusters* con un número de elementos inferior a θ_N. Luego, habrá que actualizar el parámetro N_C.

4. *Actualización de los centroides de las clases*: Se realiza calculando la media muestral de cada grupo o clase:

$$Z_i = \frac{1}{N_i} \sum_{j=1}^{N_i} X_j; \; i = 1, 2, \ldots, N_C \quad (5.30)$$

   donde N_i es el número de elementos de la clase α_i.

5. *Cálculo de la distancia euclídea media de cada cluster*: Para cada clase, se obtiene la distancia euclídea media de sus elementos respecto al correspondiente centroide:

$$\overline{D_i} = \frac{1}{N_i} \sum_{j=1}^{N_i} \|X_j - Z_i\|; \; i = 1, 2, \ldots, N_C \quad (5.31)$$

   Este parámetro da una medida de la dispersión de las muestras de cada clase respecto de su media y se utilizará posteriormente, junto con otras condiciones, para la división del grupo.

6. *Cálculo de la distancia media de todas las clases*: Es decir,

$$\overline{D} = \frac{1}{N_C} \sum_{i=1}^{N_C} N_i \overline{D_i} \quad (5.32)$$

7. *Comprobación de bifurcaciones*: En este paso se comprueba en primer lugar si se trata de la última iteración, en cuyo caso se hace $\theta_C = 0$ y se salta al paso 11.

   En segundo lugar, se realiza un *test* de posible *unión* de *clusters*, viendo si $N_C \geq 2K$, en cuyo caso se salta al paso 11.

   De no cumplirse la anterior condición, se prosigue con la secuencia natural que se describe a continuación, que consiste básicamente en efectuar un *test* de posible *división* de *clusters*.

8. *Cálculo del vector de desviaciones típicas de cada grupo*: Puesto que se trabaja con un vector de características n-dimensional, las clases presentarán un vector n-dimensional, según:

$$\sigma_i = \begin{pmatrix} \sigma_{i1} \\ \sigma_{i2} \\ \vdots \\ \sigma_{in} \end{pmatrix} \tag{5.33}$$

donde

$$\sigma_{ij} = \sqrt{\frac{1}{N_i} \sum_{k=1}^{N_i} (X_{kj} - Z_{ij})^2} \tag{5.34}$$

donde

- $i = 1, 2, \ldots, N_C$ (clases);
- $j = 1, 2, \ldots, n$ (características), y,
- $k = 1, 2, \ldots, N_i$ (elementos de la clase α_i).

9. *Obtención de las desviaciones típicas máximas de cada grupo*: En cada clase, se selecciona la componente mayor del correspondiente vector de desviaciones típicas, formándose el conjunto:

$$\{\sigma_{i_{max}} : 1 \leq i \leq N_C\} \tag{5.35}$$

10. *Posible división de clases*: Para una clase, digamos α_j, en que se cumple (condición *sine qua non*) que $\sigma_{j_{max}} > \theta_S$ y además se cumple una o las dos condiciones siguientes:

    (a) $D_j > \overline{D}$ y $N_j > 2(\theta_N + 1)$
    (b) $N_C \leq K/2$,

    entonces esta clase se divide en dos, siguiendo algún procedimiento de división que veremos más adelante.

    La condición 10a significa que la dispersión media de la clase α_j, candidata a dividirse en dos clases, es superior a la media de las dispersiones de todas las clases y, además, s que el número de sus elementos es al menos superior al doble de del número mínimo para formar un *cluster*.

    En cuanto al proceso de división de un grupo, existen varias posibilidades. Por ejemplo, crear dos nuevos centroides Z_j^+ y Z_j^- a partir del Z_j tales que todas las componentes de los nuevos centroides coincidan con las de Z_j excepto la componente con máxima dispersión, digamos la Z_K (siendo esta dispersión $\sigma_{j_{max}}$). Las correspondientes componentes de Z_{jk}^+ y Z_{jk}^- serán:

$$Z_{jk}^+ = Z_{jk} + \gamma\sigma_{j_{max}} Z_{jk}^- = Z_{jk} - \gamma\sigma_{j_{max}} \tag{5.36}$$

    donde $0 < \gamma \leq 1$.

    Con esta división, se pretenden redistribuir adecuadamente las muestras originales del *cluster* antes de la división entre los nuevos *clusters*. Otra alternativa para la división se basa en obtener las dos muestras de la clase α_j más alejadas entre sí

y con respecto al centroide. Si estas muestras se representan como Z_j^+ y Z_j^-, se escogen como los dos nuevos centroides:

$$Z_{j1} = \frac{Z_j^+ + Z_j}{2} Z_{j2} = \frac{Z_j^- + Z_j}{2} \tag{5.37}$$

11. *Cálculo de las distancias entre clases*: Para realizar la unión de dos clases, se calculan previamente todas las distancias entre parejas de *clusters*:

$$D_{ij} = D_{ji} = \|Z_i - Z_j\| \tag{5.38}$$

para $i = 1, 2, \ldots, N_C - 1$ y $j = i + 1, i + 2, \ldots, N_C$.

12. *Posible unión*: Se comparan estas distancias D_{ij} con el parámetro θ_C, de forma que se toman (si existen) las L más pequeñas en orden creciente:

$$D_1, D_2, \ldots, D_L \tag{5.39}$$

con $D_1 < D_2 < \ldots < D_L$.

13. *Proceso de unión*: Comenzando con las parejas de *clusters* con las distancias menores, se procede de la siguiente manera. Supongamos que se van a unir las clases i y j cuyas distancia D_{ij} se encuentra dentro del conjunto definido por la ecuación 5.39. Si, y sólo si, ninguna de estas dos clases ha sido previamente fusionada con otra en esta misma iteración, entonces se forma un *cluster* único cuyo centroide es:

$$Z_{ij} = \frac{1}{N_i + N_j}(N_i Z_i + N_j Z_j) \tag{5.40}$$

siendo N_i y N_j el número de muestras de los *clusters* α_i y α_j, respectivamente, antes de la fusión.

Lógicamente, con cada unión se actualiza el parámetro N_C. Puesto que un *cluster* se puede unir una sola vez en cada iteración, normalmente no se obtendrán L uniones en cada iteración.

14. *Comprobación de última iteración*: Se comprueba si se ha llegado a la última iteración I. En caso negativo se salta al paso 2 para comenzar una nueva iteración.

5.4 Algoritmos del Área de IR

5.4.1 Aplicaciones de *Clustering* a la Recuperación de Información

Eddie Rasmussen [Rasmussen92, pp. 420–421] enumera las maneras en que se ha usado el agrupamiento en el área de la recuperación de información. Por ejemplo, para:

1. Agrupar los documentos en base a su contenido (términos). Esto mejora la recuperación.

2. Para ver la estructura de un conjunto de documentos. Esto es comparable al uso de redes neuronales de Kohonen (sección 5.6.8) en los artículos de Zeller y *et. al.* [Zeller97] o el de Samuel Kaski [Kaski96], en campos totalmente disímiles como la robótica o la economía, respectivamente.

3. Agrupar los documentos en base a citas co-ocurrentes, lo que permite analizar la literatura de un campo en particular.

4. También, se pueden agrupar términos en base a los documentos en que co-ocurren.

5.4.2 Método de Pasada Simple

Definición

El *método de pasada simple* requiere que los datos se procesen sólo una vez. A continuación se describen los pasos del algoritmo citerasmussen:

1. Asignar el primer documento D_1 como representativo de C_1.

2. Para D_i, calcular la similaridad S con el representativo para cada *cluster* existente.

3. Si S_{max} es mayor que un valor umbral S_T, agregar el item al *cluster* correspondiente y recalcular el representativo del *cluster*; en otro caso, usar D_i para iniciar otro *cluster*.

4. Si queda algún D_i por clasificar, ir al paso 2.

Consideraciones Respecto al Desempeño del Algoritmo

A pesar de que este método tiene la ventaja de la simplicidad, se lo critica por su tendencia a producir *clusters* muy grandes y, además, porque su salida depende del orden en que se le presentan los documentos. Generalmente se lo usa para inicializar los métodos de reubicación [Rasmussen92].

5.4.3 Método de Reubicación

Definición

Los métodos de reubicación operan seleccionando una partición inicial del conjunto de datos y entonces moviendo items entre los *clusters* para obtener una partición mejorada. El algoritmo general es así [Rasmussen92]:

1. Seleccionar M representativos para los *clusters* o centroides.

2. For i=1 to N, asignar D_i al centroide más similar.

3. For j=1 to M, recalcular los centroides de los *clusters* C_j.

4. Repetir los pasos 2 y 3 hasta que no haya cambios (o sean muy pequeños) en la pertenencia a los *clusters* durante una pasada a través del archivo.

Consideraciones Respecto al Desempeño del Algoritmo

Primero, los requerimientos de tiempo y espacio del método de reubicación son menores al HACM, son los preferidos en la práctica y se usaron en el sistema SMART (nota al pie 30) en los primeros días de la IR.

Segundo, el método de reubicación es idéntico al método de las k-medias. Por la forma en que están planteados los algoritmos, se ve que M (en el primero) equivale a K (en el segundo). El criterio de convergencia en el método de realocación es que no haya cambio en la pertenencia a los clusters mientras que en el de las k-medias es que no haya cambios en los centroides. A menos que dos puntos de la población sean idénticos, estas dos condiciones son equivalentes.

5.4.4 Algoritmo General para HACM

Este es un método aglomerativo y se puede describir por [Rasmussen92]:

1. Identificar dos puntos más cercanos y combinarlos en un sólo *cluster*
2. Identificar y combinar los dos siguientes puntos más cercanos (tratando a los *clusters* existentes como puntos).
3. Si hay más de un *cluster*, ir al paso 1.

El tiempo de este algoritmo es $O(N^2)$ u $O(N^3)$ dependiendo de la implementación [Rasmussen92].

5.4.5 Método de Enlace Simple

El método de enlace simple une, en cada paso, el par más similar de objetos que no estén en el mismo *cluster*. Tiene tendencia a formar *clusters* grandes lo que lo hace adecuado para delinear *clusters* elipsoidales pero inadecuado para aislar *clusters* esféricos o pobremente separados [Rasmussen92, p.426]. En [Rasmussen92], se discuten varias implementaciones de este método, como ser:

- el algoritmo de Van Rijsbergen;
- el algoritmo SLINK, y,
- los algoritmos de *Minimal Spanning Tree*.

Algoritmo de Prim-Dijkstra

Un *minimal spanning tree* (MST) [Aho83b, pp. 233–240] es un árbol uniendo N objetos con $N-1$ conexiones tal que no hay ciclos y la suma de la $N-1$ disimilaridades es minimizada.

Los principios de construcción para MST son:

1. Cualquier punto aislado puede conectarse a un vecino más cercano.
2. Cualquier fragmento aislado (subconjunto de un MST) puede conectarse a un vecino más cercano con el enlace más corto disponible.

El algoritmo de Prim-Dijkstra consiste de una aplicación del principio 1, seguido de $N-1$ iteraciones del principio 2, tal que el MST crece al agregar un sólo framento:

1. Ubicar un punto arbitrario en el MST y conectar su vecino más cercano a él.
2. Encontrar el punto no en el MST más cercano a cualquier punto del MST y agregarlo al fragmento.
3. Si hay un punto fuera del fragmento, ir al paso 2.

La implementación en lenguaje C [Brokken95, Kernigham85, Kruglinski97] de este algoritmo se puede encontrar en [Rasmussen92, p. 433].

5.4.6 Método de Enlace Completo

El método de enlace completo utiliza el par menos similar entre cada par de *clusters* para determinar la similaridad *intercluster*; se lo llama de enlace completo porque todas las entidades en un *cluster* están enlazadas a otra con la mínima similaridad. Este método halla *clusters* muy ligados [Rasmussen92].

En [Rasmussen92] se analizan variantes de este método, como ser:

- el algoritmo CLINK de Defays, y,
- el algoritmo de Voorhees.

Método de Ward

El método de Ward sigue el algoritmo general HACM (sección 5.4.4), donde el *cluster* unido en cada etapa es aquel cuya unión minimiza la varianza. Cuando dos puntos D_i y D_j se unen, el incremento en la varianza I_{ij} está dado por:

$$I_{ij} = \frac{m_i m_j}{m_i + m_j} d_{ij}^2 \tag{5.41}$$

donde m_i es el número de objetos en D_i y d_{ij}^2 es el cuadrado de la distancia euclídea, dada por:

$$d_{ij}^2 = \sum_{k=1}^{L} (x_{ik} - x_{jk})^2 \tag{5.42}$$

donde D_i está representado por un vector $(x_{i1}, x_{i2}, \ldots, x_{iL})$ en un espacio L-dimensional. El centro del *cluster* para un par de puntos D_i y D_j está dado por:

$$\frac{m_i D_i + m_j D_j}{m_i + m_j} \tag{5.43}$$

Algoritmo del Vecino Recíproco Más Cercano

Las propiedades matemáticas del método de Ward lo hacen apropiado para usar el algoritmo del vecino recíproco más cercano (*reciprocal nearest neighbor*, RNN). Para cualquier punto de un *cluster*, existe una cadena de vecinos más cercanos (NN) tal que:

$$NN(i) = j; NN(j) = k; \ldots; NN(p) = q; NN(q) = p \tag{5.44}$$

El algoritmo basado en este proceso es:

1. Seleccionar un punto arbitrario.
2. Seguir la cadena NN desde este punto hasta hallar un RNN,
3. Unir esos dos puntos y reemplazarlos con un solo punto.
4. Si hay un punto en la cadena NN precediendo los puntos unidos, ir al paso 2; sino, ir al paso 1. Detenerse cuando quede un sólo punto.

5.4.7 Comparación de Métodos

Para comparar los métodos de clustering, hay que contestar dos preguntas [Rasmussen92]:

1. Cuál método de clustering es apropiado para un conjunto de datos particular.
2. Cómo se determina que los resultados obtenidos por el método de clustering elegido realmente caracterizan a los datos.

La respuesta a la primera pregunta está en los métodos de evaluación y la respuesta a la segunda está en los métodos de validación.

En general, no hay un método del área de IR que se pueda considerar *el método*. Según [Rasmussen92], el método de Ward es bueno; por otro lado, cuando los recursos disponibles demandan poco cálculo, una opción es el método de enlace simple.

5.5 Razonamiento Basado en Memoria

El *Razonamiento Basado en Memoria*[3] (MBR) es un algoritmo para capturar patrones del usuario [Lashkari97, Maes94]. El MBR está basado en los conceptos de situaciones y acciones. En [Lashkari97] el MBR se usa en el contexto de un agente de filtrado de correo electrónico (sección 9.3.1), se coteja el contenido del mensaje junto a información de contexto para representar las situaciones y el manejo de los mensajes por parte del usuario como las acciones.

Cuando el usuario toma una acción, ésta se aparea con la correspondiente situación, y el par situación-acción se graba en la memoria del agente. Por ejemplo, si el usuario lee un mensaje M, el par $\langle M', \text{acción-de-lectura} \rangle$ se memoriza, donde M' contiene detalles acerca del mensaje e información de contexto relevante (por ej., que M fue leido n-ésimo de un total de k mensajes no leidos). Cuando nuevas situaciones ocurren, son comparadas a las situaciones encontradas previamente. Después de obtener las situaciones más cercanas almacenadas en la memoria, el agente puede calcular una predicción para una acción en la nueva situación. Además, el agente puede calcular un grado de confianza en su predicción, considerando factores tales como el número de situaciones en su memoria y la proximidad de las situaciones recuperadas.

[3]Memory Based Reasoning.

5.6 Redes Neuronales

En la literatura, podemos encontrar varios agentes de filtrado de información cuyo algoritmo principal está basado en usar alguno de los modelos de redes neuronales existentes [Bigus98, Hyötyniemi96]. Por ello, en esta sección se explicará la teoría necesaria para entender en qué consiste el paradigma de redes neuronales, los modelos de redes neuronales aplicados a la clasificación de patrones, y el modelo usado en la implementación del agente *Querando!*.

5.6.1 Definición de Red Neuronal

Una *red neuronal* es un conjunto interconectado de elementos de procesamiento simples, llamados *nodos* o unidades, cuya funcionalidad está débilmente basada en la de la neurona animal. La habilidad de procesamiento de la red está almacenada en las unidades de interconexión o *pesos*, obtenidos a través de un proceso de adaptación a (o *aprendizaje* de) un conjunto de patrones de entrenamiento [Gourney99, p. 1].

5.6.2 Comparación con las Soluciones Convencionales

En esta sección se comparan las soluciones tradicionales procedurales con las realizables mediante el paradigma de redes neuronales.

Las máquinas von Neumann

Las computadoras convencionales –o de von Neumann– pueden modelarse como una unidad central de procesamiento (CPU) y una memoria donde residen los datos y los programas, dicha computadora ejecuta repetidamente el ciclo *fetch-decode-execute* en el que se toma y ejecuta una instrucción de la memoria de programas y modifica (tal vez) la memoria de los datos. En dichas computadoras, la solución a un problema determinado puede describirse como un *algoritmo*, es decir, una receta definida que garantiza hallar la respuesta al problema.

En particular, los programas manipulan cadenas de símbolos o *strings* que pueden verse como la representación de ideas o conceptos. La inteligencia artificial tradicional (o de enfoque simbólico) tuvo como objetivo primigenio representar el conocimiento humano de esta manera para poder manipularlo en una computadora convencional.

Las máquinas von Neumann tienen unas características bien definidas [Gourney99]:

- La máquina requiere un *programa*. Es decir, el programador debe decirle por adelantado la serie de pasos a seguir y contemplar todas las posibles situaciones. Si algo falla, es el programador humano quien tiene que arreglar el programa.
- Los datos manejados por la máquina deben estar en un formato preciso. Es decir, los datos que contengan *ruido* confundirán a la máquina[4].
- El hardware se degrada fácilmente. Esto es, el programa se *colgará* al corromper unas pocas direcciones de memoria.

[4]El ruido es una perturbación o desviación del valor esperado [Rao95]

- Existe una correspondencia correspondencia entre el objeto semántico manejado por el programa y el hardware subyacente. Es decir, cada dato está almacenado en una dirección de memoria particular.

El éxito del enfoque tradicional depende de la existencia de una solución (o algoritmo) a un problema. Entonces, la fortaleza del enfoque es también su debilidad ya que hay problemas que resolvemos diariamente, como reconocer una cara de un amigo que hace una década que no vemos (aún, tal vez, en un ángulo que nunca contemplamos antes) en medio de una multitud que pueden especificarse mediante una algoritmo con gran dificultad.

En cambio, las redes neuronales requieren que el desarrollador de aplicaciones especifique el algoritmo de aprendizaje y defina una interpretación para las señales que van a propagarse por la red y provea un conjunto de patrones específicos de la aplicación que colectivamente representen la conducta deseada de la red [Skapura96, p. 2]. Los detalles de cómo un patrón de entrada se transforma en una salida están ocultos en la red misma. La red tiene una arquitectura *adaptativa*. La red aprende a través de la modificación de sus pesos o valores contenidos en la estructura interna. Con cada modificación, se modifica el conocimiento contenido en la red. De ahí, la naturaleza adaptativa de las redes neuronales [Skapura96].

Entre las similitudes entre el enfoque tradicional y el de las redes neuronales podemos decir que:

- Los detalles de cómo se resuelve un problema están ocultos en ambas soluciones. En la escuela tradicional, los detalles están ocultos en el algoritmo. En la escuela de redes neuronales, los detalles están ocultos dentro de la red.

Características de las Redes Neuronales

Las redes neuronales tienen las siguientes cualidades [Gourney99]:

- El estilo de procesamiento es distinto, es orientado a las señales en vez de a los símbolos. La combinación de señales para producir otras se contrasta con la ejecución de instrucciones para modificar la memoria.

- La información se guarda en los pesos en vez de en programas. Los pesos se adaptan cuando se le presentan patrones de entrenamiento a la red.

- Las redes son robustas en presencia de ruido: unos cambios pequeños en la entrada no afectarán dramáticamente la salida.

- Las redes son robustas ante fallas en el hardware: un cambio en un peso afectará sólo a un número pequeño de patrones.

- Se pueden representar conceptos de alto nivel como un patrón de actividad entre varios nodos en vez de como los contenidos de una porción de memoria.

- La red puede manejar patrones 'no vistos' y generalizar a partir del conjunto de entrenamiento.

- Las redes son buenas en tareas de percepción y recuerdo por asociación. Estas son las tareas en las que tiene problemas el enfoque simbólico.

5.6.3 Una Neurona Simple

Las neuronas son los componentes fundamentales de las redes neuronales. Las neuronas se combinan para formar una red neuronal que puede verse como un grafo dirigido pesado; allí, las neuronas serán los nodos y las conexiones entre ellas las aristas del grafo. Los pesos de las aristas conforman los pesos de las conexiones, los cuales contienen el conocimiento de la red. La forma en que se disponen dichas neuronas (sumado a la forma de entrenar los pesos) se denomina *topología* de la red neuronal. Generalmente, las neuronas se agrupan en conjuntos denominados *capas*.

El procesamiento de señales de una neurona está basado débilmente en las neuronas naturales. En las neuronas naturales, las señales se transmiten como pulsos eléctricos que salen por el axón y llegan a las otras neuronas a través de las dendritas. Este proceso se llama *sinapsis*.

El procesamiento de señales de una neurona puede resumirse como sigue: las señales aparecen en las entrada s de las unidades. El efecto de cada señal puede aproximarse multiplicando la dicha señal por algún número o *peso* para indicar la fuerza de la sinapsis. Las seãles pesadas se suman para producir la *activación* de la unidad. Si esta activación excede un cierto umbral, la unidad produce una respuesta de salida. Esta funcionalidad se describe en la TLU de McCulloch y Pitts.

Suponemos que hay n entradas con señales $x_1, x_2, \ldots, x_n$ y pesos $w_1, w_2, \ldots, w_n$. Las señales toman los valores '1' o '0' sólamente. Por eso se dice que las señales tienen valores *booleanos* (hay otros modelos en que las seãles tienen valores analógicos). La activación a está dada por:

$$a = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n \tag{5.45}$$

Esto se puede representar más compactamente como

$$a = \sum_{i=1}^{n} w_i x_i \tag{5.46}$$

La salida y está dada por la función de umbral:

$$y = \text{if } a \geq \theta \text{ then 1 else 0 fi} \tag{5.47}$$

En general, la ecuación 5.46 puede interpretarse como una medida de la distancia entre el patrón de entrada y el vector de pesos. Así, esta ecuación puede interpretarse como el producto interno o escalar entre el vector de pesos w y el vector de entrada x, como sigue:

$$\begin{aligned} a &= \sum_{i=1}^{n} w_i x_i && (5.48)\\ &= w \cdot x && (5.49)\\ &= \|w\|\|x\| \cos\alpha && (5.50) \end{aligned}$$

donde α es el ángulo entre los vectores w y x. Por lo tanto, cuanto más pequeño sea el ángulo α, más grande será el producto escalar 5.49.

5.6.4 Perceptrón Binario

El *Perceptrón Binario* funciona como una única neurona. Es exactamente la neurona simple de la sección 5.6.3.

5.6.5 Adaline y Madaline

Adaline es un acrónimo que significa *elemento lineal adaptativo*[5] inventado por Bernard Widrow y Marcian Hoff. Es similar a un perceptrón. Las entradas son valores reales en el intervalo $[-1, 1]$ y el aprendizaje se basa en el criterio de minimizar el error cuadrático medio. Adaline tiene una gran capacidad para almacenar patrones. El adaline tiene entrenamiento supervisado [Freeman93].

El entrenamiento del Adaline tiene como objetivo calcular la ecuación de hiperplanos que dividen el hiper-espacio. Dado un patrón de entrada, esta red permite determinar de qué lado del hiperplano se halla dicho patrón [Rao95, Wasserman89].

Madaline quiere decir *muchas Adalines*. Los hiperplanos que define cada Adaline sirven para dividir el hiperespacio en regiones. Así, dado un patrón de entrada, la combinación de varias Adalines en una red neuronal permite determinar entre qué región del espacio se halla el patrón [Wasserman89].

5.6.6 Backpropagation

El algoritmo de *propagación hacia atrás*(BPN) para entrenar redes alimentadas hacia adelante fue desarrollado por Paul Werbos y, luego, por Parker y Rummelhart y McClelland. Este tipo de red es la más común debido a la facilidad de su entrenamiento. Se estima que el 80% de las aplicaciones de redes neuronales están basadas en este modelo [Rao95, cap. 5].

Aprendizaje en la BPN

Hay dos fases en el ciclo de aprendizaje de la propagación hacia atrás: uno para propagar el patrón por la red y otro para adaptar la salida cambiando los pesos de la red. Es la señal de error lo que se propaga en la operación de la red a las capas ocultas. La porción del error que recibe una neurona de una capa oculta es proporcional a su contribución al error en la salida. Basado en esto, el error cuadrático (u otra métrica) se trata de minimizar para calcular el valor final de los pesos, si esto fuera posible [Freeman93, Rao95, Wasserman89].

Esquemáticamente el algoritmo es como sigue [Skapura96, p. 31]:

La BPN se inicializa aleatoriamente para no imponer nuestros prejuicios sobre la aplicación. Luego, se toma una muestra representativa que será aprendida por la red. Sea la muestra $\{(x_1, y_1), (x_2, y_2), \ldots (x_p, y_p)\}$ donde cada x_i representa un vector de entrada y cada y_i representa la salida esperada para dicha entrada. Los pasos a seguir son:

1. Seleccionar el primer par de vectores de entrenamiento de la muestra, llamemos (x, y) a este par.

2. Use el vector de entrada, x, como la salida de la capa de entrada.

3. Use la ecuación 5.51 para calcular la activación de cada unidad sobre la capa siguiente.

$$net_i(t) = \sum_{j=1}^{n} w_{ij}(t) o_j(t) \tag{5.51}$$

[5] Adaptive linear element.

4. Aplicar la función de activación apropiada[6], $f(net^h)$ para la capa oculta y $f(net^o)$ para la capa de salida, a cada unidad de la capa subsiguiente. *Apropiada* se refiere a la función que esté más adaptada a la función a realizar por esta capa de neuronas.

5. Repetir los pasos 3 y 4 para cada capa de la red.

6. Calcular el error, δ^o_{pk}, para este patrón p a través de las K unidades de la capa de salida usando la ecuación 5.52.

$$\delta^o_{pk} = (y_k - o_k) f'(net^o_k) \tag{5.52}$$

7. Calcular el error δ^h_{pj}, para todas las J unidades ocultas usando la ecuación 5.53.

$$\delta^h_{pj} = f'(net^h_j) \sum_{k=1}^{K} \delta^o_{pk} w_{kj} \tag{5.53}$$

8. Actualizar los valores de los pesos de conexión a la capa oculta de acuerdo a la ecuación 5.54.

$$w_{ji}(t+1) = w_{ji}(t) + \eta \delta^h_{pj} x_i \tag{5.54}$$

   donde η es la tasa de aprendizaje.

9. Actualizar los valores de los pesos a la capa de salida de acuerdo a la ecuación 5.55.

$$w_{kj}(t+1) = w_{kj}(t) + \eta \delta^o_{pk} f(net^h_j) \tag{5.55}$$

10. Repetir los pasos 2 a 9 para todos los pares de vectores en el conjunto de entrenamiento. Llamar a este entrenamiento *época* [Skapura96].

11. Repetir los pasos 1 a 10 para tantas épocas como tome reducir la suma del error cuadrático a un valor mínimo. El cálculo del error cuadrático se realiza sólo en las unidades de la capa de salida, sobre todos los P patrones de entrenamiento, de acuerdo a la ecuación 5.56.

$$E = \sum_{p=1}^{P} \sum_{k=1}^{K} (\delta^o_{pk})^2 \tag{5.56}$$

Consideraciones sobre el Aprendizaje

En esta red el entrenamiento es supervisado para saber cuál fue el error cometido por la red. La salida esperada es comparada contra la salida obtenida para obtener una medida del error.

También, los patrones de entrenamiento deben presentársele varias veces a la red. Si quisiéramos que esta red aprendiera a reconocer fotos de las letras del abecedario, hay que impedir que olvide la A en su esfuerzo de aprender la B.

[6]La función de activación puede ser lineal (de escalón) o sigmoidea [Freeman93, Rao95]. La lineal sirve para interpolar puntos linealmente y la sigmoidea para interpolar puntos con una curva.

5.6.7 Contrapropagación

La *red de contrapropagación* (CPN), definida por Robert Hecht-Nielsenm es una red que aprende un *mapping* bidireccional en un espacio hiperdimensional; es decir, aprende un mapeo (de un espacio n-dimensional en un espacio m-dimensional) (y si existe, aprende también el inverso) para un conjunto de patrones [Freeman93, Rao95, Skapura96, Wasserman89].

La arquitectura de la CPN está compuesta de tres capas de neuronas: una de *fan-in*, una capa competitiva y una capa de *fan-out*.

Aprendizaje en la CPN

Inicialmente, una CPN construida para desarrollar una aplicación específica no sabrá nada acerca de la aplicación: Los pesos de la red se hallarán en un estado inicializado[7]. El proceso de aprendizaje comienza presentando un conjunto de patrones de entrenamiento que colectivamente definen la aplicación como un todo. Definiremos el conjunto de patrones como un conjunto de pares de vectores $\{(x_1, y_1), (x_2, y_2), \ldots (x_p, y_p)\}$. La CPN aprende a producir y_i dado x_i adaptándose de acuerdo al siguiente algoritmo [Freeman93, Skapura96]:

1. Selecciona aleatoriamente un par de entrenamiento (x_i, y_i) del conjunto de patrones. Llamemos (x, y) a este par de vectores.

2. Normalizar el vector x dividiendo cada una de sus componentes por la norma del vector x, donde la norma de x se define como en la ecuación 5.57, obteniendo el vector I:
$$\|x\| = \sqrt{\sum_{j=1}^{n} x_j^2} \tag{5.57}$$

3. Usando al vector normalizado como la salida de la capa de entrada en la CPN, calcular la activación de cada unidad en la capa competitiva de acuerdo usando el producto escalar de acuerdo a la ecuación 5.58.
$$activacion_i = w_{in} \cdot I \tag{5.58}$$

4. Determinar la unidad ganadora de la capa competitiva seleccionando la unidad con la activación mayor. Llamemos k a esta unidad.

5. Ajustar los pesos de conexión entre la unidad ganadora y todas las n entradas de acuerdo a la ecuación 5.59.
$$w_{k_n}^{in}(t+1) = w_{k_n}^{in}(t) + \alpha(x_n - w_{k_n}^{in}(t)) \tag{5.59}$$
donde α es la tasa de aprendizaje (un valor pequeño usado para limitar la magnitud del cambio en los pesos).

6. Repetir los pasos 1 a 5 hasta que todos los patrones p hayan sido procesados una vez.

[7] Generalmente en un valor aleatorio.

7. Repetir el paso 6 hasta que cada patrón sea asociado consistentemente con la misma unidad.

8. Seleccionar el primer par del conjunto de entrenamiento, sea éste el patrón corriente.

9. Repetir los pasos 2 a 4 para el patrón corriente.

10. Ajustar los pesos entre la unidad ganadora y las m unidades de la capa de salida de acuerdo a la ecuación 5.60.

$$w_{k_n}^{out}(t+1) = w_{k_n}^{out}(t) + \beta(x_n - w_{k_n}^{out}(t)) \tag{5.60}$$

donde β es la tasa de aprendizaje para esta capa.

11. Repetir los pasos 9 y 10 para cada vector en el conjunto de entrenamiento.

12. Repetir los pasos 8 a 11 hasta que la diferencia entre la salida deseada y el vector de pesos w^{out} se reduzca a un valor pequeño aceptable.

Consideraciones sobre el Aprendizaje

Inspeccionando el algoritmo, se observa que los vectores son normalizados antes de propagarlos por la capa competitiva. La razón de esta normalización recién se hace evidente al considerar cómo trabaja la capa competitiva. La red competitiva trabaja calculando el *producto escalar o interno* entre vectores (ecuación 5.58); esta ecuación dice que cada unidad va a recibir un estímulo que corresponde exactamente al coseno del ángulo en el espacio Euclídeo entre el vector de pesos y el vector de entrada normalizado. Así, en la CPN, las unidades de la capa oculta compiten por su derecho a adaptar sus pesos basadas en cuán parecidas son al vector de entrada sólo en dirección [Skapura96, p. 45].

Los pesos de la capa competitiva a la de salida se entrenan de la misma manera pero sólo después de haber entrenado la capa competitiva; esto es así para asegurar que sólo se entrena a la neurona que corresponde [Freeman93, Skapura96]. La capa de salida codifica así un promedio de todas los y_i de la muestra tales que x_i perteneciera al mismo cluster.

De esta manera, la red aprende a hacer clasificación de patrones; es de esperar que cada neurona de la capa competitiva codifique la posición del centroide de la clase correspondiente.

Sin embargo, la CPN adolece de algunos problemas causados por el mismo algoritmo de entrenamiento. Cuando el espacio de los vectores de entrada ocupa una pequeña región del espacio, puede ocurrir que siempre la misma unidad competitiva resulte ganadora (*problema del vector pegado* [Freeman93]), inhibiendo de esta manera el aprendizaje; existen soluciones que agregan un componente aleatorio antes de entrenar a una neurona para solucionar este problema [Freeman93].

5.6.8 Mapa Autoorganizativo de Kohonen

En esta sección, se discute un tipo de método de aprendizaje no supervisado llamado *mapa autoorganizativo de Kohonen* (KSOM)[8] Como el aprendizaje es no supervisado,

[8]El KSOM también es conocido como *mapa de características de Kohonen.*

a la red no se le presenta ninguna salida esperada; por el contrario, esta red es capaz de inferir relaciones entre los datos de entrada. El KSOM es un invento de Teuvo Kohonen[Freeman93, Rao95, Wasserman89].

Aprendizaje Competitivo en el KSOM

Un KSOM puede usarse por él mismo o como una capa de otra red neuronal (por ej, una CPN sección 5.6.7). El KSOM es un caso de la estrategia *"el ganador se lo lleva todo"*[Freeman93, Rao95].

El *aprendizaje competitivo* es una proceso adaptativo en el cual las neuronas de una red neuronal gradualmente se vuelven sensibles a diferentes categorías de entradas o conjuntos de muestras de un dominio específico del espacio de entradas [Kaski97, p. 21]. Así, surge una división del espacio de entradas cuando neuronas diferentes se especializan en representar diferentes tipos de entradas.

La especialización es reforzada por medio de la competición entre las neuronas: cuando arriba una entrada x, la neurona más apta para representarla gana la competición y se le permite que la aprenda aún mejor (hasta aquí no hay diferencia con la capa competitiva de la CPN).

Si existe un orden entre las neuronas, es decir, las neuronas están ubicadas en un reticulado discreto, el SOM, el algoritmo competitivo puede generalizarse: no sólo la neurona ganadora sino también sus *vecinas* en el reticulado aprenderán; así, las neuronas vecinas gradualmente se especializarán en representar entradas similares.

Las neuronas representan las entradas con vectores de referencia w_i, que corresponden a los pesos sinápticos. Cada vector de referencia w_i está asociado a una *unidad*. La unidad, indexada con c, cuyo vector de referencia es el más cercano a la entrada x es el ganador de la competición:

$$c = c(x) = argmin_i\{\|x - w_i\|\} \tag{5.61}$$

donde el operador $\|\cdot\|$ representa la distancia euclídea.

La unidad ganadora y sus vecinos se adaptan para representar la entrada aún mejor modificando sus vectores de referencia hacia la entrada corriente. La cantidad que las unidad aprenden será gobernado por una función de vecindad h, la cual es decreciente con respecto a la distancia de las unidades con respecto a la unidad ganadora y al tiempo.

Durante el proceso de aprendizaje en tiempo t, los vectores de referencia se modifican iterativamente de acuerdo a la siguiente regla de adaptación, donde $x(t)$ es la entrada en tiempo t y $c = c(x(t))$ es el índice de la unidad ganadora:

$$w_i(t+1) = w_i(t) + h_{ci}(t)[x(t) - w_i(t)] \tag{5.62}$$

En la práctica, la función de vecindad se elige para ser "ancha" al principio del proceso de aprendizaje y su ancho decrece lentamente durante el aprendizaje [Freeman93, Kaski97, Rao95, Wasserman89]. Esta función de vecindad puede ser una campana exponencial o una función de sombrero cuadrada [Freeman93, Rao95].

Es de notar que en caso de tener que agregar patrones nuevos a la red, ésta requiere que se la reentrene con los patrones nuevos sumados a los anteriores [Freeman93].

Aplicaciones del KSOM a IR

El KSOM admite una vasta aplicación a diversos problemas como: control de un brazo robótico [Freeman93, pp. 291–294] [Zeller97], a sociología económica [Kaski96], etc.

Sin embargo, en esta sección se explicará una aplicación del KSOM a la clasificación de documentos, ya que está fuertemente ligada al modelo de representación de documentos en este trabajo de grado.

En el artículo de Heikki Hyötiniemi [Hyötyniemi96], se describe el algoritmo CGHA. Éste consiste de la utilización de una red neuronal con arquitectura de KSOM para hacer extracción automática de características de documentos (tanto en inglés como en finlandés, aunque la técnica es de aplicación general). Esto es aplicable en el área de recuperación de información (capítulo 3).

Las ecuaciones que describen el entrenamiento de la red son las estándar (ecuaciones 5.61 y 5.62). Para el mismo se utilizó un conjunto de 40 documentos en finlandés e inglés consistente de artículos, letras de canciones y reportes técnicos.

Mientras que la capa autoorganizativa está compuesta por 16 nodos, la entrada está constituida por un vector normalizado de 30^3 números obtenidos a partir del documento de entrada como se describe a continuación. Si suponemos que el documento de entrada es una secuencia de caracteres de la forma:

$$d \equiv a_1 a_2 a_3 \dots a_n. \tag{5.63}$$

Entonces la entrada de la red neuronal será la cuenta de los trigramas del documento –una técnica que ya se mencionó en la la sección 3.9.2– normalizada para tener norma 1. Cada trigrama, digamos, '$a_1a_2a_3$', donde a_i son caracteres, se codifica como un único entero a tal que:

$$a \equiv \rho(a_1) \cdot 30^2 + \rho(a_2) \cdot 30 + \rho(a_3) \tag{5.64}$$

donde ρ mapea *<SPC>, 'a', ..., 'z', 'å', 'ä' y 'ö'* en números de 0 a 29.

El entrenamiento, como en todas las redes de este tipo, consiste de pasar varias veces por la red a los documentos del conjunto de entrenamiento.

Los resultados obtenidos por este autor indican que cada una de las neuronas de la capa de Kohonen clasifica una característica especial de los documentos como ser el idioma (inglés o finlandés), otras el área al que pertenece el documento (reportes astronómicos, canciones, etc.).

El autor demuestra, usando un documento fuera del conjunto de entrenamiento, que su método funciona. Además, apunta que su implementación es ineficiente pues sólo el 5% del vector de entrada es distinto de 0, lo cual concuerda aproximadamente con las mediciones propias (sección 4.6.5).

5.6.9 Teoría de la Resonancia Adaptativa (ART1)

La *Teoría de la Resonancia Adaptativa* (ART) [Freeman93, pp. 307–394] [Rao95, pp. 243–269] [Skapura96, pp. 46–52] [Wasserman89, pp. 127–150] sirve para la categorización de patrones usando el paradigma de aprendizaje competitivo. Esta red resuelve el dilema de la plasticidad-estabilidad. Éste supone que una red debe ser lo suficientemente plástica para aprender un patrón importante; pero al mismo tiempo debería permanecer estable cuando, en memoria de corto plazo, encuentra versiones distorcionadas del mismo patrón.

En la ART1, la clasificación de un patrón de entrada se intenta en relación a patrones almacenados, y si es infructuosa, se genera una nueva clase. El entrenamiento es no-supervisado. Hay dos versiones del entrenamiento: lento y rápido. Difieren en el tamaño del lapso que le toma a los pesos alcanzar su valor definitivo. El entrenamiento lento está gobernado por ecuaciones diferenciales mientras que el entrenamiento rápido está gobernado por ecuaciones algebráicas [Freeman93].

En el aprendizaje competitivo de la ART, se desarrolla un criterio para facilitar la ocurrencia del fenómeno *el ganador se lleva todo*. Un nodo simple con el valor más grande para el criterio de conjunto se declara ganador junto con su capa, y se dice que clasifica una clase de patrón. Si hay un empate entre varias neuronas ganadoras, entonces puede elegirse una de ellas como ganadora mediante una regla arbitraria, como el primero de ellos en orden secuencial.

La red neuronal desarrollada para esta teoría establece un sistema compuesto de dos subsistemas: Uno es el subsistema atencional (que contiene la unidad para control de ganancia). El otro es subsistema de orientación (que contiene la unidad de "reset"). Durante la operación de la red ART los patrones emergen en el subsistema atencional y son llamados trazas de la memoria de corto término (STM[9]). Las trazas de la memoria de largo término (LTM[10]) están en los pesos de conexión entre la capa de entrada y la de salida.

La red utiliza procesamiento con "feedback" entre sus dos capas hasta que ocurre la resonancia. La resonancia ocurre cuando la salida de la primer capa después del "feedback" de la segunda capa hace "match" con el patrón original usado como entrada para la primer capa en dicho ciclo de procesamiento. Un "match" de este tipo no tiene que ser perfecto. Lo que se requiere es que el grado de "match", medido acordemente, exceda un nivel predeterminado, llamado *parámetro de vigilancia*. Según [Rao95, p. 243], en referencia a la conducta de la ART1 frente a su entrada:

> *Como una fotografía hace "match" con el parecido de un individuo con un grado mayor cuando la granularidad es mayor, el "match" del patrón se hace mejor cuando el parámetro de vigilancia es cercano a 1.*

La red neuronal para el modelo ART1 consiste de los siguientes elementos [Rao95]:

1. una capa de neuronas F_1 (capa de entrada o comparación);
2. un nodo para cada capa como unidad de control de ganancia;
3. una capa de neuronas F_2 (capa de salida o reconocimiento);
4. un nodo como unidad de *reset*;
5. conexiones bottom-up de la capa F_1 a la capa F_2;
6. conexiones top-down de la capa F_2 a la capa F_1;
7. conexión inhibitoria de la capa F_1 al control de ganancia;
8. conexión exitatoria del control de ganancia a la capa F_1;
9. conexión inhibitoria de la capa F_1 al nodo de reset, y,

[9]Por "short-term memory".
[10]Por "Long-term memory".

10. conexión exhitatoria del nodo de reset a la capa F_2.

Un vector de entrada, cuando se aplica a la ART1, es primeramente comparado con los patrones existentes en el sistema. Si hay un parecido suficientemente cercano dento de una tolerancia especificada (como es indicada por el parámetro de tolerancia), entonces ese patrón almacenado se hace que se parezca aún más al patrón de entrada y la operación de clasificación es completa. Si el patrón de entrada no se parece a ninguno de los patrones en el sistema, entonces se crea una nueva categoría con el un nuevo patrón almacenado que se parece al patrón de entrada [Rao95].

Una característica especial de la ART1 es que la *regla de los dos tercios* es necesaria para determinar la actividad de las neuronas de la capa F_1. Hay tres fuentes de entrada para cada neurona de la capa F_1: la entrada externa, la salida del control de ganancia y las salidas de la capa F_2. Las neuronas de la capa F_1 no van a disparar a menos que dos de las tres respuestas estén activas. La unidad de control de ganancia y la regla de los dos tercios aseguran la respuesta correcta de la capa de neuronas de entrada. Una segunda característica se usa para determinar la actividad de la unidad de reset, la cual es activada cuando no hay "match" entre los patrones existentes durante la clasificación.

5.6.10 Teoría de la Resonancia Adaptativa Difusa

En la literatura de agentes, los mecanismos usados para clasificar texto encontrados son: el clasificador bayesiano [Balabanovic98, Billsus97, Maravall94, Pazzani97a, Pazzani97b], la red neuronal de *backpropagation* [Bigus98, Pazzani97a] y de Kohonen [Hyötyniemi96], razonamiento basado en memoria [Lashkari97]. También, en el área de IR se pueden hallar algoritmos especializados [Rasmussen92]. Además, los algoritmos de clasificación de imágenes [Maravall94] –ya que son generales– también se podrían aplicar al problema.

También, se cree que el clasificador bayesiano es una excelente opción (explorada profundamente en la literatura [Balabanovic98, Billsus97, Maravall94, Pazzani97a, Pazzani97b]); por otro lado, los enfoques de redes neuronales basados en "backpropagation" y KSOM si bien son capaces de aprender una función del conjunto de documentos en dos valores –como 'Bueno' y 'Malo'–, tienen, por otro lado, la desventaja de, al cambiar los requerimientos del usuario, requiriendo re-entrenamiento del nuevo conjunto de patrones.

Por lo anterior, la teoría de la resonancia adaptativa (ART) parece una opción viable para captar el concepto de *modelar los intereses de un usuario a través del tiempo*, ya que la solución del dilema de estabilidad-plasticidad permitiría a una red neuronal aprender patrones nuevos sin olvidar los ya aprendidos. Existe una especialización de la ART, el modelo de red neuronal auto-organizante llamado *ART difusa* (FART) [Lavoie99], que acepta entradas analógicas una a la vez y desarrolla una categorización; donde las entradas familiares activan su categoría, mientras que entradas no familiares disparan o bien el aprendizaje adaptativo de una categoría existente, o la creación de una nueva categoría. La conducta de FART tiene una explicación geométrica debido a una representación interna de los prototipos de las categorías como *hiperrectángulos* en el espacio de las entradas. El proceso de elección de las categorías siempre recupera el hiperrectángulo más pequeño conteniendo la entrada (o, si no existe, crea uno para aprenderla).

El Algoritmo de la FART

El modelo de red neuronal FART se describe en el artículo de Lavoie *et al.* [Lavoie99].

Sea a un vector M-dimensional $(a_1, a_2, \ldots, a_M)$, donde $0 \leq a_i \leq 1$. La entrada I obtenida mediante *codificación complementada* se obtiene como

$$\begin{aligned} I &= (a_1, a_2, \ldots, a_M, 1-a_1, 1-a_2, \ldots, 1-a_M) \\ &= (a, a^c). \end{aligned} \tag{5.65}$$

Asignar a cada *categoría* j un vector $w_j = (w_j 1, w_j 2, \ldots, w_j 2M)$ de *pesos* adaptativos. Cada categoría es inicialmente no comprometida[11], y sus pesos inicializados a uno. La funcionalidad de la FART puede describirse como un algoritmo de tres pasos.

1. *Elección de Categoría*: Con la presentación de una entrada I, se computa la *función de elección* T_j para cada categoría j

$$T_j = \frac{|I \wedge w_j|}{\alpha + |w_j|}. \tag{5.66}$$

   El operador norma $|\cdot|$ se define como $|x| \equiv \sum_{i=1}^{2M} |x_i|$, el símbolo $\wedge$ denota el operador de conjunción de lógica difusa[12] y α es un parámetro definido por el usuario, $\alpha > 0$. La categoría J para la cual la función de elección es maximal, es decir, $T_J = max\{T_j | j = 1, 2, 3, \cdots\}$, se elige para el test de vigilancia.

2. *Test de Vigilancia*: La similaridad entre w_J e I se compara contra un parámetro ρ llamado *vigilancia*, $0 \leq \rho \leq 1$, en el siguiente test:

$$\frac{|I \wedge w_J|}{|I|} \geq \rho. \tag{5.67}$$

   Si el test es pasado, entonces ocurre la resonancia (Paso 3) y el aprendizaje toma lugar. Si el test falla, entonces ocurre el *reset* por equivocación: el valor de T_J se setea a -1 durante la duración de la presentación de la entrada corriente, se elige otra categoría en el paso 1, y el test de vigilancia se repite. Las categorías se buscan, es decir, se eligen y se testean, hasta que una cumple con la ecuación 5.67. Esta categoría se dice *seleccionada* para I. Puede estar o no comprometida, en cuyo caso se vuelve comprometida durante la resonancia.

3. *Resonancia*: La resonancia hace referencia a la dinámica interna de la red mientras presta atención al vector $(I \wedge w_J)$. Durante la resonancia, el vector de pesos w_J de la categoría seleccionada se actualiza de acuerdo a la ecuación

$$w_J^{(new)} = \beta(I \wedge w_J^{(old)}) + (1-\beta) w_J^{(old)} \tag{5.68}$$

   donde β es un parámetro de tasa de aprendizaje, $0 < \beta \leq 1$. Lo que se aprende no es la entrada I misma, sino un vector de pesos atendido $(I \wedge w_J^{(old)})$; así, la ART difusa aprende prototipos en vez de ejemplares. El caso especial $\beta = 1$ se llama *aprendizaje rápido* y se asume en [Lavoie99].

[11] Uncommited.

[12] El *AND* difuso se define como $(u_1, \ldots, u_n) \wedge (v_1, \ldots, v_n) = (min(u_1, v_1), \ldots, min(u_n, v_n))$.

De acuerdo a [Lavoie99], cada vector de pesos w_j puede escribirse en la forma

$$w_j = (u_j, v_j^c) \tag{5.69}$$

donde u_j y v_j son vectores M-dimensionales correspondientes a las dos esquinas de un hiperrectángulo R_j.

El *tamaño* de R_j se define como

$$|R_j| \equiv \begin{cases} |v_j - u_j|, & \text{si j está comprometida} \\ -M, & \text{en caso contrario} \end{cases} \tag{5.70}$$

Este tamaño R_j está relacionado a la norma del vector de pesos $|w_j|$ por

$$|R_j| = M - |w_j| \tag{5.71}$$

Se debe notar [Lavoie99] que un vector pequeño implica una categoría grande y viceversa.

Con aprendizaje rápido, 5.68 reduce a

$$w_J^{(new)} = I \wedge w_J^{(old)} \tag{5.72}$$

y las esquinas de R_J se actualizan con

$$\begin{aligned} u_J^{(new)} &= a \wedge u_J^{(old)} \\ v_J^{(new)} &= a \vee v_J^{(old)} \end{aligned} \tag{5.73}$$

donde $\vee$ denota al operador de disyunción de lógica difusa[13]. Cuando una categoría comprometida se selecciona, R_J se expande al mínimo hiperrectángulo que conteniendo tanto a R_J y a la entrada a. Si a está dentro de R_J, entonces R_J no se modifica. Por lo tanto, cuando una categoría j está comprometida, el tamaño del hiperrectángulo sólo puede quedar igual o crecer. El tamaño máximo está determinado por el test de vigilancia, el cual puede escribirse en la forma [Lavoie99]

$$|R_J^{(new)}| \leq M(1-\rho). \tag{5.74}$$

Con ρ cercano a uno, sólo se permiten hiperrectángulos pequeños; mientras que con ρ cercano a cero, se permiten rectángulos pequeños y grandes.

Comentario sobre Inicialización de la FART

Como $w_j = (u_j, v_j^c)$ y, además, $w_j^0 = (1_M, 1_M)$, entonces u_{j_i} y v_{j_i}. Por lo tanto, inicialmente, $w_j(0)$ mide $-M$ [Lavoie99]. La demostración de esta afirmación es la siguiente:

$$\begin{aligned} |R_j(0)| &= |v_j(0) - u_j(0)| \\ &= |(v_{j_1}, \ldots, v_{j_M}) - (u_{j_1}, \ldots, u_{j_M})| \\ &= |(0, \ldots, 0) - (1, \ldots, 1)| \\ &= (-1, \ldots, -1) \\ &= \sum_{i=1}^{M} -1 \\ &= -M. \end{aligned} \tag{5.75}$$

[13]El *OR* difuso se define como $(u_1, \ldots, u_n) \vee (v_1, \ldots, v_n) = (max(u_1, v_1), \ldots, max(u_n, v_n))$.

Discusión sobre la Elección de Categoría

En esta subsección elaboraré la fórmula de elección la categoría en el paso 1 del algoritmo de la FART.

Como se recordará, la fórmula 5.66, es de la forma:

$$T_j = \frac{|I \wedge w_j|}{\alpha + |w_j|}. \tag{5.76}$$

En [Lavoie99], Lavoie *et. al.* hacen un desarrollo de esta fórmula para entender su significado, aquí se seguirán estos pasos pero se agregarán los pasos intermedios de su desarrollo.

$$\begin{aligned}
T_j &= \frac{|I \wedge w_j|}{\alpha + |w_j|} && (5.77)\\
&= \frac{|I \wedge w_j| + |w_j| - |w_j|}{\alpha + |w_j|} && (5.78)\\
&= \frac{|w_j| - (|w_j| - |I \wedge w_j|)}{\alpha + |w_j|} && (5.79)\\
&= \frac{M - |R_j| - (|w_j| - |I \wedge w_j|)}{\alpha + |w_j|} && (5.80)\\
&= \frac{M - |R_j| - (|w_j| - |I \wedge w_j|)}{\alpha + M - |R_j|} && (5.81)\\
&= \frac{M - |R_j| - |w_j - (I \wedge w_j)|}{\alpha + M - |R_j|} && (5.82)\\
&= \frac{M - |R_j| - d_j}{\alpha + M - |R_j|} && (5.83)
\end{aligned}$$

donde $d_j \equiv d(w_j, I \wedge w_j)$.

El paso de (5.77) a (5.78) se hace sumando y restando $|w_j|$ en el numerador. El paso (5.79) a (5.80) se hace usando la ecuación 5.71, como también el paso de (5.80) a (5.81). El paso de (5.81) a (5.82) se hace usando la propiedad $|u + v| = |u| + |v|$ [Rao95]. La expresión d_j denota la distancia difusa de Hamming (de Kosko), donde $d(x, y) \equiv \sum_{i=1}^{2M} |x_i - y_i|$ [Lavoie99].

De acuerdo a [Lavoie99], si el vector de entrada a está fuera del hiperrectángulo R_j, entonces d_j es equivalente a la distancia *city-block* entre a y la arista o esquina más cercana de R_j; si a está dentro de R_j, entonces d_j es igual a cero; y si j no está asignada, entonces d_j toma el valor M.

También según [Lavoie99], la distancia entre R_j y a juega un rol muy importante en la dinámica de la FART. Supongamos que $d_j = 0$, entonces la categoría j se dice que es la *elección de subconjunto* para I. Cuando se hace una elección de subconjunto, su vector de pesos no es modificado durante la resonancia porque el prototipo no es otra cosa que vector mismo (es decir, $I \wedge w_j = w_j$); en este caso, se conserva el conocimiento previo. Esto concuerda con los experimentos realizados personalmente con la implementación propia de la FART en el plano difuso. Cuando $d_j > 0$ y se elige la categoría j, los pesos cambiarán necesariamente durante la resonancia porque el prototipo aprendido $I \wedge w_j$ es distinto de w_j. Según [Lavoie99], el valor de la función de elección T_j crece haciendo que

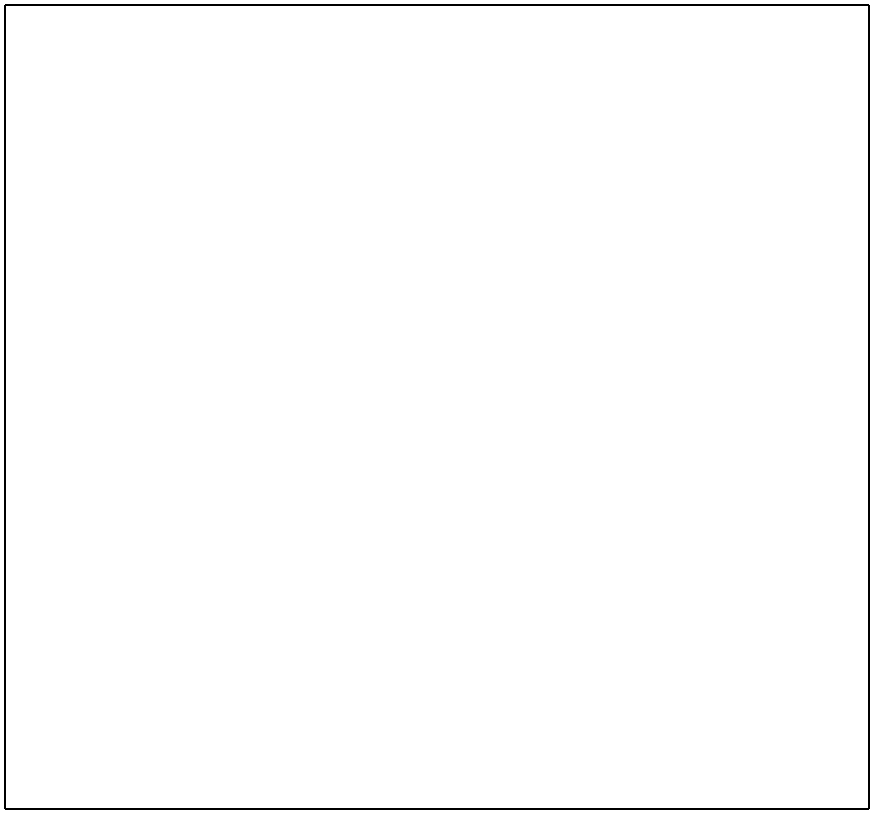

Figura 5.1: El efecto de la elección adecuada del parámetro de tolerancia en la red FART.

esta categoría sea elegida nuevamente al presentarse el mismo patrón a. Cuando se crea una categoría nueva, el criterio se satisface trivialmente.

El parámetro α sirve para determinar si una categoría puede elegirse cuando aparece un patrón I; en particular, j puede evaluarse sólo si satisface que [Lavoie99]:

$$T_j \geq \frac{M}{\alpha + 2M} \tag{5.84}$$

También, según [Lavoie99] el orden en que las categorías son exploradas es independiente del nivel de vigilancia ρ; las categorías de igual tamaño se buscan en orden creciente de su distancia a a y se buscan en orden creciente de tamaño.

Con respecto al nivel de vigilancia ρ, cuando la red chequea contra el parámetro de tolerancia para decidir si va a o no a crear una nueva clase, pueden ocurrir una de dos cosas, como se puede apreciar en la figura 5.6.10.

En el caso de la representación de características en el agente *Querando!* se puede observar. El tamaño del alfabeto Σ es 27 ($|\Sigma| = 27$). Por lo tanto, el tamaño de los vectores de "input" es $M = |\Sigma|^3 = 19683$.

Así, $2M = 39366$. De esta manera, $|w_j(0)| = 2M$. Luego, podemos plantear la siguiente conjetura:

$((w_j = 1_{2M}) \wedge (I \approx (0_M, 1_M)))$
$\Rightarrow ((I \wedge w_j) \approx (0_M, 1_M))$
$\Rightarrow (|I \wedge w_j| \approx M) \wedge |I| \approx M$
$\Rightarrow T_j \approx 1$

Esto concuerda con las observaciones (ver sección de *Mediciones*).

Comparación con el Método de la Pasada Simple

Se puede trazar una comparación entre el algorito de la FART y el método de la pasada simple.

El método de la pasada simple es similar al algoritmo de la FART. Ambos algoritmos requieren que los patrones a clasificar se les presenten una única vez. La clasificación resultante en ambos casos dependerá de dicho orden.

Sin embargo, una diferencia entre los dos métodos es que en la FART no hay cálculo de un nuevo centroide pues los clusters crecen en forma rectangular; mientras que en el algortimo de la pasada simple, el centroide de cada cluster se recalcula en cada presentación de un patrón de entrada.

La elección de la creación de un nuevo cluster dependerá en ambos métodos de la distancia del nuevo patrón al centroide y/o cluster y del tamaño de este último.

5.7 Discusión

En esta sección se comparan los distintos algoritmos de clasificación de patrones vistos en este capítulo desde el punto de vista de su posible aplicación a la implementación del agente de filtrado *Querando!*.

Hasta este punto, hemos descripto varios algoritmos aplicables a la clasificación y filtrado de documentos. De hecho, varios de ellos ya han sido aplicados con éxito a esta tarea. Dichos algoritmos son la base de la implementación de agentes que ya se describieron (por ejemplo, ver sección 5.6.8) y otros que se describen más adelante (secciones 9.3.8, 9.3.1, 9.3.2 y 9.3.13).

Primero, el clasificador bayesiano, si bien ha sido usado con éxito, requiere conocer la distribución estadística de las características dentro de la población de patrones; esto, en el caso del problema tratado en este trabajo, equivale a conocer la distribución de los términos en las páginas HTML de la web, lo cual requiere la construcción de un archivo invertido de los documentos de la web y equivale a replicar a *Altavista* o a *Yahoo*. Claramente, esta opción es inviable por la magnitud de la empresa.

Segundo, los algoritmos de clustering de las secciones 5.4.1 y 5.4, si bien funcionan bastante bien, adolecen de dos defectos: algunos requieren conocer *a priori* la cantidad de clases y todos requieren que se conozca íntegramente la población de vectores a clasificar. Estos dos hechos los hacen inviables en *Querando!* por los siguientes motivos:

1. en el caso de un usuario que dinámicamente filtra documentos es imposible conocer previamente cuántas clases de documentos se van a necesitar definir;

2. por la naturaleza dinámica de la web y de la exploración de los documentos, la población de documentos (patrones) va a crecer dinámicamente.

Tercero, el método del razonamiento basado en memoria 5.5 ha probado ser efectivo en un ambiente de filtrado de correo electrónico (sección 9.3.1). En principio parece adecuado para la labor a mano. Sin embargo, su inicialización, como se apunta en [Lashkari97], depende de otros agentes con los que se pueda intercambiar información.

Cuarto, los algoritmos de redes neuronales como el Perceptrón, la BPN, la CPN y el KSOM requieren conocer todos los patrones para lograr un entrenamiento exitoso; en particular la BPN (secciones 9.3.2 y 9.3.8) y el KSOM (en el caso de WEBSOM) han sido

usados con éxito. Además, cuando cambian las condiciones (se agregan nuevos patrones o cambia la clasificación de los mismos[14]), estas arquitecturas requieren reentrenamiento con los patrones anteriores sumados a los nuevos. Claramente, estas opciones también son inviables.

Quinto, la teoría de la resonancia adaptativa parece una opción viable para captar el concepto de modelar los intereses de un usuario a través del tiempo. La FART, al solucionar el dilema de la estabilidad-plasticidad, permite a una red neuronal aprender patrones nuevos sin olvidar los ya aprendidos. Además, esta solución no require entrenamiento previo, es no supervisada y se adapta dinámicamente a los patrones nuevos.

5.8 Resumen

En este capítulo se expusieron los diversos métodos de clasificación y clustering hallados en la literatura. Al final se hace una comparación entre ellos orientada a su posible utilización en la implementación del agente de filtrado *Querando!*, concluyendo que la arquitectura de la teoría de la resonancia adaptativa difusa es la más adecuada.

[14] De Bueno a Malo y viceversa.

Capítulo 6

Mediciones con Clustering

En este capítulo se expondrán y analizarán las mediciones realizadas para determinar si el clustering de documentos es apropiado para representar la estructura del conjunto de documentos considerados.

El usuario tiende a agrupar los documentos con un criterio subjetivo que podría definirse con expresiones como "este documento es *parecido* a este otro" o "tal documento habla del *mismo* tema que este otro documento". Por otro lado, desde el punto de vista algorítmico, cuando planteamos un método heurístico para clasificar documentos, definimos una representación para los mismos y una métrica para determinar el grado de similitud entre ellos.

El objetivo de este capítulo es entonces determinar si el criterio subjetivo de agrupación del usuario coincide (al menos en un porcentaje razonable) con el del programa que realiza esta agrupación en forma automática.

En el capítulo 5 se expusieron y analizaron los métodos de clasificación de patrones hallados en la literatura. Sin embargo, los algoritmos de clustering usados en este capítulo son cuatro: el algoritmo de las k-medias, el modelo de red neuronal de contrapropagación, la red de Kohonen y la red de la teoría de la resonancia adaptativa difusa. Se muestran las mediciones realizadas con estos algoritmos y se las analiza.

6.1 Representación de los Documentos

El problema de la representación de los documentos ya fue abordado en este trabajo de grado en la sección 4.6. También, se analizaron varias métricas de similitud entre documentos en ese mismo capítulo y se expusieron las encontradas en la literatura en la sección 3.2.2.

La representación de documentos usada en este capítulo está basada en la ya mencionada representación de los contadores de apariciones de trigramas pero esta vez escalados con una función sigmoidea (veáse capítulo 4). La función sigmoidea tiene la propiedad de escalar todos los valores mayores a cero en el intervalo real $[0, 1]$:

$$sgm(x) = \frac{1}{1 + e^{-\alpha \cdot x}} \tag{6.1}$$

Otro punto a considerar es que se analizaron el alfabeto conteniendo el blanco y sin contenerlo. La representación conteniendo el blanco retiene información de qué palabras

aparecen juntas en los documentos mientras que la representación que no lo contiene no tiene esta información.

6.2 Implementaciones de Algoritmos

Aquí se discuten las implementaciones que se realizaron de los algoritmos de clustering: K-medias, red neuronal de contrapropagación, red neuronal de Kohonen y red de la teoría de la resonancia adaptativa difusa. Dichos algoritmos ya fueron discutidos en las secciones 5.3.3, 5.6.7, 5.6.8 y 5.6.10, respectivamente. Todas las implementaciones se realizaron en el lenguaje C++ [Brokken95, Jennings99, Kernigham85, Kruglinski97, Rao95].

6.2.1 K-Medias

El algoritmo de las K-medias (sección 5.3.3) realiza agrupación de un conjunto de puntos en forma no supervisada; sin embargo, requiere que se le indique la cantidad exacta de clases en las que se quieren particionar los patrones de entrada.

Se realizaron dos implementaciones de este algoritmo:

1. Una para probar al mismo en condiciones controladas. Ésta se hizo en el plano cartesiano. Al programa se le ingresan un conjunto de puntos por medio de varios clicks del mouse en un panel, se le indican la cantidad de clases esperadas y el mismo encuentra una partición de los puntos (indicando cada clase con un color diferente) y, además, muestra dónde está la ubicación del centroide de cada clase. Este programa fue implementado en C++.

2. La otra implementación del algoritmo de las K-medias se realizó para clasificar un conjunto de documentos. El programa requiere el número de clases esperadas y un archivo de texto que contiene en cada línea del mismo el nombre de uno de los archivos a clasificar. Los documentos entonces son recuperados y convertidos a su representación respectiva, en la que se puede especificar si aplicar o no stop list y stemming, eliminar los marcadores en documentos HTML y si se va a usar el carácter blanco como parte del alfabeto de los trigramas. La interacción de este programa se detalla en la sección 6.3.1.

Implementación 2D del K-Medias

El algoritmo de las K-medias es sólo de utilidad teórica ya que, para realizar una correcta clasificación de una muestra de patrones, requiere como condición ineludible que se conozca *a priori* el número exacto de clases en las que se desea particionar dicha muestra, lo que es difícil de llevar a cabo en la práctica para problemas de grandes dimensiones.

Sin embargo, se hicieron mediciones con este algoritmo, ya que su simpleza es de gran utilidad para determinar si la representación de documentos es apropiada para realizar el agrupamiento de documentos en clases conceptuales.

La interfaz del programa que implementa el algoritmo sólo requiere que el usuario marque con el mouse los puntos a clasificar, indique la cantidad de clases deseadas y dispare la clasificación (figura 6.1 izquierda). Una vez terminada la clasificación, el programa pintará a cada punto con el color de la clase a la que pertenece y dibujará los centroides de cada clase (figura 6.1 derecha).

Figura 6.1: Clasificador K-Medias 2D con los puntos ingresados (izq.) y clasificados (der.).

Discusión

Si bien la fortaleza del K-medias está en su simplicidad, ésta también es una debilidad. Cuando el número de clases es conocido, el algoritmo generalmente funciona muy bien (más sobre esto después). Determinar el número de clases de un conjunto de puntos del plano es una tarea sencilla para el ojo humano. Sin embargo, en el problema de la clasificación de documentos, la dimensión del problema es bastante mayor. Esto se debe a que la dimensión de los vectores de características en la representación planteada tienen 27^3 (19683) entradas. Con estas dimensiones, estimar la cantidad real de clases parece un problema muy difícil.

Por las razones expuestas, se investigó cómo se comportaba el algoritmo de las K-medias cuando el número de clases especificado era incorrecto.

En la figura 6.1 de la derecha aparece una clasificación correcta de los patrones ingresados. Esto se logra pues el número de clases (5) ingresado por el usuario es el correcto. Cuando el número de clases es incorrecto, el K-medias tiende a hacer una de dos cosas:

1. Por un lado, cuando el número de clases propuesto es inferior al real, el algoritmo tiende a juntar dos clases en una. Dos ejemplos de esta situación con dos particiones diferentes se aprecian en la figura 6.2. Hay tres clases pero el usuario especifica que son dos, entonces dos clases se unen en una sola y el centroide queda a mitad de camino entre ambas.

2. Por otro lado, cuando el número de clases propuesto es superior al real, el algoritmo tiende a particionar una clase en dos clases ficticias. Esta situación se aprecia en la figura 6.3. Allí, hay claramente dos clusters pero se especificaron tres y el cluster de la derecha es dividido en dos por el algoritmo.

La inicialización de los centroides se hace eligiendo aleatoriamente k vectores diferentes de entre los vectores a clasificar. Un problema detectado en el K-medias fue que a veces realiza mal la clasificación. El *a veces* surge por la naturaleza aleatoria de la inicialización de los centroides de las clases. Dos ejemplos de esta situación se ven en la figura 6.4, donde se aprecia las clases mal particionadas.

6.2.2 Contrapropagación

El algoritmo de clasificación usando el modelo de red neuronal de contrapropagación se implementó para probar métodos de clustering de vectores clásicos en dicha área.

Figura 6.2: Clasificador K-Medias 2D con clases de menos.

Figura 6.3: Clasificador K-Medias 2D con clases de más.

Figura 6.4: Clasificador K-Medias 2D con clasificación errónea.

La implementación sólo realiza el aprendizaje de la capa competitiva o *fan-in* (veáse sección 5.6.7) con el fin de hacer el clustering de los documentos.

La implementación requiere que el usuario ingrese el nombre de un documento donde en cada línea se especifica el nombre de un archivo (de texto o HTML) a clasificar. Otro de los parámetros requeridos es la tasa de aprendizaje α. Además, el programa permite especificar la forma en que se va a generar los vectores de características de los documentos; las opciones que se pueden elegir son: usar o no el blanco, usar o no lista de stop words y stemming y eliminar o no los marcadores de los documentos HTML.

6.2.3 Mapa de Kohonen

Las mediciones usando clustering con la red neuronal autoorganizativa de Kohonen (sección 5.6.8 se hacen con el mismo programa que las de la red de contrapropagación. Los parámetros son los mismos que en dicho caso. En esta red, la ventana de actualización de pesos de conexiones de neuronas vecinas se achica progresivamente hasta que queda de tamaño 1.

En ambos casos, se tiene una opción para inicializar los centroides iniciales de los k clusters con los primeros k documentos del archivo de documentos a filtrar. Esto se hace para acortar el tiempo de entrenamiento.

6.2.4 Red FART

También se implementó la red neuronal basada en el modelo de la Teoría de la Resonancia Adaptativa Difusa (sección 5.6.10). Se realizaron dos implementaciones: una en 2D y otra para los documentos propiamente dichos.

Implementación 2D de la FART

Como en el caso del algoritmo de las K-medias, se hizo también una implementación 2D para analizar el problema del clustering[1] con este algoritmo en un ambiente controlado.

La red FART, como se explicó en la sección 5.6.10, es capaz de ir agrupando los puntos de entrada a medida que éstos son presentados a la red. Los puntos cercanos entre sí son agrupados en una misma clase. Los puntos alejados se ponen en clases nuevas a menos que el tamaño de la clase supere un umbral. La entrada sólo requiere presentarse una única vez al algoritmo. Por ello, este modelo de red parece muy útil para resolver el problema de clasificar y filtrar un flujo continuo de documentos (veáse sección 3.12).

La red FART debe su comportamiento a los valores que toman los siguientes parámetros: el parámetros para *elección de categoría* α, el parámetro de *vigilancia* ρ y el parámetro de *tasa de aprendizaje* β. En los siguientes incisos se analizará la incidencia en el entrenamiento de las variaciones en los valores de estos parámetros. Debido a las interrelaciones entre ellos, el orden en que se presentan puede parecer arbitrario pues no hay manera de explicarlos en un orden secuencial tal que los posteriores se basen en los anteriores.

1. *Parámetro ρ:*

[1] En esta sección los términos clases, categorías y clusters se usan indistintamente y todos hacen referencia al mismo concepto.

Figura 6.5: Parámetro ρ en la FART.

El valor del parámetro de *vigilancia* ρ está directamente relacionado con el tamaño de las clases resultantes. Un valor de ρ cercano a 0 da clases grandes mientras que un valor de ρ cercano a 1 da clases pequeñas.

En la figura 6.5 se muestran tres disposiciones de clases dependiendo del parámetro ρ. Los parámetros usados son los siguientes (de izquierda a derecha):

(a) $\alpha = 0.0$, $\beta = 1.0$ y $\rho = 0.1$;

(b) $\alpha = 0.0$, $\beta = 1.0$ y $\rho = 0.6$;

(c) $\alpha = 0.0$, $\beta = 1.0$ y $\rho = 0.9$.

En la misma figura, se observa que la superficie de las clases disminuye a medida que aumenta ρ en concordancia con lo afirmado por [Lavoie99].

2. *Parámetro* α

   La relación del parámetro de *elección de categoría* α ya se explicó en la sección 5.6.10. El parámetro α provee un término de escalado para la entrada, a mayor α se tiene un menor valor de T_j y viceversa. Por lo tanto, el valor de α debe elegirse conjuntamente con el de ρ.

   Un parámetro α grande promueve la proliferación de clases al principio del entrenamiento. En general cada punto nuevo genera una nueva clase (a menos que los puntos estén muy cerca). Después, a medida que transcurre el entrenamiento, las clases comenzarán a crecer y habrá superposición de las mismas.

   Por el contrario, un α pequeño no promueve la proliferación de clases al principio del entrenamiento. Para mismos $\beta = 1$ y ρ, los puntos se agruparán en un pocas clases.

   Otro efecto de este parámetro es que si es mayor a 0, aún cuando un nuevo patrón esté contenido en una clase, puede hacer crecer una clase vecina en la que el patrón *no* estaba contenido. En la figura 6.6 se puede observar que el punto $(0.55, 0.58)$[2], a pesar de estar contenido sólo en la clase de la izquierda, hace que la competencia sea ganada por la clase de la derecha, haciéndola crecer. Cuando $\alpha = 0$, esto no ocurre.

[2]Este punto está casi en el medio del espacio difuso $[0, 1] \times [0, 1]$. Además, este punto se encuentra en la intersección de las dos regiones.

Figura 6.6: FART con parámetro $\alpha = 6$.

Figura 6.7: Parámetro $\beta = 1$ en la FART.

3. *Parámetro β:*

   El valor de β debe ser obligatoriamente mayor a 0 pues sino la red no aprende ya que, por la ecuación 5.68, siempre $w_J^{(new)} = w_J^{(old)}$.

   Hacer $\beta = 1$ se llama *aprendizaje rápido* [Lavoie99]. Su efecto consiste en que cada vez que aparece un nuevo patrón, la clase ganadora se agranda para contenerlo (en caso de que el patrón esté fuera de la clase) o permanece estable (en caso de que el patrón esté contenido en ella). En la figura 6.7, se observa el efecto de clasificar los puntos $(0.16, 0.11)$ (izq.), $(0.28, 0.18)$ y $(0.22, 0.13)$ (centro) y $(0.24, 0.40)$ (derecha) con $\alpha = 0$, $\beta = 1$ y $\rho = 0.7$. El primer punto crea un cluster de superficie nula, el segundo punto agranda el cluster tal que ambos puntos constituyen las esquinas. El tercer punto cae dentro del cluster y no lo agranda. El cuarto punto cae debajo y el cluster crece para incluirlo.

   Hacer $\beta = 0.3$ tiene un efecto curioso. El resultado de agregar patrones progresivamente se puede ver en la figura 6.8; la secuencia se lee de izquierda a derecha y de arriba hacia abajo. Los puntos agregados son los siguientes: $(0.45, 0.44)$, $(0.51, 0.59)$, $(0.53, 0.53)$, $(0.42, 0.47)$, $(0.45, 0.52)$, $(0.32, 0.51)$, $(0.62, 0.96)$, $(0.27, 0.45)$ y $(0.50, 0.20)$. En este caso, también $\alpha = 0$ y $\rho = 0.7$. Se puede observar, que a diferencia del caso anterior, la clase empieza siendo grande, luego decrece y por último empieza a crecer nuevamente. También, se observa que hay puntos que quedan fuera de la clase. Por lo tanto, de esta manera el método tiene una fuerte inercia.

4. *Superposición de Clases:*

   Uno de los defectos de la FART es la superposición de clases. La superposición de clases ocurre cuando la intersección entre las regiones que simbolizan a las mismas tienen intersección no nula.

Figura 6.8: Parámetro $\beta = 0.3$ en la FART.

Figura 6.9: Formulario *TestFART.htm*.

Cuando hay superposición de dos clases i y j y se agrega un patrón en la intersección de ellas, ambas T_i y T_j valen 1. Por la forma secuencial del algoritmo, en realidad se elegirá a la clase más vieja.

FART para Documentos

El algoritmo de la FART para hacer las mediciones iniciales con documentos está basado en un CGI, el usuario llena los datos de la simulación en un formulario HTML (*/querando/testFART.htm*) y éstos se envían a la aplicación */querando/cgi/TestFART.exe*. Esta interfaz permite especificar los parámetros de la red neuronal: α, β, ρ y el número máximo de clases que tendrá la red (esta restricción no está en el agente). Además, se ingresa un lista de URLs correpondientes a los documentos a clasificar. Dicha interfaz se puede apreciar en la figura 6.9.

El resultado de la simulación se presenta también como una página HTML y se ve en donde se muestra los valores de la activación de las distintas unidades de la red neuronal (veáse la figura 6.10).

En la página HTML devuelta por este programa se puede seguir de cerca el comportamiento de la red neuronal. Por ejemplo, al considerar la clasificación de un documento vemos lo siguiente (figura 6.11 [3]):

1. Nombre del documento a clasificar: *http://localhost/tesis/testpool/dosclases/1/vcg03.htm*.

2. Norma del vector de entrada o representación del documento $|a|$ y la norma del vector complementado $|I|$:

3. Para cada categoría, los valores de la función de elección de categoría T_j, los valores del numerador y denominador de esta expresión $|I \wedge w_j|$ y $|w_j|$, el tamaño de la región definida por la categoría $|R_j|$.

[3]Lo que se muestra es la versión editada de la salida real. La editada real está en formato HTML mientras que la editada está en formato LaTeX.

Figura 6.10: Resultado de *TestFART.exe.*

4. El número de las categorías a medida que son elegidas hasta que una de ellas resuena, también el valor del test de vigilancia para cada categoría evaluada.

5. El número de la categoría que resonó.

6.3 Mediciones con Dos Idiomas

A continuación se muestran y analizan las mediciones realizadas para determinar si la representación sirve para distinguir documentos en dos idiomas distintos: castellano e inglés. Esta hipótesis fue sugerida en la literatura por [Hyötyniemi96] quien usó una red de KSOM para esta tarea pero aplicada al finlandés y al inglés.

6.3.1 Clustering con el Algoritmo de las K-Medias

Se hicieron pruebas con el alfabeto incluyendo sólo las letras del alfabeto occidental, y con y sin el caracter blanco. Como se quería determinar si la representación servía para diferenciar entre documentos escritos en uno u otro idioma, no se aplicaron stop lists ni stemming a las palabras provenientes de los documentos clasificados. Como los documentos estaban en formato de texto plano, no se requirió la eliminación de marcadores HTML.

También, las mediciones con el algoritmo de las K-Medias se hicieron con el programa *TestKMediasDocs.exe*, el cual fue implementado *ad-hoc*.

Los archivos usados en esta medición están en el directorio *Tesis/TestPool/InglesEspaniol*. Los documentos en idioma inglés de esta muestra fueron archivos *readme.txt* de diversas aplicaciones comerciales. Por otro lado, los archivos de la muestra que pertenecían al idioma español correspondieron a apuntes de la cátedra de *Seminario B* de esta facultad y a una resolución del *Concejo Académico* de esta facultad.

Aquí se detalla la interacción con el usuario para ver los parámetros de la medición.

Clasificando doc: *http://localhost/tesis/testpool/dosclases/1/vcg03.htm*
Downloading... Listo.
Código devuelto por el servidor: 200.
Decorando página... Listo.
Generando vector de características... Listo.
Propagando vector en la FART...
$|a|$: 2226.634521
$|I|$: 19682.970703
$|I \wedge w_j|$: 17253.587891, $|w_j|$: 18533.474609.
$|I \wedge w_j|$: 18391.023438, $|w_j|$: 19682.980469.
$|I \wedge w_j|$: 19682.970703, $|w_j|$: 39366.000000.
$|I \wedge w_j|$: 19682.970703, $|w_j|$: 39366.000000.
$|I \wedge w_j|$: 19682.970703, $|w_j|$: 39366.000000.
$|I \wedge w_j|$: 19682.970703, $|w_j|$: 39366.000000.
$|I \wedge w_j|$: 19682.970703, $|w_j|$: 39366.000000.
$|I \wedge w_j|$: 19682.970703, $|w_j|$: 39366.000000.
$|I \wedge w_j|$: 19682.970703, $|w_j|$: 39366.000000.
$|I \wedge w_j|$: 19682.970703, $|w_j|$: 39366.000000.
T_0: 0.883283. $|R_0|$ = M - w_0 = 1149.525391.
T_1: 0.889186. $|R_1|$ = M - w_1 = 0.019531.
T_2: 0.487613. $|R_2|$ = M - w_2 = -19683.000000.
T_3: 0.487613. $|R_3|$ = M - w_3 = -19683.000000.
T_4: 0.487613. $|R_4|$ = M - w_4 = -19683.000000.
T_5: 0.487613. $|R_5|$ = M - w_5 = -19683.000000.
T_6: 0.487613. $|R_6|$ = M - w_6 = -19683.000000.
T_7: 0.487613. $|R_7|$ = M - w_7 = -19683.000000.
T_8: 0.487613. $|R_8|$ = M - w_8 = -19683.000000.
T_9: 0.487613. $|R_9|$ = M - w_9 = -19683.000000.
Category Choice: 1
Cociente: 0.934362 Rho: 0.900000.
Resonancia? (s/n): Sí
El cluster ganador es: 1.

Figura 6.11: Una iteración de TestFART.exe.

```
Ingrese la cantidad de clases esperadas: 2
Uso blanco? (s/n): s
Uso stop list y stemming (ingles)? (s/n): n
Tiro tags HTML? (s/n): n
Archivo de docs: c:/Inetpub/wwwroot/Tesis/TestPool/InglesEspaniol/ingesp.dir
Archivo de salida: c:/Inetpub/wwwroot/Tesis/TestPool/InglesEspaniol/res1.txt
```

1. El resultado de una ejecución de la medición fue el siguiente:

```
Documento             Clase
------------------------------
ingles2.txt           1
ingles3.txt           1
ingles4.txt           1
ingles5.txt           0
ingles6.txt           0
ingles7.txt           1
ingles1.txt           0
ingles8.txt           1
espa1.txt             0
espa2.txt             0
espa3.txt             0
```

2. Otra ejecución del algoritmo con la misma muestra dio:

```
Documento             Clase
------------------------------
ingles2.txt           0
ingles3.txt           0
ingles4.txt           0
ingles5.txt           1
ingles6.txt           1
ingles7.txt           0
ingles1.txt           1
ingles8.txt           0
espa1.txt             1
espa2.txt             1
espa3.txt             1
```

3. Otra ejecución más del algoritmo con la misma muestra dio:

```
Documento             Clase
------------------------------
ingles2.txt           1
ingles3.txt           1
ingles4.txt           1
ingles5.txt           1
ingles6.txt           1
ingles7.txt           1
```

```
ingles1.txt          1
ingles8.txt          1
espa1.txt            0
espa2.txt            1
espa3.txt            1
```

Los resultados obtenidos indican que la representación no funciona para distinguir los documentos de los dos idiomas. También, se evidencia la sensibilidad del K-medias a los patrones que se utilizó para definir los centroides iniciales para los clusters. Otra explicación de la dispar clasificación puede ser que la cantidad de clases no sea 2, como se supuso a priori. Los siguientes experimentos fueron con número de clases 3, 4, 5, 6 y 7, respectivamente.

4. La siguiente fue la salida para cantidad de clusters igual a 3, usando blancos, sin stop list ni stemming:

```
Documento            Clase
-----------------------------
ingles2.txt          2
ingles3.txt          2
ingles4.txt          2
ingles5.txt          2
ingles6.txt          0
ingles7.txt          2
ingles1.txt          2
ingles8.txt          2
espa1.txt            1
espa2.txt            1
espa3.txt            0
```

Aquí, se ve que los dos documentos de la clase 0 están escritos uno en castellano y el otro en inglés, los demás documentos están *a priori* correctamente separados.

5. Otra corrida para cantidad de clusters igual a 3 en las mismas condiciones dio:

```
Documento            Clase
-----------------------------
ingles2.txt          1
ingles3.txt          1
ingles4.txt          1
ingles5.txt          2
ingles6.txt          0
ingles7.txt          1
ingles1.txt          2
ingles8.txt          2
espa1.txt            0
espa2.txt            0
espa3.txt            0
```

La diferencia entre las corridas indica que la cantidad de clases siguió siendo incorrecta, ya que en la clase 0 hubo tres documentos en castellano pero uno en inglés.

6. Luego, se hizo una prueba suponiendo que hay 4 clases, la cual dio:

```
Documento              Clase
-----------------------------
ingles2.txt            3
ingles3.txt            3
ingles4.txt            3
ingles5.txt            0
ingles6.txt            0
ingles7.txt            1
ingles1.txt            2
ingles8.txt            1
espa1.txt              0
espa2.txt              0
espa3.txt              0
```

Apareció el mismo problema, en la clase 0 se mezclaron documentos en castellano con un documento en inglés.

7. Otra corrida con cuatro clases dio:

```
Documento              Clase
-----------------------------
ingles2.txt            1
ingles3.txt            1
ingles4.txt            1
ingles5.txt            3
ingles6.txt            2
ingles7.txt            1
ingles1.txt            3
ingles8.txt            3
espa1.txt              0
espa2.txt              0
espa3.txt              2
```

En esta corrida, la clase número 2 contuvo documentos en castellano e inglés.

En general, los resultados sugieren que la representación no sirve para distinguir entre el castellano y el inglés.

6.3.2 Clustering con el Algoritmo de la FART

Aquí, se enumeran los mismos experimentos que en la sección anterior, pero esta vez usando la arquitectura de red neuronal de la resonancia adaptativa difusa (ver 5.6.10) donde se usaron los mismos documentos que con el algoritmo de K-medias.

En esta medición se repitió el análisis de los documentos en el directorio */Tesis/TestPool/InglesEspaniol*.

En todos los casos, los archivos de las mediciones así como de la evolución del algoritmo se encuentran grabados en un subdirectorio llamado emphfart en los directorios correspondientes. Aquí, sólo se reproducen los resultados fundamentales por una cuestión de espacio.

Las mediciones fueron las siguientes:

1. En esta medición se usaron los siguientes parámetros. Se usó eliminación de tags HTML, se usaron stop lists y stemming[4], como alfabeto se incluyeron a las letras y al blanco; mientras que los parámetros de la red neuronal fueron los siguientes: $\alpha = 1000$, $\beta = 1.0$, $\rho = 0.9$ y cantidad de clusters = 10.

   Los archivos se procesaron en el siguiente orden: *ingles2.txt, ingles3.txt, ingles4.txt, ingles5.txt, ingles6.txt, ingles7.txt, ingles1.txt, ingles8.txt, espa1.txt, espa2.txt y espa3.txt*.

   Los resultados fueron los siguientes:

   ```
   Documento         Clase
   ---------------------------
   ingles2.txt       0
   ingles3.txt       0
   ingles4.txt       0
   ingles5.txt       0
   ingles6.txt       0
   ingles7.txt       0
   ingles1.txt       0
   ingles8.txt       0
   espa1.txt         1
   espa2.txt         1
   espa3.txt         1
   ```

   En esta medición, la red neuronal dividió a los documentos adecuadamente. Esta medición se puede ver completa en el archivo *Tesis/TestPool/InglesEspaniol/fart/TestFART1.htm*

2. Presentación de los documentos en otro orden usando los mismos parámetros que en la medición anterior.

   Los resultados fueron los siguientes:

   ```
   Documento         Clase
   ---------------------------
   espa1.txt         0
   ingles2.txt       0
   ingles3.txt       0
   ingles4.txt       0
   ingles6.txt       0
   ```

[4] A pesar de que la intuición sugiere lo contrario, ya que esta técnica está orientada sólo al inglés.

```
ingles7.txt       0
ingles1.txt       1
ingles8.txt       0
espa2.txt         1
ingles5.txt       1
espa3.txt         1
```

Aquí, vemos que la red no pudo diferenciar los documentos correctamente. Además, parece que los resultados satisfactorios del test anterior estaban relacionados con el orden en que los documentos se presentaron a la red.

Esta medición se puede ver completa en el archivo
Tesis/TestPool/InglesEspaniol/fart/TestFART2.htm

3. Otra medición pero con un cambio de parámetros: ahora no se usaron stop list ni stemming.

   Los resultados fueron los siguientes:

```
Documento         Clase
---------------------------
espa1.txt         0
ingles2.txt       0
ingles3.txt       0
ingles4.txt       0
ingles6.txt       0
ingles7.txt       0
ingles1.txt       1
ingles8.txt       0
espa2.txt         1
ingles5.txt       1
espa3.txt         1
```

   De nuevo, la red no pudo clasificar correctamente.

   Esta medición se puede ver completa en el archivo
   Tesis/TestPool/InglesEspaniol/fart/TestFART3.htm

4. Otra medición con estos parámetros:

```
Eliminaci\'on de Tags HTML: No
Usar Stop List y Stemming: No.
Alfabeto: Letras y blanco.
Alfa: 0.
Beta: 1.0.
Rho: 0.9.
CantidadClusters: 10.
```

   El resultado fue incorrecto. La bitácora completa de esta medición está en */Tesis/TestPool/OtrosFart/TestFART1.htm.*

```
Documento     Clase
-----------------------
espa1.txt     0
ingles2.txt   0
ingles3.txt   0
ingles4.txt   0
ingles6.txt   0
ingles7.txt   0
ingles1.txt   1
ingles8.txt   0
espa2.txt     1
ingles5.txt   1
espa3.txt     1
```

Cuando los documentos son presentados en forma mezclada, la red no sabe cómo separarlos.

Los resultados indican que la FART no puede diferenciar documentos escritos en ingés de documentos escritos en castellano.

6.4 Mediciones con Dos Clases Conceptuales

Aquí, se explora la capacidad de la representación para discriminar documentos de dos clases conceptuales como un paso previo para determinar si esto se puede hacer a gran escala.

6.4.1 Mediciones con K-medias

Las siguientes mediciones se hicieron con el algoritmo de las k-medias. De nuevo se hicieron con el programa *TestKMediasDocs.exe*. Las mediciones se aplicaron sobre distintos conjuntos de documentos.

Palm Pilot versus Bases de Datos en C++

En esta medición se usaron nueve documentos, de los cuales seis eran sobre la Palm Pilot y los otros tres de un libro sobre bases de datos en Visual C++. Estos documentos son en formato HTML y se encuentran en el directorio *Tesis/TestPool/DosClases/1/*.

Un ejemplo de la interacción con el programa que calcula el clustering de los documentos:

```
Ingrese la cantidad de clases esperadas: 2
Uso blanco? (s/n): s
Uso stop list y stemming (ingles)? (s/n): s
Tiro tags HTML? (s/n): s
Archivo de docs: c:/Inetpub/wwwroot/tesis/testpool/dosclases/1/1.dir
Archivo de salida: c:/Inetpub/wwwroot/tesis/testpool/dosclases/1/res1.txt
```

En todos los casos se aplicaron las claves aportadas por los *meta keywords*. Se hacen varias corridas con los mismos datos, ya que la elección de los primeros dos vectores se hace al azar. Los resultados de las corridas son los siguientes:

1. Corrida con 2 clases, usando blancos, usando stemming y stop lists. Esta corrida se hizo usando los documentos ubicados en el directorio */tesis/testpool/dosclases/1/*.

```
Documento             Clase
-----------------------------
palm5.htm             1
palm2.htm             1
palm3.htm             1
palm4.htm             1
palm1.htm             1
palm6.htm             1
vcg04.htm             0
vcg03.htm             0
vcg05.htm             0
```

   Aquí, se puede ver que el resultado coincide con el esperado.

   Se realizó otra corrida con los mismos parámetros y el resultado fue el mismo (salvo renumerado de las clases):

```
Documento             Clase
-----------------------------
palm5.htm             0
palm2.htm             0
palm3.htm             0
palm4.htm             0
palm1.htm             0
palm6.htm             0
vcg04.htm             1
vcg03.htm             1
vcg05.htm             1
```

   Otra corrida más y un nuevo éxito:

```
Documento             Clase
-----------------------------
palm5.htm             0
palm2.htm             0
palm3.htm             0
palm4.htm             0
palm1.htm             0
palm6.htm             0
vcg04.htm             1
vcg03.htm             1
vcg05.htm             1
```

2. Corrida con blancos, sin aplicar stemming pero sí tirando los tags HTML:

   El clasificador funcionó bien.

```
Documento             Clase
------------------------------
palm5.htm             0
palm2.htm             0
palm3.htm             0
palm4.htm             0
palm1.htm             0
palm6.htm             0
vcg04.htm             1
vcg03.htm             1
vcg05.htm             1
```

3. Corrida con blancos, sin aplicar stemming ni tirando los tags HTML; funcionó bien.

```
Documento             Clase
------------------------------
palm5.htm             0
palm2.htm             0
palm3.htm             0
palm4.htm             0
palm1.htm             0
palm6.htm             0
vcg04.htm             1
vcg03.htm             1
vcg05.htm             1
```

4. Una vez más. . . y funcionó bien.

```
Documento             Clase
------------------------------
palm5.htm             0
palm2.htm             0
palm3.htm             0
palm4.htm             0
palm1.htm             0
palm6.htm             0
vcg04.htm             1
vcg03.htm             1
vcg05.htm             1
```

5. Corrida con blancos, sin stemming pero con los tags HTML:

```
Documento             Clase
------------------------------
palm5.htm             0
palm2.htm             0
palm3.htm             0
palm4.htm             0
```

```
palm1.htm             0
palm6.htm             0
vcg04.htm             1
vcg03.htm             1
vcg05.htm             1
```

Los resultados obtenidos fueron satisfactorios.

6. Corrida sin blancos, con stemming y stop list y tirando los tags HTML:

```
Documento             Clase
-----------------------------
palm5.htm             1
palm2.htm             1
palm3.htm             1
palm4.htm             1
palm1.htm             1
palm6.htm             1
vcg04.htm             0
vcg03.htm             0
vcg05.htm             0
```

Funcionó bien.

7. Corrida sobre la misma muestra pero calculando la distancia euclídea de cada vector a su centroide respectivo. La medida se hizo usando stemming, usando blanco, usando stop list y tirando los tags HTML.

```
Cantidad de Docs: 9
Cantidad de clases: 2
Uso blancos: Si
Uso stemming y stoplist: Si
Tirar tags: Si

Documento   Clase   Distancia con centroide
-------------------------------------------
palm5.htm   0       13.577563
palm2.htm   0       10.844114
palm3.htm   0       15.647186
palm4.htm   0       13.612128
palm1.htm   0       12.448748
palm6.htm   0       11.446465
vcg04.htm   1       16.960625
vcg03.htm   1       20.747650
vcg05.htm   1       16.615425
```

8. Ahora, otra corrida con los mismos parámetros pero calculando la dispersión en el cada cluster.

La fórmula que se usó para calcular la dispersión en la clase j es la siguiente:

$$Dispersion_j = \frac{1}{N_j} \cdot \sum_{i=1}^{N_j} |X_i - Z_j|^2 \tag{6.2}$$

donde N_i es la cantidad de vectores en la clase j.

```
Cantidad de Docs: 9
Cantidad de clases: 2
Uso blancos: Si
Uso stemming y stoplist: Si
Tirar tags: Si

Documento  Clase   Distancia con centroide
--------------------------------------------------
palm5.htm  1       15.160474
palm2.htm  1       14.884890
palm3.htm  1       17.252062
palm4.htm  1       16.185032
palm1.htm  1       14.806178
palm6.htm  1       16.568197
vcg04.htm  1       30.416870
vcg03.htm  0        0.000000
vcg05.htm  1       29.719828

Clase   Dispersion
----------------------
0        0.000000
1      414.146362
```

Aquí, las cosas no dieron como se esperaba. Si bien los los documentos *vcg04.htm* y *vcg05.htm* estaban claramente lejos el centroide de la clase 1, el algoritmo no los reacomodó.

9. Otra medición más en las mismas condiciones:

```
Cantidad de Docs: 9
Cantidad de clases: 2
Uso blancos: Si
Uso stemming y stoplist: Si
Tirar tags: Si

Documento     Clase    Distancia con centroide
-----------------------------------------------
palm5.htm     1        13.577563
palm2.htm     1        10.844114
palm3.htm     1        15.647186
palm4.htm     1        13.612128
palm1.htm     1        12.448748
```

```
palm6.htm     1        11.446465
vcg04.htm     0        16.960625
vcg03.htm     0        20.747650
vcg05.htm     0        16.615425

Clase  Dispersion
----------------------
0      331.400024
1      169.677063
```

Esta medición dio en forma satisfactoria.

10. Otra medición con otros parámetros:

```
Cantidad de Docs: 9
Cantidad de clases: 2
Uso blancos: Si
Uso stemming y stoplist: No
Tirar tags: No

Documento     Clase    Distancia con centroide
-------------------------------------------------
palm5.htm     0        13.501898
palm2.htm     0        11.669211
palm3.htm     0        16.659861
palm4.htm     0        14.075050
palm1.htm     0        12.675407
palm6.htm     0        11.995979
vcg04.htm     1        16.361837
vcg03.htm     1        19.954266
vcg05.htm     1        15.717479

Clase  Dispersion
----------------------
0      183.116536
1      304.307190
```

Esta medición dio resultados satisfactorios.

11. Otra medición más, que también dio bien:

```
Cantidad de Docs: 9
Cantidad de clases: 2
Uso blancos: Si
Uso stemming y stoplist: Si
Tirar tags: No

Documento     Clase  Distancia con centroide
-------------------------------------------------
```

```
palm5.htm     1      12.445627
palm2.htm     1      10.751185
palm3.htm     1      16.700100
palm4.htm     1      13.179738
palm1.htm     1      11.870475
palm6.htm     1      11.002831
vcg04.htm     0      16.436357
vcg03.htm     0      19.791071
vcg05.htm     0      15.765175

Clase  Dispersion
----------------------
0      303.460347
1      164.175140
```

12. Otra medición con los mismos documentos, pero ahora se estudia la matriz de similitud entre documentos para analizar relación entre la dispersión de los mismos y la disposición de las clases encontradas.

```
Cantidad de Docs: 9

Cantidad de clases: 2
Uso blancos: Si
Uso stemming y stoplist: Si
Tirar tags: Si

Documento     Clase     Distancia con centroide
-----------------------------------------------
palm5.htm          1     13.57
palm2.htm          1     10.82
palm3.htm          1     15.64
palm4.htm          1     13.46
palm1.htm          1     12.46
palm6.htm          1     11.45
vcg04.htm          0     17.07
vcg03.htm          0     20.68
vcg05.htm          0     16.74

Clase        Dispersion
--------------------------------
0   333.1058
1   168.9012


Distancias entre centroides


     d(0, 0) :       0.00
     d(0, 1) :      34.70
```

```
d(1, 0) :      34.70
d(1, 1) :       0.00


Distancias cuadradas entre centroides


d^2(0, 0) :       0.00
d^2(0, 1) :    1203.82
d^2(1, 0) :    1203.82
d^2(1, 1) :       0.00


Distancias de vector mas cercano y mas lejano de cada centroide


    Cluster     Mas Cercano          Mas Lejano
-----------------------------------------------------
          0        16.742926         20.677101
          1        10.815556         15.636584
```

La siguiente es la matriz de similitud entre documentos considerando la distancia euclídea entre los mismos:

	0	1	2	3	4	5	6	7	8
0	0.00	18.08	23.05	21.85	19.23	20.33	39.37	40.90	38.95
1	18.08	0.00	21.51	19.23	17.50	15.79	41.37	42.92	41.12
2	23.05	21.51	0.00	22.71	22.71	21.32	40.45	41.91	40.11
3	21.85	19.23	22.71	0.00	20.10	18.28	40.73	42.62	40.76
4	19.23	17.50	22.71	20.10	0.00	18.68	39.78	41.32	39.58
5	20.33	15.79	21.32	18.28	18.68	0.00	43.46	44.48	43.10
6	39.37	41.37	40.45	40.73	39.78	43.46	0.00	34.02	26.76
7	40.90	42.92	41.91	42.62	41.32	44.48	34.02	0.00	33.53
8	38.95	41.12	40.11	40.76	39.58	43.10	26.76	33.53	0.00

En la siguiente tabla se pueden ver las distancias al cuadrado entre documentos:

	0	1	2	3	4	5	6	7	8
0	0.00	326.80	531.26	477.25	369.83	413.28	1550.07	1673.16	1517.15
1	326.80	0.00	462.74	369.88	306.42	249.42	1711.86	1842.21	1690.72
2	531.26	462.74	0.00	515.89	515.89	454.64	1635.95	1756.76	1608.70
3	477.25	369.88	515.89	0.00	404.16	334.02	1658.86	1816.80	1661.46
4	369.83	306.42	515.89	404.16	0.00	348.94	1582.83	1707.63	1566.39
5	413.28	249.42	454.64	334.02	348.94	0.00	1888.52	1978.76	1857.28
6	1550.07	1711.86	1635.95	1658.86	1582.83	1888.52	0.00	1157.65	716.00
7	1673.16	1842.21	1756.76	1816.80	1707.63	1978.76	1157.65	0.00	1124.28
8	1517.15	1690.72	1608.70	1661.46	1566.39	1857.28	716.00	1124.28	0.00

En estas tablas, se puede observar que los documentos pueden ser separados correctamente por el algoritmo porque las distancias entre los documentos no relacionados son bien mayores a la distancia mismos con respecto a sus respectivos centroides. Esta situación le permite formar al algoritmo grupos hiperesféricos de documentos. También se puede ver que la distancia al cuadrado entre los documentos no relacionados es mucho mayor que la dispersión (entendida como el promedio del cuadrado de las distancias de los documentos con su centroide).

13. Ahora, se describe otra medición para mostrar el efecto de suponer un número mayor de clases al correcto.

```
Cantidad de Docs: 9

Cantidad de clases: 3
Uso blancos: Si
Uso stemming y stoplist: Si
Tirar tags: Si

Documento     Clase     Distancia con centroide
--------------------------------------------------
palm5.htm          0     13.17
palm2.htm          0     10.32
palm3.htm          1      0.00
palm4.htm          0     13.16
palm1.htm          0     11.91
palm6.htm          0     11.19
vcg04.htm          2     17.07
vcg03.htm          2     20.68
vcg05.htm          2     16.74

Clase        Dispersion
--------------------------------
0   144.0004
1     0.0000
2   333.1058


Distancias entre centroides


     d(0, 0) :       0.00
     d(0, 1) :      18.76
     d(0, 2) :      35.16
     d(1, 0) :      18.76
     d(1, 1) :       0.00
     d(1, 2) :      36.52
     d(2, 0) :      35.16
     d(2, 1) :      36.52
     d(2, 2) :       0.00
```

```
Distancias cuadradas entre centroides


     d^2(0, 0) :       0.00
     d^2(0, 1) :     352.08
     d^2(0, 2) :    1236.46
     d^2(1, 0) :     352.08
     d^2(1, 1) :       0.00
     d^2(1, 2) :    1334.02
     d^2(2, 0) :    1236.46
     d^2(2, 1) :    1334.02
     d^2(2, 2) :       0.00


Distancias de vector mas cercano y mas lejano de cada centroide


     Cluster     Mas Cercano          Mas Lejano
-----------------------------------------------------------
          0        10.320081         13.169391
          1        0.000000         0.000000
          2        16.742926         20.677101
```

Lo ocurrido puede explicarse de la siguiente manera: Los documentos fueron separados correctamente. Sin embargo, al particionar una clase real en otras más pequeñas, se crean clases ficticias.

14. El mismo caso que el anterior, pero con más clases:

```
Cantidad de Docs: 9

Cantidad de clases: 4
Uso blancos: Si
Uso stemming y stoplist: Si
Tirar tags: Si

Documento     Clase     Distancia con centroide
-----------------------------------------------------
palm5.htm          2     12.99
palm2.htm          2     10.04
palm3.htm          0      0.00
palm4.htm          2     12.63
palm1.htm          1      0.00
palm6.htm          2     10.65
vcg04.htm          3     17.07
vcg03.htm          3     20.68
vcg05.htm          3     16.74
```

```
Clase        Dispersion
--------------------------------
0     0.0000
1     0.0000
2   135.6665
3   333.1058


Distancias entre centroides


     d(0, 0) :       0.00
     d(0, 1) :      22.71
     d(0, 2) :      18.85
     d(0, 3) :      36.52
     d(1, 0) :      22.71
     d(1, 1) :       0.00
     d(1, 2) :      14.89
     d(1, 3) :      35.86
     d(2, 0) :      18.85
     d(2, 1) :      14.89
     d(2, 2) :       0.00
     d(2, 3) :      35.62
     d(3, 0) :      36.52
     d(3, 1) :      35.86
     d(3, 2) :      35.62
     d(3, 3) :       0.00


Distancias cuadradas entre centroides


     d^2(0, 0) :       0.00
     d^2(0, 1) :     515.89
     d^2(0, 2) :     355.47
     d^2(0, 3) :    1334.02
     d^2(1, 0) :     515.89
     d^2(1, 1) :       0.00
     d^2(1, 2) :     221.67
     d^2(1, 3) :    1285.84
     d^2(2, 0) :     355.47
     d^2(2, 1) :     221.67
     d^2(2, 2) :       0.00
     d^2(2, 3) :    1268.46
     d^2(3, 0) :    1334.02
     d^2(3, 1) :    1285.84
     d^2(3, 2) :    1268.46
     d^2(3, 3) :       0.00
```

```
Distancias de vector mas cercano y mas lejano de cada centroide


     Cluster     Mas Cercano         Mas Lejano
-----------------------------------------------------
          0        0.000000        0.000000
          1        0.000000        0.000000
          2        10.042916        12.987201
          3        16.742926        20.677101
```

En este caso, se crearon dos clases con un único elemento cada una.

Java versus Palm Pilot

Luego, se realizaron mediciones para separar otro conjunto de documentos:

1. Algunos capítulos de un libro de *Java*: los documentos CH05.HTM, CH02.HTM, CH03.HTM, CH04.HTM, CH01.HTM, CH06.HTM, CH07.HTM, CH08.HTM, CH09.HTM y CH10.HTM.
2. Páginas sobre el PDA *Palm Pilot*: los documentos palm2.htm, palm3.htm, palm4.htm, palm5.htm, palm6.htm y palm1.htm.

Las mediciones realizadas fueron las siguientes:

1. En la siguiente medición se supusieron dos clases de documentos:

```
Cantidad de Docs: 16

Cantidad de clases: 2
Uso blancos: Si
Uso stemming y stoplist: Si
Tirar tags: Si

Documento     Clase     Distancia con centroide
-------------------------------------------------
CH05.HTM          1     20.20
CH02.HTM          1     22.01
CH03.HTM          1     19.38
CH04.HTM          1     19.02
CH01.HTM          1     21.10
CH06.HTM          1     17.13
CH07.HTM          1     19.22
CH08.HTM          1     18.43
CH09.HTM          1     18.87
CH10.HTM          1     18.93
palm2.htm          0     10.86
palm3.htm          0     15.66
palm4.htm          0     13.44
```

```
palm5.htm          0     13.78
palm6.htm          0     11.53
palm1.htm          0     12.37

Clase        Dispersion
--------------------------------
0   169.9320
1   379.2008


Distancias entre centroides


     d(0, 0) :       0.00
     d(0, 1) :      24.22
     d(1, 0) :      24.22
     d(1, 1) :       0.00


Distancias cuadradas entre centroides


     d^2(0, 0) :       0.00
     d^2(0, 1) :     586.52
     d^2(1, 0) :     586.52
     d^2(1, 1) :       0.00


Distancias de vector mas cercano y mas lejano de cada centroide


     Cluster     Mas Cercano          Mas Lejano
-----------------------------------------------------------
          0        10.861692         15.659389
          1        17.126085         22.006216
```

Los documentos fueron correctamente separados. La explicación de ello es exactamente la misma que en el caso visto anteriormente. La distancia entre los documentos y su centroide es mucho menor a la distancia entre los centroides.

A continuación, se reproducen las matrices de similitud tomando distancia euclídea y distancia al cuadrado entre documentos.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0.00	30.50	30.16	30.36	31.05	28.21	29.10	28.74	28.85	29.07	34.45	34.96	34.56	33.24	36.16	33.47
1	30.50	0.00	29.59	31.08	31.28	29.95	30.22	31.75	32.11	32.16	39.45	38.98	38.83	37.33	41.39	37.86
2	30.16	29.59	0.00	26.19	29.41	26.70	29.25	29.57	29.87	29.59	35.08	35.04	35.17	32.35	36.71	32.89
3	30.36	31.08	26.19	0.00	29.18	26.07	29.48	27.99	28.79	28.72	32.49	32.67	32.74	30.41	33.93	31.03
4	31.05	31.28	29.41	29.18	0.00	28.58	31.22	30.07	30.37	31.07	33.95	34.48	34.24	32.76	35.73	33.05
5	28.21	29.95	26.70	26.07	28.58	0.00	27.41	25.60	26.73	26.47	31.48	32.42	31.76	29.72	33.27	30.47
6	29.10	30.22	29.25	29.48	31.22	27.41	0.00	27.50	27.20	27.92	36.02	36.08	35.82	34.45	37.74	34.74
7	28.74	31.75	29.57	27.99	30.07	25.60	27.50	0.00	26.68	25.89	31.28	31.44	31.69	30.17	32.66	30.50
8	28.85	32.11	29.87	28.79	30.37	26.73	27.20	26.68	0.00	26.09	30.92	31.57	31.22	30.15	32.11	30.38
9	29.07	32.16	29.59	28.72	31.07	26.47	27.92	25.89	26.09	0.00	32.19	32.51	31.96	31.09	33.47	31.30
10	34.45	39.45	35.08	32.49	33.95	31.48	36.02	31.28	30.92	32.19	0.00	21.51	19.23	18.37	15.97	17.39
11	34.96	38.98	35.04	32.67	34.48	32.42	36.08	31.44	31.57	32.51	21.51	0.00	22.69	23.24	21.40	22.71
12	34.56	38.83	35.17	32.74	34.24	31.76	35.82	31.69	31.22	31.96	19.23	22.69	0.00	21.97	18.32	20.00
13	33.24	37.33	32.35	30.41	32.76	29.72	34.45	30.17	30.15	31.09	18.37	23.24	21.97	0.00	20.59	19.38
14	36.16	41.39	36.71	33.93	35.73	33.27	37.74	32.66	32.11	33.47	15.97	21.40	18.32	20.59	0.00	18.56
15	33.47	37.86	32.89	31.03	33.05	30.47	34.74	30.50	30.38	31.30	17.39	22.71	20.00	19.38	18.56	0.00

	0	1	2	3	4	5	6	7
0	0.00	930.43	909.91	921.89	964.31	796.04	846.90	825.74
1	930.43	0.00	875.51	966.17	978.50	897.25	913.37	1008.09
2	909.91	875.51	0.00	685.68	864.76	712.63	855.65	874.11
3	921.89	966.17	685.68	0.00	851.30	679.89	868.93	783.33
4	964.31	978.50	864.76	851.30	0.00	817.06	974.47	904.27
5	796.04	897.25	712.63	679.89	817.06	0.00	751.42	655.59
6	846.90	913.37	855.65	868.93	974.47	751.42	0.00	756.02
7	825.74	1008.09	874.11	783.33	904.27	655.59	756.02	0.00
8	832.43	1031.02	892.51	829.01	922.48	714.54	739.72	712.03
9	845.25	1034.31	875.56	825.12	965.38	700.57	779.78	670.28
10	1186.58	1556.58	1230.83	1055.35	1152.81	990.71	1297.34	978.59
11	1222.47	1519.10	1227.92	1067.21	1188.54	1050.92	1301.41	988.75
12	1194.34	1507.87	1237.14	1071.73	1172.21	1008.58	1283.41	1004.43
13	1104.94	1393.68	1046.46	924.94	1073.54	883.08	1187.11	910.21
14	1307.28	1713.41	1347.88	1150.98	1276.29	1107.16	1423.95	1066.66
15	1119.91	1433.39	1081.47	962.92	1092.62	928.64	1206.94	930.18

	8	9	10	11	12	13	14	15
0	832.43	845.25	1186.58	1222.47	1194.34	1104.94	1307.28	1119.91
1	1031.02	1034.31	1556.58	1519.10	1507.87	1393.68	1713.41	1433.39
2	892.51	875.56	1230.83	1227.92	1237.14	1046.46	1347.88	1081.47
3	829.01	825.12	1055.35	1067.21	1071.73	924.94	1150.98	962.92
4	922.48	965.38	1152.81	1188.54	1172.21	1073.54	1276.29	1092.62
5	714.54	700.57	990.71	1050.92	1008.58	883.08	1107.16	928.64
6	739.72	779.78	1297.34	1301.41	1283.41	1187.11	1423.95	1206.94
7	712.03	670.28	978.59	988.75	1004.43	910.21	1066.66	930.18
8	0.00	680.53	955.82	996.94	974.46	908.72	1030.75	922.87
9	680.53	0.00	1036.41	1056.64	1021.47	966.51	1120.14	979.89
10	955.82	1036.41	0.00	462.78	369.88	337.33	255.00	302.46
11	996.94	1056.64	462.78	0.00	514.69	539.92	457.97	515.52
12	974.46	1021.47	369.88	514.69	0.00	482.66	335.59	399.87
13	908.72	966.51	337.33	539.92	482.66	0.00	424.05	375.48
14	1030.75	1120.14	255.00	457.97	335.59	424.05	0.00	344.33
15	922.87	979.89	302.46	515.52	399.87	375.48	344.33	0.00

También, en este caso se puede observar que la distancia entre los documentos no relacionados es mucho mayor a la dispersión de las clases.

2. En la siguiente medición se ve la misma muestra pero indicando más clases:

```
Cantidad de Docs: 16

Cantidad de clases: 3
Uso blancos: Si
Uso stemming y stoplist: Si
Tirar tags: Si

Documento      Clase     Distancia con centroide
-------------------------------------------------
CH05.HTM           0     20.20
CH02.HTM           0     22.01
CH03.HTM           0     19.38
CH04.HTM           0     19.02
CH01.HTM           0     21.10
CH06.HTM           0     17.13
CH07.HTM           0     19.22
CH08.HTM           0     18.43
CH09.HTM           0     18.87
CH10.HTM           0     18.93
palm2.htm          2     10.39
palm3.htm          1      0.00
palm4.htm          2     13.14
palm5.htm          2     13.37
```

```
palm6.htm         2     11.26
palm1.htm         2     11.81

Clase        Dispersion
--------------------------------
0   379.2008
1     0.0000
2   145.0660


Distancias entre centroides


     d(0, 0) :       0.00
     d(0, 1) :      27.98
     d(0, 2) :      24.62
     d(1, 0) :      27.98
     d(1, 1) :       0.00
     d(1, 2) :      18.79
     d(2, 0) :      24.62
     d(2, 1) :      18.79
     d(2, 2) :       0.00


Distancias cuadradas entre centroides


     d^2(0, 0) :       0.00
     d^2(0, 1) :     782.79
     d^2(0, 2) :     606.12
     d^2(1, 0) :     782.79
     d^2(1, 1) :       0.00
     d^2(1, 2) :     353.11
     d^2(2, 0) :     606.12
     d^2(2, 1) :     353.11
     d^2(2, 2) :       0.00


Distancias de vector mas cercano y mas lejano de cada centroide


     Cluster     Mas Cercano          Mas Lejano
-----------------------------------------------------------
          0        17.126085        22.006216
          1         0.000000         0.000000
          2        10.385929        13.373055
```

El algoritmo separó los documentos correctamente. Sin embargo, creó una clase ficticia con un único documento.

3. Otra medición del estilo de la anterior:

```
Cantidad de Docs: 16

Cantidad de clases: 3
Uso blancos: Si
Uso stemming y stoplist: Si
Tirar tags: Si

Documento     Clase     Distancia con centroide
-----------------------------------------------------
CH05.HTM          2     19.83
CH02.HTM          0     17.77
CH03.HTM          0     16.67
CH04.HTM          2     19.24
CH01.HTM          0     17.67
CH06.HTM          2     16.83
CH07.HTM          2     18.62
CH08.HTM          2     17.27
CH09.HTM          2     17.70
CH10.HTM          2     17.67
palm2.htm         1     10.86
palm3.htm         1     15.66
palm4.htm         1     13.44
palm5.htm         1     13.78
palm6.htm         1     11.53
palm1.htm         1     12.37

Clase        Dispersion
--------------------------------
0   302.0901
1   169.9320
2   330.9223


Distancias entre centroides


     d(0, 0) :       0.00
     d(0, 1) :      28.63
     d(0, 2) :      16.46
     d(1, 0) :      28.63
     d(1, 1) :       0.00
     d(1, 2) :      23.83
     d(2, 0) :      16.46
     d(2, 1) :      23.83
     d(2, 2) :       0.00
```

```
Distancias cuadradas entre centroides


     d^2(0, 0) :       0.00
     d^2(0, 1) :     819.75
     d^2(0, 2) :     271.09
     d^2(1, 0) :     819.75
     d^2(1, 1) :       0.00
     d^2(1, 2) :     567.90
     d^2(2, 0) :     271.09
     d^2(2, 1) :     567.90
     d^2(2, 2) :       0.00


Distancias de vector mas cercano y mas lejano de cada centroide


     Cluster     Mas Cercano          Mas Lejano
-----------------------------------------------------------
          0        16.673569        17.774208
          1        10.861692        15.659389
          2        16.825285        19.827347
```

Nuevamente, los documentos fueron separados correctamente. Sin embargo, una clase se separó en dos clases ficticias.

Relojes Breitling versus Java

A continuación se muestran los resultados de las mediciones para otro conjunto de prueba: los documentos HTML del sitio de los relojes *Breitling* y los capítulos de un libro de *Java*.

Los documentos se dividen en dos clases:

1. *Relojes Breitling:* Breitling Chrono Avenger.htm, Breitling New in 2000.htm, Breitling Produits.htm, Breitling Produits2.htm, Breitling Produits3.htm, Breitling Produits4.htm, Breitling Produits5.htm, Breitling Produits6.htm, Breitling Produits7.htm y Breitling Produits8.htm.
2. *Java:* CH05.HTM, CH02.HTM, CH03.HTM, CH04.HTM, CH01.HTM, CH06.HTM, CH07.HTM, CH08.HTM, CH09.HTM y CH10.HTM.

Los parámetros usados en la medición eran los siguientes:

```
Cantidad de Docs: 20

Cantidad de clases: 2
Uso blancos: Si
Uso stemming y stoplist: Si
Tirar tags: Si
```

Los resultados de las mediciones fueron los siguientes:

```
Documento                       Clase     Distancia con centroide
-----------------------------------------------------------------
Breitling Chrono Avenger.htm    1          14.26
Breitling New in 2000.htm       1           6.12
Breitling Produits.htm          1          10.55
Breitling Produits2.htm         1           5.89
Breitling Produits3.htm         1           6.17
Breitling Produits4.htm         1          10.74
Breitling Produits5.htm         1          10.07
Breitling Produits6.htm         1          10.30
Breitling Produits7.htm         1          10.14
Breitling Produits8.htm         1           8.00
CH05.HTM                        0          20.23
CH02.HTM                        0          21.75
CH03.HTM                        0          19.19
CH04.HTM                        0          19.06
CH01.HTM                        0          21.09
CH06.HTM                        0          17.09
CH07.HTM                        0          19.28
CH08.HTM                        0          18.49
CH09.HTM                        0          18.87
CH10.HTM                        0          18.83

Clase        Dispersion
--------------------------------
0            377.5239
1            91.4636


Distancias entre centroides


     d(0, 0) :       0.00
     d(0, 1) :      28.40
     d(1, 0) :      28.40
     d(1, 1) :       0.00


Distancias cuadradas entre centroides


     d^2(0, 0) :       0.00
     d^2(0, 1) :     806.31
     d^2(1, 0) :     806.31
     d^2(1, 1) :       0.00


Distancias de vector mas cercano y mas lejano de cada centroide
```

```
   Cluster      Mas Cercano          Mas Lejano
-----------------------------------------------------------
         0         17.088039         21.754419
         1          5.893647         14.257151
```

La matriz de similitud entre documentos explica el porqué de la correcta separación de los documentos. La distancia entre documentos no relacionados es mucho mayor a la distancia entre centroides.

	0	1	2	3	4	5	6	7	8	9
0	0.00	17.65	18.79	17.55	17.54	18.30	18.49	18.62	17.94	17.94
1	17.65	0.00	12.70	1.70	4.51	12.95	14.14	14.39	11.93	8.69
2	18.79	12.70	0.00	12.56	12.82	15.66	15.54	15.73	15.75	14.46
3	17.55	1.70	12.56	0.00	4.18	12.83	14.01	14.26	11.80	8.52
4	17.54	4.51	12.82	4.18	0.00	13.20	13.88	14.13	11.84	8.89
5	18.30	12.95	15.66	12.83	13.20	0.00	16.57	16.75	15.33	13.85
6	18.49	14.14	15.54	14.01	13.88	16.57	0.00	2.65	16.28	14.47
7	18.62	14.39	15.73	14.26	14.13	16.75	2.65	0.00	16.44	14.63
8	17.94	11.93	15.75	11.80	11.84	15.33	16.28	16.44	0.00	13.45
9	17.94	8.69	14.46	8.52	8.89	13.85	14.47	14.63	13.45	0.00
10	34.55	37.67	36.23	37.65	37.39	35.77	35.16	35.15	36.03	36.72
11	38.65	42.56	40.70	42.54	42.23	40.32	39.87	39.83	40.91	41.53
12	35.00	38.32	36.61	38.28	38.01	36.46	36.24	36.25	36.70	37.37
13	33.04	36.05	34.60	36.01	35.68	34.41	33.97	33.94	34.74	35.23
14	34.14	37.47	35.94	37.45	37.18	35.77	35.42	35.42	36.11	36.58
15	32.61	35.18	34.01	35.15	34.93	33.41	33.28	33.26	33.84	34.29
16	36.14	39.17	37.61	39.14	38.93	37.21	36.89	36.88	37.77	38.21
17	32.07	34.46	33.22	34.43	34.07	32.74	32.62	32.64	33.11	33.72
18	31.53	33.69	32.54	33.66	33.43	32.07	32.24	32.24	32.67	32.99
19	32.32	34.50	33.40	34.46	34.34	32.64	32.68	32.67	33.25	33.70

	10	11	12	13	14	15	16	17	18	19
0	34.55	38.65	35.00	33.04	34.14	32.61	36.14	32.07	31.53	32.32
1	37.67	42.56	38.32	36.05	37.47	35.18	39.17	34.46	33.69	34.50
2	36.23	40.70	36.61	34.60	35.94	34.01	37.61	33.22	32.54	33.40
3	37.65	42.54	38.28	36.01	37.45	35.15	39.14	34.43	33.66	34.46
4	37.39	42.23	38.01	35.68	37.18	34.93	38.93	34.07	33.43	34.34
5	35.77	40.32	36.46	34.41	35.77	33.41	37.21	32.74	32.07	32.64
6	35.16	39.87	36.24	33.97	35.42	33.28	36.89	32.62	32.24	32.68
7	35.15	39.83	36.25	33.94	35.42	33.26	36.88	32.64	32.24	32.67
8	36.03	40.91	36.70	34.74	36.11	33.84	37.77	33.11	32.67	33.25
9	36.72	41.53	37.37	35.23	36.58	34.29	38.21	33.72	32.99	33.70
10	0.00	30.25	30.01	30.42	31.11	28.21	29.03	28.91	28.91	29.16
11	30.25	0.00	29.31	30.91	30.99	29.77	29.99	31.46	31.84	32.07
12	30.01	29.31	0.00	25.99	29.21	26.58	29.13	29.42	29.65	29.48
13	30.42	30.91	25.99	0.00	29.07	26.07	29.65	28.17	28.89	28.64
14	31.11	30.99	29.21	29.07	0.00	28.43	31.33	30.17	30.41	31.14
15	28.21	29.77	26.58	26.07	28.43	0.00	27.43	25.67	26.71	26.34
16	29.03	29.99	29.13	29.65	31.33	27.43	0.00	27.67	27.35	27.82
17	28.91	31.46	29.42	28.17	30.17	25.67	27.67	0.00	26.88	25.56
18	28.91	31.84	29.65	28.89	30.41	26.71	27.35	26.88	0.00	25.75
19	29.16	32.07	29.48	28.64	31.14	26.34	27.82	25.56	25.75	0.00

Al incrementar la cantidad de clases esperadas, el algoritmo de las k-medias creó, como en casos anteriores, clases ficticias pero separó correctamente los documentos.

```
Cantidad de Docs: 20

Cantidad de clases: 4
Uso blancos: Si
Uso stemming y stoplist: Si
```

Tirar tags: Si

Documento	Clase	Distancia con centroide
Breitling Chrono Avenger.htm	0	0.00
Breitling New in 2000.htm	3	2.82
Breitling Produits.htm	2	10.19
Breitling Produits2.htm	3	2.55
Breitling Produits3.htm	3	3.53
Breitling Produits4.htm	2	10.70
Breitling Produits5.htm	2	8.07
Breitling Produits6.htm	2	8.27
Breitling Produits7.htm	2	10.54
Breitling Produits8.htm	3	6.33
CH05.HTM	1	20.23
CH02.HTM	1	21.75
CH03.HTM	1	19.19
CH04.HTM	1	19.06
CH01.HTM	1	21.09
CH06.HTM	1	17.09
CH07.HTM	1	19.28
CH08.HTM	1	18.49
CH09.HTM	1	18.87
CH10.HTM	1	18.83

Clase	Dispersion
0	0.0000
1	377.5239
2	92.5814
3	16.7399

Distancias entre centroides

d(0, 0) : 0.00
d(0, 1) : 27.99
d(0, 2) : 15.72
d(0, 3) : 17.19
d(1, 0) : 27.99
d(1, 1) : 0.00
d(1, 2) : 27.76
d(1, 3) : 30.86
d(2, 0) : 15.72
d(2, 1) : 27.76
d(2, 2) : 0.00
d(2, 3) : 8.46
d(3, 0) : 17.19
d(3, 1) : 30.86

```
     d(3, 2) :       8.46
     d(3, 3) :       0.00


Distancias cuadradas entre centroides


     d^2(0, 0) :       0.00
     d^2(0, 1) :     783.20
     d^2(0, 2) :     247.07
     d^2(0, 3) :     295.52
     d^2(1, 0) :     783.20
     d^2(1, 1) :       0.00
     d^2(1, 2) :     770.89
     d^2(1, 3) :     952.57
     d^2(2, 0) :     247.07
     d^2(2, 2) :     770.89
     d^2(2, 2) :       0.00
     d^2(2, 3) :      71.51
     d^2(3, 0) :     295.52
     d^2(3, 1) :     952.57
     d^2(3, 2) :      71.51
     d^2(3, 3) :       0.00


Distancias de vector mas cercano y mas lejano de cada centroide


     Cluster     Mas Cercano         Mas Lejano
-----------------------------------------------------------
          0        0.000000        0.000000
          1        17.088039        21.754419
          2        8.065623        10.699879
          3        2.551321        6.328360
```

6.4.2 Mediciones con la FART

Aquí, se describen las mediciones sobre los mismos documentos que en la sección anterior pero usando la red neuronal de la teoría de la resonancia adaptativa difusa.

Palm Pilot versus Bases de Datos en C++

Los documentos de esta sección son los mismos que en la correspondiente realizada con el algoritmo de las K-medias. Todos los archivos de la evolución de la clasificación de estos documentos pueden hallarse en el directorio *Tesis/TestPool/DosClases/1/fart/* Las mediciones con estos documentos en formato HTML son las siguientes:

1. Los parámetros usados en esta medición son los siguientes:

   ```
   Eliminacion de Tags HTML: Si
   ```

```
Usar Stop List y Stemming: Si
Alfabeto: Letras y blanco
Alfa: 1000
Beta: 1.0
Rho: 0.9
CantidadClusters: 10
```

Los resultados son los siguientes:

```
Documento      Clase
-----------------------
palm5.htm      0
palm2.htm      0
palm3.htm      0
palm4.htm      0
palm1.htm      0
palm6.htm      0
vcg04.htm      1
vcg03.htm      1
vcg05.htm      1
```

Vemos que aquí la red clasificó correctamente.

Estos resultados corresponden al archivo *TestFart1.htm*.

2. Otro test con los siguientes parámetros:

```
Eliminacion de Tags HTML: No
Usar Stop List y Stemming: Si
Alfabeto: Letras y blanco
Alfa: 1000
Beta: 1.0
Rho: 0.9
CantidadClusters: 10
```

Los resultados son los siguientes:

```
Documento      Clase
-----------------------
palm5.htm      0
palm2.htm      0
palm3.htm      0
palm4.htm      0
palm1.htm      0
palm6.htm      0
vcg04.htm      1
vcg03.htm      1
vcg05.htm      1
```

Los resultados son análogos a los anteriores y la red clasificó correctamente a los documentos. Estos resultados corresponden al archivo *TestFart2.htm*.

3. Otra prueba con los parámetros:

```
Eliminacion de Tags HTML: Si
Usar Stop List y Stemming: No
Alfabeto: Letras y blanco
Alfa: 1000
Beta: 1.0
Rho: 0.9
CantidadClusters: 10
```

   Los resultados son los siguientes y se hallan en el archivo *TestFart3.htm*.

```
Documento      Clase
-----------------------
palm5.htm      0
palm2.htm      0
palm3.htm      0
palm4.htm      0
palm1.htm      0
palm6.htm      0
vcg04.htm      1
vcg03.htm      1
vcg05.htm      1
```

   De vuelta, la red clasificó bien.

4. Otra medición con los parámetros:

```
Eliminacion de Tags HTML: No
Usar Stop List y Stemming: No
Alfabeto: Letras y blanco
Alfa: 1000
Beta: 1.0
Rho: 0.9
CantidadClusters: 10
```

   Los resultados fueron los siguientes y corresponden al archivo *TestFart4.htm*.

```
Documento      Clase
-----------------------
palm5.htm      0
palm2.htm      0
palm3.htm      0
palm4.htm      0
palm1.htm      0
palm6.htm      0
vcg04.htm      1
vcg03.htm      1
vcg05.htm      1
```

Nuevamente, la red funcionó bien.

5. Otra medición con los siguientes parámetros y orden de los documentos:

```
Eliminacion de Tags HTML: Si
Usar Stop List y Stemming: Si
Alfabeto: Letras
Alfa: 1000
Beta: 1.0
Rho: 0.9
CantidadClusters: 10.
```

   El archivo completo de esta corrida está en el archivo *TestFart5.htm*. Los resultados fueron análogos a los anteriores:

```
Documento      Clase
-----------------------
palm5.htm      0
palm2.htm      0
palm3.htm      0
palm4.htm      0
palm1.htm      0
palm6.htm      0
vcg04.htm      1
vcg03.htm      1
vcg05.htm      1
```

6. Otra corrida con los parámetros:

```
Eliminacion de Tags HTML: No
Usar Stop List y Stemming: Si
Alfabeto: Letras
Alfa: 1000
Beta: 1.0
Rho: 0.9
CantidadClusters: 10
```

   Los resultados fueron los mismos y están en el archivo *TestFart6.htm*.

7. Otra corrida con los parámetros:

```
Eliminacion de Tags HTML: Si
Usar Stop List y Stemming: No
Alfabeto: Letras
Alfa: 1000
Beta: 1.0
Rho: 0.9
CantidadClusters: 10
```

   Los resultados fueron nuevamente los mismos y están en el archivo *TestFart7.htm*.

8. Otra corrida con los parámetros:

```
Eliminacin de Tags HTML: No
Usar Stop List y Stemming: No
Alfabeto: Letras
Alfa: 1000
Beta: 1.0
Rho: 0.9
CantidadClusters: 10
```

   Y los resultados fueron también los mismos y se pueden encontrar en el archivo *TestFart8.htm*.

9. *Un caso donde los resultados son malos:* Ahora, tenemos una corrida donde se intercambió el orden en que los documentos fueron presentados a la red.

   Los parámetros usados fueron los siguientes:

```
Eliminacion de Tags HTML: Si
Usar Stop List y Stemming: Si
Alfabeto: Letras
Alfa: 1000
Beta: 1.0
Rho: 0.9
CantidadClusters: 10
```

   Los resultados completos de la corrida están en el archivo *TestFart9.htm*. Los resultados, de acuerdo al orden de los documentos son los siguientes:

```
Documento      Clase
-----------------------
palm5.htm      0
vcg04.htm      0
palm2.htm      1
palm3.htm      1
vcg03.htm      2
palm4.htm      1
vcg05.htm      2
palm1.htm      1
palm6.htm      1
```

   Analizando los resultados, podemos descubrir lo siguiente:

   - Aparecieron más de dos clases, lo cual no es malo en sí mismo.
   - El problema radica en que documentos de clases "conceptuales" distintas fueron apareados en la misma clase. Tal es el caso de los documentos *palm5.htm* y *vcg04.htm*, los cuales aparecen juntos en la clase 0.
   - El resto de los documentos fueron separados correctamente: los de la serie *Palm* aparecen en la clase 1 mientras que los de la serie *C++* lo hacen en la clase 2.

10. *Salvando el caso anterior:* A pesar del fracaso del caso anterior, no se puede afirmar que la FART no sirve para resolver el problema. Incrementando el parámetro ρ que tiene como efecto disminuir el tamaño de los clusters y, por lo tanto, aumentar la precisión, la red logra separar correctamente a los documentos.

    Usando los parámetros:

    Eliminación de Tags HTML: Si

    Usar Stop List y Stemming: Si.

    Alfabeto: Letras y blanco.

    Alfa: 0.

    Beta: 1.0.

    Rho: 0.92.

    Se obtuvieron los resultados:

```
Url          Cluster
---------------------
palm5.htm     0
vcg04.htm     1
palm2.htm     0
palm3.htm     0
vcg03.htm     1
palm4.htm     0
vcg05.htm     2
palm1.htm     0
palm6.htm     0
```

    La red separó correctamente los documentos pero con el costo de agregar un nuevo cluster.

    Además, se reafirma la hipótesis de que la formación de clusters es dependiente del orden en que ingresan los patrones. Este es el precio a pagar por no necesitar reentrenamiento ni múltiples pasadas de patrones.

Java versus Palm Pilot

En esta sección se repiten las mediciones de la sección correspondiente del K-medias pero usando la red de la FART. Nuevamente, se hicieron mediciones para ver cómo era el comportamiento de esta red neuronal para separar los documentos.

Las mediciones fueron las siguientes:

1. En esta medición se usaron los siguientes parámetros:

    Eliminación de Tags HTML: Si

    Usar Stop List y Stemming: Si.

    Alfabeto: Letras y blanco.

    Alfa: 0.

Beta: 1.0.

Rho: 0.92.

Cantidad de Clusters: 10.

Los resultados obtenidos fueron:

```
Url          Cluster
----------------------
CH05.HTM     0
CH02.HTM     0
CH03.HTM     0
CH04.HTM     1
CH01.HTM     1
CH06.HTM     1
CH07.HTM     2
CH08.HTM     2
CH09.HTM     2
CH10.HTM     3
palm2.htm    3
palm3.htm    3
palm4.htm    3
palm5.htm    4
palm6.htm    4
palm1.htm    4
```

La red usa más custers que el algoritmo de las k-medias. Pero separa correctamente los documentos salvo *ch10* que se mezcla con los de *Palm*.

2. *Salvando la medida anterior:* Nuevamente, fue prematuro suponer que la red no era capaz de separar los documentos. Un aumento en ρ salvó la medida.

   Los parámetros usados fueron:

   Eliminación de Tags HTML: Si

   Usar Stop List y Stemming: Si.

   Alfabeto: Letras y blanco.

   Alfa: 0.

   Beta: 1.0.

   Rho: 0.94.

   CantidadClusters: 10.

   Y los resultados fueron:

```
Url          Cluster
----------------------
CH05.HTM       0
CH02.HTM       0
CH03.HTM       1
```

```
CH04.HTM       1
CH01.HTM       2
CH06.HTM       2
CH07.HTM       3
CH08.HTM       3
CH09.HTM       4
CH10.HTM       4
palm2.htm      5
palm3.htm      5
palm4.htm      5
palm5.htm      5
palm6.htm      5
palm1.htm      5
```

Los documentos fueron separados correctamente a expensas de la creación de un número mayor de clusters.

3. En esta medición, la red fue capaz de agrupar correctamente, dos patrones con los que no había sido entrenada.

```
Eliminacion de Tags HTML: Si
Usar Stop List y Stemming: Si.
Alfabeto: Letras y blanco.
Alfa: 0.
Beta: 1.0.
Rho: 0.92.
CantidadClusters: 10
```

```
Url            Cluster
----------------------
CH05.HTM       0
CH02.HTM       0
CH03.HTM       1
CH04.HTM       1
CH01.HTM       1
CH07.HTM       2
CH08.HTM       2
CH09.HTM       2
CH10.HTM       3
palm3.htm      3
palm5.htm      3
palm6.htm      4
palm1.htm      4
CH06.HTM       0
palm4.htm      4
```

Relojes Breitling versus Java

Aquí, se repiten las pruebas equivalentes a las del k-medias aplicadas a los documentos HTML del sitio de los relojes Breitling y algunos capítulos del libro de Java, pero esta vez realizadas con la FART.

Se describen dos medidas: en la primera los documentos fueron correctamente separados salvo una excepción; en la segunda, se incrementó ρ para tener mayor precisión y los documentos fueron entonces correctamente separados en su totalidad.

1. En esta medición se usaron los parámetros:

   Eliminación de Tags HTML: Si

   Usar Stop List y Stemming: Si.

   Alfabeto: Letras y blanco.

   Alfa: 0.0.

   Beta: 1.0.

   Rho: 0.9.

   CantidadClusters: 10.

   Los resultados fueron:

```
Url                              Cluster
---------------------------------------
Breitling Chrono Avenger.htm        0
Breitling New in 2000.htm           0
Breitling Produits.htm              0
Breitling Produits2.htm             0
Breitling Produits3.htm             0
Breitling Produits4.htm             0
Breitling Produits5.htm             0
Breitling Produits6.htm             0
Breitling Produits7.htm             0
Breitling Produits8.htm             0
CH05.HTM                            0
CH02.HTM                            1
CH03.HTM                            1
CH04.HTM                            1
CH01.HTM                            1
CH06.HTM                            2
CH07.HTM                            2
CH08.HTM                            2
CH09.HTM                            2
CH10.HTM                            2
```

   El documento *ch05* se mezcló con los de la clase anterior. El resto fueron correctamente separados.

2. En esta medición se incrementó el parámetro ρ para tener más precisión en la formación de nuevas clases. Los parámetros son los mismos que en la medición anterior, salvo que $\rho = 0.92$.

   Los resultados fueron:

Url	Cluster
Breitling Chrono Avenger.htm	0
Breitling New in 2000.htm	0
Breitling Produits.htm	0
Breitling Produits2.htm	0
Breitling Produits3.htm	0
Breitling Produits4.htm	0
Breitling Produits5.htm	0
Breitling Produits6.htm	0
Breitling Produits7.htm	0
Breitling Produits8.htm	0
CH05.HTM	1
CH02.HTM	1
CH03.HTM	1
CH04.HTM	2
CH01.HTM	2
CH06.HTM	2
CH07.HTM	3
CH08.HTM	3
CH09.HTM	3
CH10.HTM	4

   Los documentos fueron separados correctamente.

6.5 Mediciones Orientadas al Largo de Documentos

En estas medidas se quizo determinar si la representación sabe agrupar documentos que deben estar en la misma clase conceptual pero con medidas muy diferentes.

Los archivos de esta prueba están en el directorio */TestPool/DosClases/3/* y son los siguientes:

- Breitling Produits2.htm
- Por10Breitling Produits2.htm
- Anatomy.htm
- Por10Anatomy.htm

Donde los dos primeros archivos corresponden al archivo original y su versión multiplicada por diez en extensión mediante la técnica de cortar y pegar; análogamente, lo mismo ocurre con los dos últimos. Un par de documentos no tiene relación con el otro ya que pertenecen a temáticas diferentes.

6.5.1 Mediciones con K-medias

Aquí, se documenta la medición hecha sobre estos documentos con el algoritmo de las k-medias.

1. Pretendo tener dos clases. Para ello haré lo siguiente, voy a tener dos documentos distintos, y los mismos multiplicados por 10 en extensión. Los archivos de bitácora de estas mediciones están en el directorio *Tesis/TestPool/DosClases/3/kmedias3*.

```
Cantidad de Docs: 4
Cantidad de clases: 2
Uso blancos: Si
Uso stemming y stoplist: Si
Tirar tags: Si

Documento                          Clase    Distancia con centroide
-------------------------------------------------------------------
Breitling Produits2.htm            0        0.608886
Por10Anatomy.htm                   1        3.696265
Por10Breitling Produits2.htm       0        0.608886
Anatomy.htm                        1        3.696265

Clase  Dispersion
----------------------
0       0.370743
1      13.662378
```

   La bitácora completa de esta medida está en el archivo *sal3.txt*.

   Los resultados de esta medida indican que la representación es invariante al largo de los textos.

6.5.2 Mediciones con la FART

Se repitió el mismo experimento pero con la red de resonancia adaptativa difusa. El seguimiento completo de la medida está en los archivos del subdirectorio *fart*.

Las mediciones fueron las siguientes:

1. Una medida con los parámetros:

```
Eliminacion de Tags HTML: Si
Usar Stop List y Stemming: Si.
Alfabeto: Letras y blanco.
Alfa: 1000.
Beta: 1.0.
Rho: 0.9.
CantidadClusters: 10.
```

La bitácora completa de esta medida está en el archivo *TestFart.htm*.

Los resultados fueron los siguientes (los documentos se especifican en el orden en que fueron presentados a la red neuronal):

```
Documento                       Clase
-----------------------------------
Breitling Produits2.htm         0
Anatomy.htm                     1
Por10Anatomy.htm                1
Por10Breitling Produits2.htm    0
```

Los resultados indican que la red fue capaz de agrupar los documentos correctamente irrespectivamente de su largo.

2. Otra medida con los parámetros:

```
Eliminacion de Tags HTML: Si
Usar Stop List y Stemming: Si
Alfabeto: Letras y blanco
Alfa: 1000
Beta: 1.0
Rho: 0.9
CantidadClusters: 10
```

La bitácora de esta medida está en el archivo *TestFart2.htm*. Los resultados fueron:

```
Documento                       Clase
-----------------------------------
Breitling Produits2.htm         0
Por10Anatomy.htm                1
Por10Breitling Produits2.htm    0
Anatomy.htm                     1
```

Aquí, vemos que los documentos (presentados en otro orden) fueron correctamente clasificados por la red neuronal.

6.6 Mediciones con Varias Clases de Documentos

En esta sección se analizan mediciones con clustering de documentos con el algoritmo de las K-medias y los modelos de redes neuronales CPN, KSOM y FART. Esta vez, se estudia el comportamiento de los métodos de clasificación cuando los documentos pertenecen a varias clases conceptuales.

La lista de documentos separados en las clases esperadas es la siguiente:

1. Sobre la serie de televisión *Highlander*: Faq.txt, high4.htm, peter2.htm, peter.htm, The Richie Memorial Soundtrack.htm, Search for highlander 4.htm, Marina's Highlander Page.htm.

2. Capıtulos de un libro de *Java*: CH05.HTM, CH02.HTM, CH03.HTM, CH04.HTM, CH01.HTM, CH06.HTM, CH07.HTM, CH08.HTM, CH09.HTM, CH10.HTM.

3. Capítulos de un libro de *C++*: vcg03.htm, vcg04.htm, vcg05.htm.

4. Sobre el PDA *Palm Pilot*: palm2.htm, palm3.htm, palm4.htm, palm5.htm, palm6.htm, palm1.htm.

5. Sobre los relojes *Breitling*: Breitling Chrono Avenger.htm, Breitling New in 2000.htm, Breitling Produits.htm, Breitling Produits2.htm, Breitling Produits3.htm, Breitling Produits4.htm, Breitling Produits5.htm, Breitling Produits6.htm, Breitling Produits7.htm y Breitling Produits8.htm.

6.6.1 Mediciones con K-medias

Se hicieron las siguientes mediciones:

1. En esta primera medición, sólo se usaron tres clases de documentos. El algoritmo de las k-medias los separó correctamente.

```
Cantidad de Docs: 19

Cantidad de clases: 3
Uso blancos: Si
Uso stemming y stoplist: Si
Tirar tags: Si

Documento     Clase    Distancia con centroide
-------------------------------------------------
CH05.HTM         0     20.210928
CH02.HTM         1     19.757427
CH03.HTM         0     19.499220
CH04.HTM         0     18.824253
CH01.HTM         0     21.134048
CH06.HTM         0     16.908762
CH07.HTM         0     19.161245
CH08.HTM         0     18.110840
CH09.HTM         0     18.435369
CH10.HTM         0     18.468323
vcg03.htm        1     22.459448
vcg04.htm        1     17.636517
vcg05.htm        1     17.812748
palm2.htm        2     10.962114
palm3.htm        2     15.639450
palm4.htm        2     13.442506
palm5.htm        2     13.843060
palm6.htm        2     11.544771
palm1.htm        2     12.436668

Clase        Dispersion
```

```
--------------------------------
0   361.3006
1   380.7809
2   170.8407


Distancias entre centroides


     d(0, 0) : 0.000000
     d(0, 1) : 18.896053
     d(0, 2) : 23.640003
     d(1, 0) : 18.896053
     d(1, 1) : 0.000000
     d(1, 2) : 33.176151
     d(2, 0) : 23.640003
     d(2, 1) : 33.176151
     d(2, 2) : 0.000000


Distancias cuadradas entre centroides


     d^2(0, 0) : 0.000000
     d^2(0, 1) : 357.060822
     d^2(0, 2) : 558.849731
     d^2(1, 0) : 357.060822
     d^2(1, 1) : 0.000000
     d^2(1, 2) : 1100.656982
     d^2(2, 0) : 558.849731
     d^2(2, 1) : 1100.656982
     d^2(2, 2) : 0.000000


Distancias de vector mas cercano y mas lejano de cada centroide


     Cluster     DMasCerca        DMasLejos
-----------------------------------------------------------
          0        16.908762        21.134048
          1        17.636517        22.459448
          2        10.962114        15.639450
```

2. En esta medición se evidencia que los documentos de la serie *vcg* y los *ch* están muy cerca entre sí. El algoritmo de las k-medias no los separa correctamente:

3. Primero, se realizó una medición suponiendo que había 6 clases de documentos con los siguientes parámetros:

```
Cantidad de Docs: 27
Cantidad de clases: 6
Uso blancos: Si
Uso stemming y stoplist: Si
Tirar tags: Si
```

El resultado obtenido fue:

```
Documento          Clase     Distancia con centroide
-------------------------------------------------
Faq.txt             4      37.727936
high4.htm           3      13.222043
peter2.htm          5      11.298969
peter.htm           2      0.000000
The Richie...htm    3      19.681658
Search highl.htm    0      0.000000
Marina's High.htm   5     11.298969
CH05.HTM            4     20.780630
CH02.HTM            4     20.820068
CH03.HTM            4     19.800432
CH04.HTM            4     20.463768
CH01.HTM            4     21.946783
CH06.HTM            4     18.896231
CH07.HTM            4     19.574770
CH08.HTM            4     20.231264
CH09.HTM            4     20.556973
CH10.HTM            4     20.553324
seven.htm           3     17.426676
vcg03.htm           4     26.871544
vcg04.htm           4     22.196028
vcg05.htm           4     21.762882
palm2.htm           3     13.579535
palm3.htm           1     14.662004
palm4.htm           1     13.157313
palm5.htm           1     13.193653
palm6.htm           3     12.391019
palm1.htm           1     12.064942

Clase  Dispersion
----------------------
0      0.000000
1    176.931137
2      0.000000
3    240.764038
4    518.866420
5    127.666702
```

Se puede ver que los documentos de la serie Highlander fueron separados en varias clases, esto se puede explicar basándose en que las distancias de estos documentos

entre sí es mayor que en los de las otras clases como se vio en las medidas del capítulo 4.

Los documentos de la serie CH están juntos pero también unidos a los de C++ (vg...).

También se puede ver que los documentos de la serie Palm Pilot están juntos, a pesar de estar en dos clases (1 y 3).

4. Otra corrida con los mismos documentos y parámetros:

```
Cantidad de Docs: 27
Cantidad de clases: 6
Uso blancos: Si
Uso stemming y stoplist: Si
Tirar tags: Si
```

Los resultados fueron:

```
Documento                              Clase     Distancia con centroide
-----------------------------------------------------------------------
Faq.txt                                3         30.989807
high4.htm                              1         15.128151
peter2.htm                             0         15.599105
peter.htm                              0         17.336391
The Richie Memorial Soundtrack.htm     5         0.000000
Search for highlander 4.htm            0         15.156948
Marina's Highlander Page.htm           1         16.241371
CH05.HTM                               2         20.037029
CH02.HTM                               3         20.569035
CH03.HTM                               2         18.419212
CH04.HTM                               2         18.149647
CH01.HTM                               2         20.340372
CH06.HTM                               2         16.584023
CH07.HTM                               2         19.401640
CH08.HTM                               2         18.297766
CH09.HTM                               4         12.993454
CH10.HTM                               4         12.993454
seven.htm                              1         18.988237
vcg03.htm                              3         24.599272
vcg04.htm                              3         18.632650
vcg05.htm                              3         19.285046
palm2.htm                              1         11.861135
palm3.htm                              1         16.625462
palm4.htm                              1         14.514598
palm5.htm                              1         15.372019
palm6.htm                              1         11.382506
palm1.htm                              1         13.614860

Clase  Dispersion
----------------------
```

```
0       257.871867
1       225.798570
2       352.878871
3       541.533252
4       168.829849
5         0.000000
```

Nuevamente, los documentos de la serie Highlander mostraron tener poca relación entre ellos mismos. Los documentos de la serie Palm aparecieron todos juntos, lo cual es correcto. Los documentos sobre bases de datos en C aparecieron juntos, lo cual está bien. También, los documentos de la serie CH (salvo por dos excepciones) aparecieron bien clasificados.

5. Otra medición con los mismos documentos pero esta vez suponiendo que la cantidad de clases era 5 y con los siguientes parámetros:

```
Cantidad de Docs: 27
Cantidad de clases: 5
Uso blancos: Si
Uso stemming y stoplist: Si
Tirar tags: Si
```

Los resultados obtenidos fueron:

```
Documento                          Clase Distancia con centroide
-------------------------------------------------------------------
Faq.txt                                4 34.139751
high4.htm                              3 15.084669
peter2.htm                             3 16.487732
peter.htm                              1 14.919231
The Richie Memorial Soundtrack.htm     1 14.919231
Search for highlander 4.htm            3 19.314318
Marina's Highlander Page.htm           3 15.886867
CH05.HTM                               4 21.673414
CH02.HTM                               4 19.926001
CH03.HTM                               4 21.152710
CH04.HTM                               2 18.245316
CH01.HTM                               0  0.000000
CH06.HTM                               2 16.112135
CH07.HTM                               4 20.566366
CH08.HTM                               2 16.555052
CH09.HTM                               2 17.255951
CH10.HTM                               2 16.798845
seven.htm                              3 18.789183
vcg03.htm                              4 25.361582
vcg04.htm                              4 19.705652
vcg05.htm                              4 19.864641
palm2.htm                              3 12.581549
palm3.htm                              3 16.974974
```

```
palm4.htm                                3 15.004211
palm5.htm                                3 15.828267
palm6.htm                                3 12.065149
palm1.htm                                3 14.251615

Clase  Dispersion
----------------------
0       0.000000
1     222.583466
2     289.306250
3     249.876642
4     541.105530
```

Los documentos de la serie Palm fueron agrupados correctamente. Los documentos de C++ se mezclaron con los de Java quienes fueron separados en dos clases. Los documentos de Highlander se repartieron con el de Startrek.

6. Se realizó otra prueba suponiendo 5 clases con los parámetros:

```
Cantidad de Docs: 27
Cantidad de clases: 5
Uso blancos: Si
Uso stemming y stoplist: Si
Tirar tags: Si
```

Los resultados fueron:

```
Documento     Clase     Distancia con centroide
-------------------------------------------------
Faq.txt                                  2 32.290348
high4.htm                                3 14.008338
peter2.htm                               0  0.000000
peter.htm                                1 21.400543
The Richie Memorial Soundtrack.htm       3 18.832066
Search for highlander 4.htm              3 17.142485
Marina's Highlander Page.htm             3 13.907357
CH05.HTM                                 1 20.290871
CH02.HTM                                 2 20.297489
CH03.HTM                                 1 19.339823
CH04.HTM                                 1 18.649622
CH01.HTM                                 1 20.987497
CH06.HTM                                 1 16.947449
CH07.HTM                                 2 21.385178
CH08.HTM                                 1 18.228342
CH09.HTM                                 1 18.702652
CH10.HTM                                 1 18.593222
seven.htm                                3 17.194981
vcg03.htm                                2 24.818769
vcg04.htm                                2 18.951693
vcg05.htm                                2 19.218351
```

```
palm2.htm                              4 10.842659
palm3.htm                              4 15.633821
palm4.htm                              4 13.455783
palm5.htm                              4 13.888109
palm6.htm                              4 11.506332
palm1.htm                              4 12.308367

Clase  Dispersion
----------------------
0       0.000000
1     371.888943
2     542.743937
3     266.765381
4     169.968140
```

En esta corrida los documentos de la seria Palm aparecieron correctamente clasificados. Sin embargo, los demás documentos aparecieron mal clasificados. Esto se puede atribuir al número inferior de clases.

6.6.2 Mediciones con la CPN y el KSOM

Las mediciones son las siguientes.

Medición 1

Esta medición se hizo con la red CPN con los parámetros: cantidad de clusters = 5, tiempo de entrenamiento = 40, alfabeto con el blanco, usando stop list y stemming y eliminando marcadores HTML, velocidad de aprendizaje=0.01. Los pesos iniciales se obtienen aleatoriamente.

Se observa que todos los documentos son ubicados en el cluster número 3 (columna CPN_1 de la tabla 6.6.2). La explicación es que la inicialización de los pesos en forma aleatoria hace que el primer ganador gane siempre ya que todos las demás unidades quedaron lejos del espacio de patrones.

Medición 2

Esta medición se hizo con la red CPN con los parámetros: cantidad de clusters = 6, tiempo de entrenamiento = 20, se usan blancos, velocidad de aprendizaje = 0.01, se eliminaron marcadores HTML y se usa stop list y stemming.

Los representantes iniciales de los clusters fueron elegidos a mano y son los documentos: faq.txt, ch05.htm, palm1.htm, vcg04.htm, peter.htm y The Ritchie....

Los documentos se presentaron a la red 20 veces cada uno siempre en el mismo orden y no en forma aleatoria.

Los resultados están en la columna CPN_2 de la tabla 6.6.2. Se puede observar que sólo los documentos de la serie *ch0i.htm* fueron clasificados incorrectamente. Sin embargo, se estos documentos sobre C++ fueron agrupadoscon los documentos sobre Java.

Documento	*Clase* CPN_1	*Clase* CPN_2	*Clase KSOM*
Faq.txt	3	0	0
CH05.HTM	3	1	1
palm1.htm	3	2	2
vcg04.htm	3	3	3
peter.htm	3	4	4
The Richie Memorial Soundtrack.htm	3	5	5
high4.htm	3	4	6
peter2.htm	3	4	7
Search for highlander 4.htm	3	4	8
Marina's Highlander Page.htm	3	4	9
CH02.HTM	3	3	3
CH03.HTM	3	3	3
CH04.HTM	3	3	3
CH01.HTM	3	3	3
CH06.HTM	3	3	3
CH07.HTM	3	3	3
CH08.HTM	3	3	3
CH09.HTM	3	3	3
CH10.HTM	3	3	3
vcg03.htm	3	3	3
vcg05.htm	3	3	3
palm2.htm	3	2	2
palm3.htm	3	2	2
palm4.htm	3	2	2
palm5.htm	3	2	2
palm6.htm	3	2	2

Tabla 6.1: Medidas con la CPN y el KSOM.

Medición 3

Esta medición se hizo con la red KSOM achicando la ventana de actualización de pesos progresivamente. Los parámetros usados son: cantidad de clusters = 10, tiempo de entrenamiento = 50, velocidad de aprendizaje = 0.01, se usó el blanco, eliminar marcadores HTML y stemming y stop list.

Los centroides de los clusters se inicializaron con los primeros 10 documentos.

Los resultados son equivalentes a los de la medición 2 y están en la columna $KSOM$ de la tabla 6.6.2.

6.6.3 Mediciones con la FART

1. En esta medida se usaron los parámetros:

```
Eliminacion de Tags HTML: Si
Usar Stop List y Stemming: Si.
Alfabeto: Letras y blanco.
Alfa: 0.
Beta: 1.0.
```

```
Rho: 0.92.
CantidadClusters: 10.
```

La red separó los documentos en forma correcta. La configuración de clases es la siguiente:

```
Url                              Cluster
-----------------------------------------
Breitling Chrono Avenger.htm        0
Breitling New in 2000.htm           0
Breitling Produits.htm              0
Breitling Produits2.htm             0
Breitling Produits3.htm             0
Breitling Produits4.htm             0
Breitling Produits5.htm             0
Breitling Produits6.htm             0
Breitling Produits7.htm             0
Breitling Produits8.htm             0
CH05.HTM                            1
CH02.HTM                            1
CH03.HTM                            2
CH04.HTM                            2
CH01.HTM                            2
CH06.HTM                            1
CH07.HTM                            3
CH08.HTM                            1
CH09.HTM                            3
CH10.HTM                            3
palm2.htm                           4
palm3.htm                           4
palm4.htm                           4
palm5.htm                           4
palm6.htm                           4
palm1.htm                           4
```

2. Otra medición con los parámetros:

   Eliminación de Tags HTML: Si

   Usar Stop List y Stemming: Si

   Alfabeto: Letras y blanco

   Alfa: 1000

   Beta: 1.0

   Rho: 0.9

   La bitácora completa de esta medición está en el archivo */tesis/testpool/varios/varias /fart/testfart1.htm*. Los resultados fueron:

```
Documento                                Cluster
-----------------------------------------------
Faq.txt                                   0
high4.htm                                 1
peter2.htm                                1
peter.htm                                 1
The Richie Memorial Soundtrack.htm        1
Search for highlander 4.htm               1
Marina's Highlander Page.htm              1
CH05.HTM                                  2
CH02.HTM                                  2
CH03.HTM                                  2
CH04.HTM                                  2
CH01.HTM                                  3
CH06.HTM                                  3
CH07.HTM                                  3
CH08.HTM                                  3
CH09.HTM                                  4
CH10.HTM                                  4
vcg03.htm                                 4
vcg04.htm                                 0
vcg05.htm                                 5
palm2.htm                                 5
palm3.htm                                 6
palm4.htm                                 6
palm5.htm                                 5
palm6.htm                                 6
palm1.htm                                 6
```

A partir de los resultados de esta medición, se puede ver que si bien documentos afines no son agrupados en clases únicas, salvo en un sólo caso, los documentos no afines no son mezclados.

3. Otra medición exitosa:

```
Eliminacin de Tags HTML: Si
Usar Stop List y Stemming: Si.
Alfabeto: Letras y blanco.
Alfa: 0.
Beta: 1.0.
Rho: 0.94.
CantidadClusters: 10.


Documento                           Cluster
-------------------------------------------
Breitling Chrono Avenger.htm          0
Breitling New in 2000.htm             0
Breitling Produits.htm                0
Breitling Produits2.htm               0
```

```
Breitling Produits3.htm          0
Breitling Produits4.htm          0
Breitling Produits5.htm          0
Breitling Produits6.htm          0
Breitling Produits7.htm          0
Breitling Produits8.htm          0
CH05.HTM                         1
CH02.HTM                         1
CH03.HTM                         2
CH04.HTM                         2
CH01.HTM                         3
CH06.HTM                         3
CH07.HTM                         4
CH08.HTM                         4
CH09.HTM                         5
CH10.HTM                         5
palm2.htm                        6
palm3.htm                        6
palm4.htm                        6
palm5.htm                        6
palm6.htm                        6
palm1.htm                        6
```

6.7 Discusión

Las pruebas con el algoritmo de las K-Medias son exitosas cuando el número de clases es mayor o igual al correcto.

Las pruebas con la red de CPN apoyan los defectos de dicho modelo mencionados en la literatura[Freeman93] (también en la sección 5.6.7). Cuando los patrones están muy cerca entre sí, siempre resulta ganadora la misma unidad durante el entrenamiento. El significado de ello es que todos los patrones terminan en un único cluster. Esta situación se puede observar en la figura 6.12, en donde la misma está adaptada al caso 2D. En la figura de la izquierda, como todos los patrones están ubicados en una región definida del espacio, al inicializar cada clase con representantes aleatorios, hay unidades que quedan lejos de ganar la competencia y una se lleva a todos los patrones. Por el contrario, en la figura de la derecha, los centroides iniciales están mejor distribuidos y ganarán patrones acordemente dando una mejor clasificación de los mismos.

Cuando las unidades se inicializan con patrones representantes de clusters o al menos de los que se creen que son clusters), las pruebas dan en forma más satisfactoria. Dichos vectores serán los centroides de los clusters. Las sorpresas no desaparecen del todo pues puede suceder que dos clases se junten en una. Un ejemplo de ello se puede observar en la tabla 6.6.2, donde los patrones representando al libro de Java se unen con los del libro de C++; lo cual puede decirse que era esperado ya los textos trataban una misma área del conocimiento, a saber, lenguajes de programación.

Con respecto al entrenamiento de la FART, las pruebas indican que ésta funciona bien cuando los documentos se le presentan separados por clase. Siempre existe un valor de ρ tal que los documentos son separados correctamente. El caso extremo es cuando $\rho \to 1$ y se tiene un cluster por patrón. Una vez que la red está entrenada irá adaptando sus

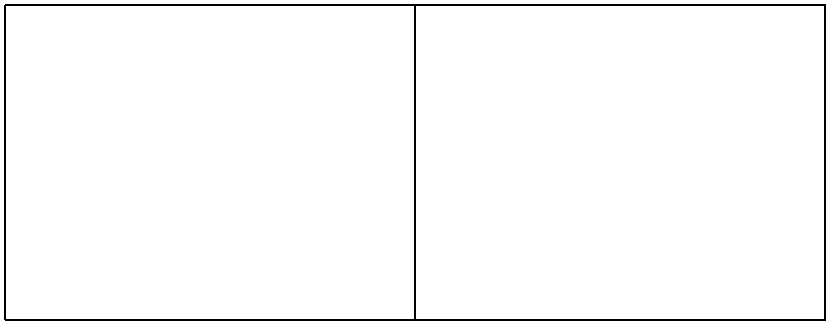

Figura 6.12: Contrapropagación en 2D.

Figura 6.13: Clases no circulares con método de clases circulares.

pesos de acuerdo a los documentos que fueran ingresando.

La FART (y todos los otros métodos) fallan cuando la forma de las clases que se quieren separar no son (hiper) circulares ni (hiper) rectangulares. Este parece ser el caso de los documentos en ingles y en castellano que no puedieron ser separados ni por el k-medias ni por la FART. Un ejemplo de esta situción se ve en la figura 6.13.

Se puede afirmar que la FART es una opción válida para resolver el problema de la clasificación de los documentos relevantes a un usuario debido a que si otros métodos pueden hallar una partición de éstos, la FART también lo hará.

La FART es muy sensible al orden de entrada de los patrones; Pero, para cualquier orden en la presentación de los mismos existe un ρ adecuado para realizar con éxito la separación.

6.8 Conclusiones

En este capítulo se expusieron el conjunto de mediciones complementarias a las del capítulo 4. Las mediciones realizadas consistieron en el clustering de conjuntos de documentos con los algoritmos de las K-medias, red de contrapropagación, red de Kohonen y red de la Teoría de la Resonancia Adaptativa Difusa. Las mediciones demostraron que esta última red es capaz de separar adecuadamente los documentos de entrada aprendiendo la distribución de clases de los mismos.

Capítulo 7

Mediciones con Método de Claves

En este capítulo se plantea el método de claves, que es un método de representación de documentos alternativo al de los trigramas. Se describe el método, la motivación de su uso y los resultados de las mediciones con los algoritmos de clustering realizadas con él. Finalmente, se analizan los resultados obtenidos.

7.1 Nuevo Vector de Características

7.1.1 Motivos del Cambio de Representación

Debido a los errores que, bajo condiciones reales, comete el algoritmo de la FART usando la representación de los trigramas, se decidió explorar otra representación de los documentos. La búsqueda de la nueva representación se orientó hacia las investigaciones ya probadas en la literatura de IR (veáse capítulo 3).

Una solución para que las pruebas con una red neuronal den bien consiste en eliminar de las pruebas a aquellos documentos que resultaban problemáticos [Skapura96]; sin embargo, esta es una solución temporal que a la larga no trae sino más problemas ya que lo que se está haciendo es reducir la efectividad de la solución con la red neuronal.

Otra forma de solucionar los problemas consiste en ajustar la representación de los datos [Skapura96]. El paso de la representación de trigramas a la de claves toma exactamente esa dirección.

En la siguiente parte se explica cuál es esa nueva representación.

7.1.2 Nueva Representación

Dado un perfil de filtrado formado por N claves $(k_1, \ldots, k_N)$. La nueva representación de un documento HTML o de texto D se obtiene de la siguiente manera:

1. Si el documento D es HTML, eliminar los marcadores HTML y las porciones de código JavaScript o VBScript para obtener un documento D_2 formado solamente por la lista de términos que componen el documento D original.

2. Si el documento D era HTML, obtener la lista de *meta keywords* L (si los hubiera).

Figura 7.1: Curva sigmoidea desplazada hacia la derecha en 5 unidades. La curva a es asintótica con 1 cuando $x \to \infty$.

3. Si la opción de filtrado de *stop words* está activa, eliminar de D_2 todas las palabras stop para obtener D_3; luego, aplicar *stemming* a los términos restantes para obtener D_4. Si dicha opción no está activa, entonces $D_4 \equiv D_2$.

4. Contar cuántas veces aparecen las claves de filtrado $k_1, \ldots, k_M$ en D_4 para obtener un vector de contadores $(c_1, \ldots, c_N)$.

5. A los contadores c_i distintos de 0 aplicarles la función sigmoide F desplazada hacia la derecha en 5 unidades para obtener un vector *fuzzyficado* $vf = (F(c_i), \ldots, F(c_N))$ (veáse ecuación 7.1 y figura 7.1).

$$F(x) = \frac{1}{1 + e^{-(x-5)}} \tag{7.1}$$

6. Setear a 1 a las componentes i de vf tal que sus claves de filtrado i estuvieran en la lista de meta keywords (previo stemming de éstas si correspondiera).

7.2 Programa de Análisis

Para realizar las mediciones con esta nueva representación, se modificó el programa usado para realizar las mediciones con la representación de trigramas. El programa de mediciones está basado en un formulario HTML (*http://localhost/querando/testFARTConClaves.htm*) donde el usuario ingresa los parámetros del generador de características (si se usa o no filtrado por stop list y stemming y las claves de filtrado para la representación de los documentos), el algoritmo de clustering a usar (algoritmo de las k-medias, red neuronal de contrapropagación o red neuronal de la teoría de la resonancia adaptativa difusa o sólo calcular la matriz de similitud entre documentos sumado a los parámetros de las redes) y la lista de URLs correspondientes a los documentos a clasificar (figura 7.2).

Figura 7.2: TestFARTConClaves.HTM es la interfaz de prueba con el nuevo método de representación de documentos.

Luego, todos estos datos son pasados a un programa CGI (*testfartconclaves.exe*), el cual ejecuta el algoritmo correspondiente y genera un documento HTML con los resultados (figura 7.3).

7.3 Mediciones con Dos Clases Conceptuales

En esta sección se exponen los resultados de las mediciones con dos clases conceptuales para determinar la performance de los algoritmos de clustering.

7.3.1 Palm Pilot versus Bases de Datos en C++

Los documentos en la prueba usados fueron:

Matriz de Similitud

Las claves de filtrado fueron: *palm, pilot, handheld, c, programming, language.* Los vectores que representan a estos documentos son:

- 0 - http://localhost/Tesis/TestPool/DosClases/1/palm5.htm : (1.000000 0.017986 0.006693 0.047426 0.006693 0.006693) y (1.000000 0.017986 0.000000 0.000000 0.000000 0.000000) (con y sin stemming respectivamente).
- 1 - http://localhost/Tesis/TestPool/DosClases/1/palm2.htm : (0.880797 0.017986 0.006693 0.006693 0.006693 0.006693) y (0.880797 0.017986 0.000000 0.000000 0.000000 0.000000).
- 2 - http://localhost/Tesis/TestPool/DosClases/1/palm3.htm : (0.993307 0.017986 0.006693 0.006693 0.006693 0.006693) y (0.993307 0.017986 0.000000 0.017986 0.000000 0.000000).

Figura 7.3: Salida de TestFARTConClaves.EXE con los resultados de la clasificación con la FART.

- 3 - http://localhost/Tesis/TestPool/DosClases/1/palm4.htm : (1.000000 0.047426 0.006693 0.006693 0.006693 0.006693) y (1.000000 0.047426 0.000000 0.000000 0.000000 0.000000).
- 4 - http://localhost/Tesis/TestPool/DosClases/1/palm1.htm : (1.000000 0.119203 0.006693 0.006693 0.006693 0.006693) y (1.000000 0.119203 0.000000 0.000000 0.000000 0.000000).
- 5 - http://localhost/Tesis/TestPool/DosClases/1/palm6.htm : (0.999089 0.017986 0.006693 0.006693 0.006693 0.006693) y (0.999089 0.017986 0.000000 0.000000 0.000000 0.000000).
- 6 - http://localhost/Tesis/TestPool/DosClases/1/vcg04.htm : (0.006693 0.006693 0.006693 0.999994 0.731059 0.982014) y (0.000000 0.000000 0.000000 0.000000 0.952574 0.982014).
- 7 - http://localhost/Tesis/TestPool/DosClases/1/vcg03.htm : (0.006693 0.006693 0.006693 1.000000 0.119203 0.017986) y (0.000000 0.000000 0.000000 0.000000 0.880797 0.017986).
- 8 - http://localhost/Tesis/TestPool/DosClases/1/vcg05.htm : (0.006693 0.006693 0.006693 1.000000 0.952574 1.000000) y (0.000000 0.000000 0.000000 0.000000 0.997527 1.000000).

Las matrices de similitud entre los documentos aparecen en las tablas 7.3.1 (7.1) (sin stemming) y 7.3.1 (7.2) (con stemming). Se puede observar que los documentos de la serie *Palm* están cerca entre sí mientras que están alejados de los de la serie *vcg* y viceversa. Esta cualidad del conjunto de los documentos es lo que posibilita el clustering de los mismos.

	0	1	2	3	4	5	6	7	8
0	0.000000	0.125970	0.041279	0.050258	0.109105	0.040743	1.835791	1.380931	1.943074
1	0.125970	0.000000	0.112510	0.122784	0.156378	0.118292	1.796327	1.328018	1.905832
2	0.041279	0.112510	0.000000	0.030191	0.101438	0.005782	1.853683	1.404628	1.959986
3	0.050258	0.122784	0.030191	0.000000	0.071777	0.029454	1.857666	1.409881	1.963754
4	0.109105	0.156378	0.101438	0.071777	0.000000	0.101221	1.860624	1.413776	1.966552
5	0.040743	0.118292	0.005782	0.029454	0.101221	0.000000	1.856766	1.408695	1.962903
6	1.835791	1.796327	1.853683	1.857666	1.860624	1.856766	0.000000	1.141804	0.222245
7	1.380931	1.328018	1.404628	1.409881	1.413776	1.408695	1.141804	0.000000	1.287967
8	1.943074	1.905832	1.959986	1.963754	1.966552	1.962903	0.222245	1.287967	0.000000

Tabla 7.1: Matriz de similitud entre serie *Palm* y *C++* sin stemming.

	0	1	2	3	4	5	6	7	8
0	0.000000	0.119203	0.019191	0.029440	0.101217	0.000911	1.694719	1.332836	1.730718
1	0.119203	0.000000	0.113939	0.122784	0.156378	0.118292	1.627229	1.245895	1.664688
2	0.019191	0.113939	0.000000	0.035142	0.103020	0.018893	1.690874	1.327943	1.726953
3	0.029440	0.122784	0.035142	0.000000	0.071777	0.029454	1.695287	1.333558	1.731274
4	0.101217	0.156378	0.103020	0.071777	0.000000	0.101221	1.698811	1.338035	1.734725
5	0.000911	0.118292	0.018893	0.029454	0.101221	0.000000	1.694181	1.332152	1.730192
6	1.694719	1.627229	1.690874	1.695287	1.698811	1.694181	0.000000	0.966696	0.048418
7	1.332836	1.245895	1.327943	1.333558	1.338035	1.332152	0.966696	0.000000	0.988927
8	1.730718	1.664688	1.726953	1.731274	1.734725	1.730192	0.048418	0.988927	0.000000

Tabla 7.2: Matriz de similitud entre serie *Palm* y *C++* con stemming.

Figura 7.4: Dos clusters con poca dispersión y centroides distantes entre sí.

Pruebas con K-Medias

El programa para ejecutar el algoritmo de las k-medias, realiza una mezcla aleatoria de los patrones de entrada antes de clasificar los mismos; por esta razón, siempre es necesario hacer varias pruebas con un mismo conjunto de patrones.

Las pruebas se realizaron indicando dos clases esperadas y se usó como variable si se aplicaba o no stemming y filtrado de palabras stop.

Los resultados de las pruebas fueron los siguientes:

1. KM1-CS: El resultado de la clasificación de los documentos está expresado en la tabla 1 (7.3) (los cuales fieron separados correctamente). En la tabla 1 (7.4) se observa la dispersión en cada clase. La distancia entre los centroides de las clases que se obtuvo es 1.514812.

   Se puede observar que el método funciona porque la distancia de los documentos a sus centroides es mucho menor que la de los centroides entre sí. Gráficamente, esto se puede observar en la figura 1 (7.4). La dispersión en el cluster número 0 es 0.003352, lo cual quiere decir que, aproximadamente, los vectores están a $\sqrt{0.003352} = 0.05$ unidades de su centroide; en el caso del cluster número 1, la dispersión es 0.212758, lo que quiere decir, que los puntos están a una distancia promedio de 0.46 unidades de su centroide. Además, a partir de las distancias con su centroide se puede corroborar esto: se ve que en el caso del cluster 0, la distancia máxima de un vector con su centroide es 0.10 mientras que en el caso del cluster 1 es de 0.65. Como la mitad de 1.51 es 0.755. Por lo tanto, los vectores están ubicados a una distancia de su centroide menor a este valor y eso hace que puedan separarse adecuadamente. El mismo análisis puede llevarse a cabo con las demás mediciones.

2. KM2-CS: En esta medición se usó stemming y stop list y dio los mismos resultados que la anterior salvo renombrado de clases (dieron en forma invertida). Ver archivo km2-cs.htm.

3. KM3-SS: En esta medición no se usó filtrado de stop list ni stemming (archivo km3-ss.htm), los resultados fueron los siguientes están en las tablas 3 (7.5) y 3 (7.6). Se pueden hacer las mismas consideraciones que en el caso KM1-CS.

   En este caso, la distancia entre centroides obtenida fue 1.658188. La mitad de este valor es aproximadamente 0.82. Vemos que el vector más lejano de la clase 0 está

Documento	*Clase*	*Distancia con centroide*
http://localhost/Tesis/TestPool/DosClases/1/palm5.htm	0	0.030493
http://localhost/Tesis/TestPool/DosClases/1/palm2.htm	0	0.100502
http://localhost/Tesis/TestPool/DosClases/1/palm3.htm	0	0.030123
http://localhost/Tesis/TestPool/DosClases/1/palm4.htm	0	0.022680
http://localhost/Tesis/TestPool/DosClases/1/palm1.htm	0	0.082259
http://localhost/Tesis/TestPool/DosClases/1/palm6.htm	0	0.029869
http://localhost/Tesis/TestPool/DosClases/1/vcg04.htm	1	0.315474
http://localhost/Tesis/TestPool/DosClases/1/vcg03.htm	1	0.651717
http://localhost/Tesis/TestPool/DosClases/1/vcg05.htm	1	0.337662

Tabla 7.3: Clustering KM1-CS.

Clase	*Dispersión*
0	0.003352
1	0.212758

Tabla 7.4: Dispersión en cada clase KM1-CS.

a 0.10 unidades de su centroide y el más lejano de la clase 1 está a 0.80. Esto hace que los mismos puedan separarse correctamente.

4. KM4-SS: Esta medición fue hecha en las mismas condiciones que KM3-SS y dio los mismos resultados que aquélla (archivo km4-ss.htm).

Pruebas con CPN

1. CPN1-SS: Esta medida se hizo con dos clases y sin stop list ni stemming y 50 iteraciones y tasa de aprendizaje 0.01 (archivo cpn1ss.htm). Los resultados están en la tabla 1 (7.7).

   Los documentos fueron clasificados correctamente. Se obtuvo una distancia inter-centroides de 1.061559 (la mitad es 0.53). Vemos que las distancias de los vectores de la clase 0 a su centroide tienen un máximo de 0.58 mientras que los de la clase 1 tienen un máximo en 0.07. Que la distancia máxima de la clase 0 sea mayor a la

Documento	*Clase*	*Distancia con centroide*
http://localhost/Tesis/TestPool/DosClases/1/palm5.htm	0	0.049828
http://localhost/Tesis/TestPool/DosClases/1/palm2.htm	0	0.100768
http://localhost/Tesis/TestPool/DosClases/1/palm3.htm	0	0.027299
http://localhost/Tesis/TestPool/DosClases/1/palm4.htm	0	0.023830
http://localhost/Tesis/TestPool/DosClases/1/palm1.htm	0	0.082583
http://localhost/Tesis/TestPool/DosClases/1/palm6.htm	0	0.030752
http://localhost/Tesis/TestPool/DosClases/1/vcg04.htm	1	0.341135
http://localhost/Tesis/TestPool/DosClases/1/vcg03.htm	1	0.807999
http://localhost/Tesis/TestPool/DosClases/1/vcg05.htm	1	0.484514

Tabla 7.5: Clasificación km3-ss.

Clase	*Dispersión*
0	0.003619
1	0.334663

Tabla 7.6: Dispersión en km3-ss.

Documento	*Clase*	*Distancia con centroide*
http://localhost/Tesis/TestPool/DosClases/1/palm5.htm	1	0.045435
http://localhost/Tesis/TestPool/DosClases/1/palm2.htm	1	0.020315
http://localhost/Tesis/TestPool/DosClases/1/palm3.htm	1	0.022486
http://localhost/Tesis/TestPool/DosClases/1/palm4.htm	1	0.010850
http://localhost/Tesis/TestPool/DosClases/1/palm1.htm	1	0.079550
http://localhost/Tesis/TestPool/DosClases/1/palm6.htm	1	0.022585
http://localhost/Tesis/TestPool/DosClases/1/vcg04.htm	0	0.413533
http://localhost/Tesis/TestPool/DosClases/1/vcg03.htm	0	0.580861
http://localhost/Tesis/TestPool/DosClases/1/vcg05.htm	0	0.436000

Tabla 7.7: Clasificación CPN1-SS.

mitad de la distancia entre centroides puede deberse a un valor muy alto de la tasa de aprendizaje (lo cual genera un sistema oscilante).

2. CPN2-CS: Esta medición se hizo con stop list y stemming. Los resultados fueron exitosos ya que los documentos fueron agrupados correctamente en dos clases a través de 50 ciclos de entrenamiento (archivo cpn2cs.htm). Los resultados están en la tabla 2 (7.8). Los centroides de los dos clusters son:

   (a) Centroide 0: (0.998551 0.039146 0.000000 0.002854 0.000000 0.000000)

   (b) Centroide 1: (0.221388 0.004797 0.000000 0.000000 0.623456 0.375288)

   En este caso, la distancia entre centroides es de 1.065227. La distancia máxima a un centroide en la clase 0 es de 0.07 y en la clase 1 es de 0.56. En el caso de la clase 0, el documento más lejano está a una distancia bien lejos de la mitad de la distancia de los centroides. En el caso de la clase 1, está justo en el límite.

Documento	*Clase*	*Distancia con centroide*
http://localhost/Tesis/TestPool/DosClases/1/palm5.htm	0	0.021393
http://localhost/Tesis/TestPool/DosClases/1/palm2.htm	0	0.018987
http://localhost/Tesis/TestPool/DosClases/1/palm3.htm	0	0.026012
http://localhost/Tesis/TestPool/DosClases/1/palm4.htm	0	0.008714
http://localhost/Tesis/TestPool/DosClases/1/palm1.htm	0	0.079466
http://localhost/Tesis/TestPool/DosClases/1/palm6.htm	0	0.021377
http://localhost/Tesis/TestPool/DosClases/1/vcg04.htm	1	0.414296
http://localhost/Tesis/TestPool/DosClases/1/vcg03.htm	1	0.562671
http://localhost/Tesis/TestPool/DosClases/1/vcg05.htm	1	0.408133

Tabla 7.8: Clasificación CPN2-CS.

Documento	*Clase*	*Distancia con centroide*
http://localhost/Tesis/TestPool/DosClases/1/palm1.htm	0	0.074491
http://localhost/Tesis/TestPool/DosClases/1/palm6.htm	0	0.026325
http://localhost/Tesis/TestPool/DosClases/1/vcg04.htm	1	0.415226
http://localhost/Tesis/TestPool/DosClases/1/vcg03.htm	1	0.563214
http://localhost/Tesis/TestPool/DosClases/1/vcg05.htm	1	0.409085
http://localhost/Tesis/TestPool/DosClases/1/palm5.htm	0	0.026341
http://localhost/Tesis/TestPool/DosClases/1/palm2.htm	0	0.023927
http://localhost/Tesis/TestPool/DosClases/1/palm3.htm	0	0.030168
http://localhost/Tesis/TestPool/DosClases/1/palm4.htm	0	0.004407

Tabla 7.9: Clasificación CPN3-CS.

Documento	*Clase*	*Distancia con centroide*
http://localhost/Tesis/TestPool/DosClases/1/palm1.htm	0	0.074737
http://localhost/Tesis/TestPool/DosClases/1/palm6.htm	0	0.027251
http://localhost/Tesis/TestPool/DosClases/1/vcg04.htm	1	0.416055
http://localhost/Tesis/TestPool/DosClases/1/vcg03.htm	1	0.588726
http://localhost/Tesis/TestPool/DosClases/1/vcg05.htm	1	0.437294
http://localhost/Tesis/TestPool/DosClases/1/palm5.htm	0	0.047802
http://localhost/Tesis/TestPool/DosClases/1/palm2.htm	0	0.024937
http://localhost/Tesis/TestPool/DosClases/1/palm3.htm	0	0.027151
http://localhost/Tesis/TestPool/DosClases/1/palm4.htm	0	0.008097

Tabla 7.10: Clasificación CPN4.

3. CPN3-CS: Esta medición se realizó usando stop list y stemming, 50 ciclos de iteraciones y pero los documentos se presentaron en otro orden (archivo cpn3cs.htm). La prueba resultó exitosa con los documentos agrupados correctamente en dos clases. Los resultados están en la tabla 3 (7.9).

   En este caso, la distancia entre centroides fue de 1.063983. Por otro lado, la distancia máxima de un documento a un centroide fue de 0.56 (en el centroide 1); en el caso del centroide 0, el documento más alejado estuvo a 0.07 (que es una distancia muy inferior a la mitad de la distancia entre centroides).

4. CPN4-SS: Esta prueba se hizo sin usar stemming ni stop list pero con el orden de la prueba CPN3-CS (archivo cpn4ss.htm). Los resultados también fueron exitosos y se muestran en la tabla 4 (7.10). En este caso, la distancia entre centroides fue de 1.054391.

Pruebas con FART

La prueba fart1-cs (tabla 7.3.1 (7.11)) reveló que un umbral $\rho = 0.9$ es muy alto ya que los documentos del libro de C fueron separados en dos clases. La prueba fart2-ss fue idéntica a la anterior pero sin usar stemming y se obtuvieron los mismos resultados. Cabe acotar que en todas las pruebas se usó *aprendizaje rápido* ($\beta = 1$) y $\alpha = 0$ para evitar la superposición de clases al máximo.

Documento	*Clase*
http://localhost/Tesis/TestPool/DosClases/1/palm1.htm	0
http://localhost/Tesis/TestPool/DosClases/1/palm6.htm	0
http://localhost/Tesis/TestPool/DosClases/1/palm5.htm	0
http://localhost/Tesis/TestPool/DosClases/1/palm2.htm	0
http://localhost/Tesis/TestPool/DosClases/1/palm3.htm	0
http://localhost/Tesis/TestPool/DosClases/1/palm4.htm	0
http://localhost/Tesis/TestPool/DosClases/1/vcg04.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg03.htm	2
http://localhost/Tesis/TestPool/DosClases/1/vcg05.htm	1

Tabla 7.11: Prueba fart1-cs.

Documento	*Clase*
http://localhost/Tesis/TestPool/DosClases/1/palm1.htm	0
http://localhost/Tesis/TestPool/DosClases/1/palm6.htm	0
http://localhost/Tesis/TestPool/DosClases/1/palm5.htm	0
http://localhost/Tesis/TestPool/DosClases/1/palm2.htm	0
http://localhost/Tesis/TestPool/DosClases/1/palm3.htm	0
http://localhost/Tesis/TestPool/DosClases/1/palm4.htm	0
http://localhost/Tesis/TestPool/DosClases/1/vcg04.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg03.htm	2
http://localhost/Tesis/TestPool/DosClases/1/vcg05.htm	1

Tabla 7.12: Prueba fart3-cs.

La prueba fart3-cs usando stemming con $\rho = 0.85$ dio igual a las anteriores (tabla 7.3.1 (7.12)). Igualmente resultó sin usar stemming (fart4-ss).

Las pruebas con $\rho = 0.80$ resultaron de la siguiente manera: al no usar stemming (fart5-ss), los resultados dieron como antes; al usar stemming (fart6-cs), los documentos fueron particionados en sólo dos clusters y en forma correcta (tabla 7.3.1 (7.13)). Este resultado coincide con la intuición ya que el stemming elimina las terminaciones de las palabras eliminando de esta manera más detalle en la representación de los documentos.

Al presentar los documentos en forma mezclada, el algoritmo los separó correctamente en ambos casos (fart7-cs (tabla 7.3.1 (7.14)) y fart8-ss (7.3.1 (7.15))).

En estas pruebas se observa que el algoritmo con una única pasada separa correcta-

Documento	*Clase*
http://localhost/Tesis/TestPool/DosClases/1/palm5.htm	0
http://localhost/Tesis/TestPool/DosClases/1/palm2.htm	0
http://localhost/Tesis/TestPool/DosClases/1/palm3.htm	0
http://localhost/Tesis/TestPool/DosClases/1/palm4.htm	0
http://localhost/Tesis/TestPool/DosClases/1/palm1.htm	0
http://localhost/Tesis/TestPool/DosClases/1/palm6.htm	0
http://localhost/Tesis/TestPool/DosClases/1/vcg04.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg03.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg05.htm	1

Tabla 7.13: Prueba fart6-cs.

Documento	*Clase*
http://localhost/Tesis/TestPool/DosClases/1/vcg03.htm	0
http://localhost/Tesis/TestPool/DosClases/1/palm5.htm	1
http://localhost/Tesis/TestPool/DosClases/1/palm2.htm	1
http://localhost/Tesis/TestPool/DosClases/1/palm3.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg04.htm	0
http://localhost/Tesis/TestPool/DosClases/1/palm4.htm	1
http://localhost/Tesis/TestPool/DosClases/1/palm6.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg05.htm	0
http://localhost/Tesis/TestPool/DosClases/1/palm1.htm	1

Tabla 7.14: Clasificación FART7-CS.

Documento	*Clase*
http://localhost/Tesis/TestPool/DosClases/1/vcg03.htm	0
http://localhost/Tesis/TestPool/DosClases/1/palm5.htm	1
http://localhost/Tesis/TestPool/DosClases/1/palm2.htm	1
http://localhost/Tesis/TestPool/DosClases/1/palm3.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg04.htm	2
http://localhost/Tesis/TestPool/DosClases/1/palm4.htm	1
http://localhost/Tesis/TestPool/DosClases/1/palm6.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg05.htm	2
http://localhost/Tesis/TestPool/DosClases/1/palm1.htm	1

Tabla 7.15: Prueba fart8-ss.

mente los documentos; en el peor caso, cuando el ρ o parámetro de vigilancia es muy alto, se forman clases de más pero nunca se mezclan los documentos de distintas clases.

7.3.2 Java versus Palm Pilot

La lista de documentos usados en las siguientes pruebas corresponde a dos clases conceptuales: cinco documentos de un libro de *Java* (CH01.HTM, CH02.HTM, CH03.HTM, CH04.HTM y CH05.HTM) y cinco documentos sobre el PDA *Palm Pilot* (palm1.htm, palm2.htm, palm3.htm, palm4.htm y palm5.htm).

Las claves usadas para obtener el vector de características de los documentos fueron: *palm, pilot, java, programming, language, handheld*.

Luego, la representación de los documentos es la siguiente:

- Sin usar stop list y stemming:
  - 0 - CH05.HTM: (0.000000 0.000000 0.997527 1.000000 0.017986 0.000000)
  - 1 - CH02.HTM: (0.000000 0.000000 1.000000 0.880797 1.000000 0.000000)
  - 2 - CH03.HTM: (0.000000 0.000000 1.000000 0.047426 0.047426 0.000000)
  - 3 - CH04.HTM: (0.000000 0.000000 1.000000 0.017986 0.017986 0.000000)
  - 4 - CH01.HTM: (0.000000 0.000000 1.000000 0.047426 0.993307 0.000000)
  - 5 - palm3.htm: (0.993307 0.017986 0.000000 0.000000 0.000000 0.000000)
  - 6 - palm1.htm: (1.000000 0.119203 0.000000 0.000000 0.000000 0.000000)

	0	1	2	3	4	5	6	7	8	9
0	0.000000	0.989225	0.953032	0.982017	1.363325	1.726953	1.734818	1.731368	1.664786	1.730811
1	0.989225	0.000000	1.265664	1.307208	0.833398	1.939790	1.946796	1.943721	1.884657	1.943226
2	0.953032	1.265664	0.000000	0.041634	0.945881	1.411198	1.420812	1.416597	1.334401	1.415917
3	0.982017	1.307208	0.041634	0.000000	0.975765	1.409833	1.419456	1.415237	1.332957	1.414557
4	1.363325	0.833398	0.945881	0.975765	0.000000	1.725077	1.732951	1.729496	1.662840	1.728940
5	1.726953	1.939790	1.411198	1.409833	1.725077	0.000000	0.101438	0.030191	0.112510	0.006693
6	1.734818	1.946796	1.420812	1.419456	1.732951	0.101438	0.000000	0.071777	0.156378	0.101217
7	1.731368	1.943721	1.416597	1.415237	1.729496	0.030191	0.071777	0.000000	0.122784	0.029440
8	1.664786	1.884657	1.334401	1.332957	1.662840	0.112510	0.156378	0.122784	0.000000	0.119203
9	1.730811	1.943226	1.415917	1.414557	1.728940	0.006693	0.101217	0.029440	0.119203	0.000000

Tabla 7.16: Matriz de similitud entre documentos JP-SS.

  - 7 - palm4.htm: (1.000000 0.047426 0.000000 0.000000 0.000000 0.000000)
  - 8 - palm2.htm: (0.880797 0.017986 0.000000 0.000000 0.000000 0.000000)
  - 9 - palm5.htm: (1.000000 0.017986 0.000000 0.000000 0.000000 0.000000)

- Usando stop list y stemming:
  - 0 - CH05.HTM: (0.000000 0.000000 0.997527 1.000000 0.017986 0.000000)
  - 1 - CH02.HTM: (0.000000 0.000000 1.000000 0.999089 1.000000 0.000000)
  - 2 - CH03.HTM: (0.000000 0.000000 1.000000 0.880797 0.047426 0.000000)
  - 3 - CH04.HTM: (0.000000 0.000000 1.000000 0.017986 0.017986 0.000000)
  - 4 - CH01.HTM: (0.000000 0.000000 1.000000 0.119203 0.993307 0.000000)
  - 5 - palm3.htm: (0.993307 0.017986 0.000000 0.000000 0.000000 0.000000)
  - 6 - palm1.htm: (1.000000 0.119203 0.000000 0.000000 0.000000 0.000000)
  - 7 - palm4.htm: (1.000000 0.047426 0.000000 0.000000 0.000000 0.000000)
  - 8 - palm2.htm: (0.880797 0.017986 0.000000 0.000000 0.000000 0.000000)
  - 9 - palm5.htm: (1.000000 0.017986 0.000000 0.000000 0.000000 0.000000)

Matriz de Similitud entre Documentos

Las matrices de similitud entre documentos –sin usar stemming (tabla 7.3.2 (7.16)) y usando stemming (tabla 7.3.2 (7.17))– muestran que, nuevamente, los documentos de la serie *Palm* están cerca entre sí y están lejos de los de la serie *Java* y viceversa. Esto implica que los algoritmos de clustering tienen que ser efectivos con esta colección de documentos.

Pruebas con K-Medias

Las pruebas realizadas con el algoritmo de las K-medias en sus dos variantes dieron en forma exitosa con una separación de los documentos en dos clusters.

En el caso de la medición sin usar usar filtrado de stop word ni stemming están expresados en las tablas 7.3.2 (7.18) y 7.3.2 (7.19); además, la distancia entre los centroides de los clusters fue de 1.510869. Nuevamente, se aprecia que la distancia entre los centroides es mucho mayor a la dispersión en los clusters; indicando, de esta manera, que los clusters

	0	1	2	3	4	5	6	7	8	9
0	0.000000	0.982017	0.122809	0.982017	1.314177	1.726953	1.734818	1.731368	1.664786	1.730811
1	0.982017	0.000000	0.959891	1.388133	0.879911	1.996287	2.003095	2.000107	1.942757	1.999626
2	0.122809	0.959891	0.000000	0.863313	1.214379	1.662840	1.671006	1.667424	1.598180	1.666846
3	0.982017	1.388133	0.863313	0.000000	0.980559	1.409833	1.419456	1.415237	1.332957	1.414557
4	1.314177	0.879911	1.214379	0.980559	0.000000	1.728540	1.736398	1.732951	1.666432	1.732395
5	1.726953	1.996287	1.662840	1.409833	1.728540	0.000000	0.101438	0.030191	0.112510	0.006693
6	1.734818	2.003095	1.671006	1.419456	1.736398	0.101438	0.000000	0.071777	0.156378	0.101217
7	1.731368	2.000107	1.667424	1.415237	1.732951	0.030191	0.071777	0.000000	0.122784	0.029440
8	1.664786	1.942757	1.598180	1.332957	1.666432	0.112510	0.156378	0.122784	0.000000	0.119203
9	1.730811	1.999626	1.666846	1.414557	1.732395	0.006693	0.101217	0.029440	0.119203	0.000000

Tabla 7.17: Matriz de similitud JP-CS.

Documento	*Clase*	*Distancia con centroide*
http://localhost/test/paginas/CH05.HTM	0	0.720711
http://localhost/test/paginas/CH02.HTM	0	0.757772
http://localhost/test/paginas/CH03.HTM	0	0.508699
http://localhost/test/paginas/CH04.HTM	0	0.550322
http://localhost/test/paginas/CH01.HTM	0	0.676356
http://localhost/test/paginas/palm3.htm	1	0.032009
http://localhost/test/paginas/palm1.htm	1	0.079195
http://localhost/test/paginas/palm4.htm	1	0.025396
http://localhost/test/paginas/palm2.htm	1	0.097588
http://localhost/test/paginas/palm5.htm	1	0.036288

Tabla 7.18: Clasificación KM1-JP-SS.

tienen forma circular o, al menos, pueden ser circundados por un círculo. Esta medición completa está en el archivo KM1-JP-SS.htm.

Por otro lado, los resultados de la prueba con los mismos documentos pero esta vez usando filtrado de stop words y stemming en los términos restantes están en las tablas 7.3.2 (7.20) y 7.3.2 (7.21); por otro lado, la distancia entre centroides fue de 1.577292. Lo expresado en el caso anterior también es válido en este caso. Esta medición completa está en el archivo KM2-JP-CS.htm.

Pruebas con CPN

Las pruebas con la red de CPN se hicieron también con la variable uso de stop list y stemming. La tasa de aprendizaje usada en este caso también fue $\alpha = 0.01$, la cantidad de pasos de entrenamiento fue 50 y la cantidad de clases o clusters esperados fue 2.

- CPN1: Los resultados fueron nuevamente exitosos. Para el caso sin stemming los centroides de las clases obtenidos fueron:

Clase	*Dispersión*
0	0.422546
1	0.003756

Tabla 7.19: Dispersión en clases KM1-JP-SS.

Documento	*Clase*	*Distancia con centroide*
http://localhost/test/paginas/CH05.HTM	1	0.561404
http://localhost/test/paginas/CH02.HTM	1	0.705963
http://localhost/test/paginas/CH03.HTM	1	0.460763
http://localhost/test/paginas/CH04.HTM	1	0.707544
http://localhost/test/paginas/CH01.HTM	1	0.753994
http://localhost/test/paginas/palm3.htm	0	0.032009
http://localhost/test/paginas/palm1.htm	0	0.079195
http://localhost/test/paginas/palm4.htm	0	0.025396
http://localhost/test/paginas/palm2.htm	0	0.097588
http://localhost/test/paginas/palm5.htm	0	0.036288

Tabla 7.20: Clasificación KM2-JP-CS.

Clase	*Dispersión*
0	0.003756
1	0.418997

Tabla 7.21: Dispersión KM2-JP-CS.

Centroide 0: (0.896749 0.039749 0.083543 0.044909 0.002061 0.000000)

Centroide 1: (0.000000 0.000000 0.765758 0.303718 0.343567 0.000000); el resto de los resultados están en la tabla 7.3.2 (7.22). La bitácora completa de esta prueba está en el archivo CPN1-JP-SS.HTM. La distancia entre los centroides fue de 1.206136.

- CPN2: Para el caso con stemming, los centroides obtenidos fueron:

  Centroide 0: (0.902494 0.040004 0.073711 0.054650 0.001617 0.000000)

  Centroide 1: (0.000000 0.000000 0.721289 0.426689 0.326592 0.000000); el resto de los resultados se hallan en la tabla 7.3.2 (7.23) y la bitácora completa de la medición está en el archivo CPN2-JP-CS.HTM. La distancia entre centroides en esta medida fue de 1.216337.

Documento	*Clase*	*Distancia con centroide*
http://localhost/test/paginas/CH05.HTM	1	0.525723
http://localhost/test/paginas/CH02.HTM	1	0.379308
http://localhost/test/paginas/CH03.HTM	1	0.455331
http://localhost/test/paginas/CH04.HTM	1	0.492311
http://localhost/test/paginas/CH01.HTM	1	0.454215
http://localhost/test/paginas/palm3.htm	0	0.141760
http://localhost/test/paginas/palm1.htm	0	0.156331
http://localhost/test/paginas/palm4.htm	0	0.139602
http://localhost/test/paginas/palm2.htm	0	0.141393
http://localhost/test/paginas/palm5.htm	0	0.141780

Tabla 7.22: Clasificación CPN1-JP-SS.

Documento	*Clase*	*Distancia con centroide*
http://localhost/test/paginas/CH05.HTM	1	0.421698
http://localhost/test/paginas/CH02.HTM	1	0.325928
http://localhost/test/paginas/CH03.HTM	1	0.374441
http://localhost/test/paginas/CH04.HTM	1	0.582909
http://localhost/test/paginas/CH01.HTM	1	0.508482
http://localhost/test/paginas/palm3.htm	0	0.135564
http://localhost/test/paginas/palm1.htm	0	0.150827
http://localhost/test/paginas/palm4.htm	0	0.133291
http://localhost/test/paginas/palm2.htm	0	0.135178
http://localhost/test/paginas/palm5.htm	0	0.135585

Tabla 7.23: Clasificación CPN2-JP-CS.

Pruebas con FART

En el caso de la FART se realizaron varias pruebas primero presentando los documentos en orden favorable al algoritmo para ajustar el valor óptimo del parámetro de vigilancia ρ; además, se usó *aprendizaje rápido* ($beta = 1$) y $\alpha = 0$. Una vez que se determinó cuál era el valor óptimo de ρ, se hicieron pruebas con los documentos presentados en forma arbitraria.

Se puede arguir que los documentos siempre son separados correctamente por el algoritmo, aunque cuando el parámetro de vigilancia es demasiado alto, la red genera subclases dentro de las clases esperadas.

Las pruebas fueron las siguientes (donde se especifican los parámetros usados, los resultados y el archivo con la bitácora completa de las pruebas):

1. fart1-jp-ss.htm: $\rho = 0.9$ sin stemming. Los resultados fueron satisfactorios aunque hay clases de más (tabla 7.3.2 (7.24)).

2. fart2-jp-cs.htm: $\rho = 0.9$ con stemming. Los resultados también fueron satisfactorios aunque también aparecieron clases de más (tabla 7.3.2 (7. 25)).

3. fart3-jp-ss.htm: $\rho = 0.8$ sin stemming. Los resultados también dieron bien, el número de clases disminuyó pero sigue habiendo clases de más (tabla 7.3.2 (7.26)).

4. fart4-jp-cs.htm: $\rho = 0.8$ con stemming. Ver tabla 7.3.2 (7.27). Los resultados si bien no son idénticos al caso anterior (fart3-jp-ss), son similares.

5. fart5-jp-ss.htm: $\rho = 0.7$ sin stemming. Ver tabla 7.3.2 (7.28). Los resultados son similares al caso fart4-jp-cs.

6. fart6-jp-cs.htm: $\rho = 0.7$ sin stemming. Ver tabla 7.3.2 (7.29). Los resultados son similares.

7. fart7-jp-ss.htm: $\rho = 0.6$ sin stemming. Ver tabla 7.3.2 (7.30). En este caso, los documentos fueron separados correctamente en dos clases.

8. fart8-jp-cs.htm: $\rho = 0.6$ con stemming. Ver tabla 7.3.2 (7.31). También en este caso, los documentos fueron separados correctamente en dos clases.

Documento	*Clase*
http://localhost/test/paginas/CH05.HTM	0
http://localhost/test/paginas/CH02.HTM	1
http://localhost/test/paginas/CH03.HTM	2
http://localhost/test/paginas/CH04.HTM	2
http://localhost/test/paginas/CH01.HTM	3
http://localhost/test/paginas/palm3.htm	4
http://localhost/test/paginas/palm1.htm	4
http://localhost/test/paginas/palm4.htm	4
http://localhost/test/paginas/palm2.htm	4
http://localhost/test/paginas/palm5.htm	4

Tabla 7.24: Clasificación FART1-JP-SS.

Documento	*Clase*
http://localhost/test/paginas/CH05.HTM	0
http://localhost/test/paginas/CH02.HTM	1
http://localhost/test/paginas/CH03.HTM	0
http://localhost/test/paginas/CH04.HTM	2
http://localhost/test/paginas/CH01.HTM	3
http://localhost/test/paginas/palm3.htm	4
http://localhost/test/paginas/palm1.htm	4
http://localhost/test/paginas/palm4.htm	4
http://localhost/test/paginas/palm2.htm	4
http://localhost/test/paginas/palm5.htm	4

Tabla 7.25: Clasificación FART2-JP-CS.

9. fart9-jp-ss.htm: $\rho = 0.6$ sin stemming. Ver tabla 7.3.2 (7.32). Los documentos presentados a la red neuronal en forma arbitraria fueron correctamente clasificados.

10. fart10-jp-cs.htm: $\rho = 0.6$ con stemming. Ver tabla 7.3.2 (7.33). En este caso también, los documentos presentados en forma arbitraria fueron separados en dos clases en forma correcta.

7.3.3 Relojes Breitling versus Java

Las siguientes pruebas también se realizaron con dos clases: *Relojes Breitling* contra *Lenguaje Java*. Los documentos usados fueron:

- 0 - Breitling Produits.htm
- 1 - Breitling Produits2.htm
- 2 - Breitling Produits3.htm
- 3 - Breitling Produits4.htm
- 4 - Breitling Produits5.htm
- 5 - CH01.HTM

Documento	*Clase*
http://localhost/test/paginas/CH05.HTM	0
http://localhost/test/paginas/CH02.HTM	0
http://localhost/test/paginas/CH03.HTM	1
http://localhost/test/paginas/CH04.HTM	1
http://localhost/test/paginas/CH01.HTM	1
http://localhost/test/paginas/palm3.htm	2
http://localhost/test/paginas/palm1.htm	2
http://localhost/test/paginas/palm4.htm	2
http://localhost/test/paginas/palm2.htm	2
http://localhost/test/paginas/palm5.htm	2

Tabla 7.26: Clasificación FART3-JP-SS.

Documento	*Clase*
http://localhost/test/paginas/CH05.HTM	0
http://localhost/test/paginas/CH02.HTM	0
http://localhost/test/paginas/CH03.HTM	0
http://localhost/test/paginas/CH04.HTM	1
http://localhost/test/paginas/CH01.HTM	1
http://localhost/test/paginas/palm3.htm	2
http://localhost/test/paginas/palm1.htm	2
http://localhost/test/paginas/palm4.htm	2
http://localhost/test/paginas/palm2.htm	2
http://localhost/test/paginas/palm5.htm	2

Tabla 7.27: Clasificación FART4-JS-CS.

Documento	*Clase*
http://localhost/test/paginas/CH05.HTM	0
http://localhost/test/paginas/CH02.HTM	0
http://localhost/test/paginas/CH03.HTM	1
http://localhost/test/paginas/CH04.HTM	1
http://localhost/test/paginas/CH01.HTM	1
http://localhost/test/paginas/palm3.htm	2
http://localhost/test/paginas/palm1.htm	2
http://localhost/test/paginas/palm4.htm	2
http://localhost/test/paginas/palm2.htm	2
http://localhost/test/paginas/palm5.htm	2

Tabla 7.28: Clasificación FART5-JP-SS.

Documento	*Clase*
http://localhost/test/paginas/CH05.HTM	0
http://localhost/test/paginas/CH02.HTM	0
http://localhost/test/paginas/CH03.HTM	0
http://localhost/test/paginas/CH04.HTM	1
http://localhost/test/paginas/CH01.HTM	1
http://localhost/test/paginas/palm3.htm	2
http://localhost/test/paginas/palm1.htm	2
http://localhost/test/paginas/palm4.htm	2
http://localhost/test/paginas/palm2.htm	2
http://localhost/test/paginas/palm5.htm	2

Tabla 7.29: Clasificación FART6-JP-CS.

Documento	*Clase*
http://localhost/test/paginas/CH05.HTM	0
http://localhost/test/paginas/CH02.HTM	0
http://localhost/test/paginas/CH03.HTM	0
http://localhost/test/paginas/CH04.HTM	0
http://localhost/test/paginas/CH01.HTM	0
http://localhost/test/paginas/palm3.htm	1
http://localhost/test/paginas/palm1.htm	1
http://localhost/test/paginas/palm4.htm	1
http://localhost/test/paginas/palm2.htm	1
http://localhost/test/paginas/palm5.htm	1

Tabla 7.30: Clasificación FART7-JP-SS.

Documento	*Clase*
http://localhost/test/paginas/CH05.HTM	0
http://localhost/test/paginas/CH02.HTM	0
http://localhost/test/paginas/CH03.HTM	0
http://localhost/test/paginas/CH04.HTM	0
http://localhost/test/paginas/CH01.HTM	0
http://localhost/test/paginas/palm3.htm	1
http://localhost/test/paginas/palm1.htm	1
http://localhost/test/paginas/palm4.htm	1
http://localhost/test/paginas/palm2.htm	1
http://localhost/test/paginas/palm5.htm	1

Tabla 7.31: Clasificación FART8-JP-CS.

Documento	*Clase*
http://localhost/test/paginas/palm1.htm	0
http://localhost/test/paginas/CH05.HTM	1
http://localhost/test/paginas/palm2.htm	0
http://localhost/test/paginas/CH02.HTM	1
http://localhost/test/paginas/palm4.htm	0
http://localhost/test/paginas/CH03.HTM	1
http://localhost/test/paginas/palm5.htm	0
http://localhost/test/paginas/CH04.HTM	1
http://localhost/test/paginas/CH01.HTM	1
http://localhost/test/paginas/palm3.htm	0

Tabla 7.32: Clasificación FART9-JP-SS.

Documento	*Clase*
http://localhost/test/paginas/palm1.htm	0
http://localhost/test/paginas/CH05.HTM	1
http://localhost/test/paginas/palm2.htm	0
http://localhost/test/paginas/CH02.HTM	1
http://localhost/test/paginas/palm4.htm	0
http://localhost/test/paginas/CH03.HTM	1
http://localhost/test/paginas/palm5.htm	0
http://localhost/test/paginas/CH04.HTM	1
http://localhost/test/paginas/CH01.HTM	1
http://localhost/test/paginas/palm3.htm	0

Tabla 7.33: Clasificación FART10-JP-CS.

- 6 - CH02.HTM
- 7 - CH03.HTM
- 8 - CH04.HTM
- 9 - CH05.HTM

Matriz de Similitud entre Documentos

Las claves de filtrado usadas fueron: *watch, Breitling, java, programming, language.*

Los vectores de características de los documentos son los siguientes:

- Sin stemming ni stop list:
  - 0 - Breitling Produits.htm: (0.000000 1.000000 0.000000 0.000000 0.000000)
  - 1 - Breitling Produits2.htm: (0.000000 1.000000 0.000000 0.000000 0.000000)
  - 2 - Breitling Produits3.htm: (0.000000 1.000000 0.000000 0.000000 0.000000)
  - 3 - Breitling Produits4.htm: (0.000000 1.000000 0.000000 0.000000 0.000000)
  - 4 - Breitling Produits5.htm: (0.017986 1.000000 0.000000 0.000000 0.000000)
  - 5 - CH01.HTM: (0.000000 0.000000 1.000000 0.047426 0.993307)
  - 6 - CH02.HTM: (0.000000 0.000000 1.000000 0.880797 1.000000)
  - 7 - CH03.HTM: (0.000000 0.000000 1.000000 0.047426 0.047426)
  - 8 - CH04.HTM: (0.000000 0.000000 1.000000 0.017986 0.017986)
  - 9 - CH05.HTM: (0.000000 0.000000 0.997527 1.000000 0.017986)
- Con stemming y stop list:
  - 0 - Breitling Produits.htm: (0.000000 1.000000 0.000000 0.000000 0.000000)
  - 1 - Breitling Produits2.htm: (0.000000 1.000000 0.000000 0.000000 0.000000)
  - 2 - Breitling Produits3.htm: (0.000000 1.000000 0.000000 0.000000 0.000000)
  - 3 - Breitling Produits4.htm: (0.000000 1.000000 0.000000 0.000000 0.000000)
  - 4 - Breitling Produits5.htm: (0.017986 1.000000 0.000000 0.000000 0.000000)
  - 5 - CH01.HTM: (0.000000 0.000000 1.000000 0.119203 0.993307)
  - 6 - CH02.HTM: (0.017986 0.000000 1.000000 0.999089 1.000000)
  - 7 - CH03.HTM: (0.000000 0.000000 1.000000 0.880797 0.047426)
  - 8 - CH04.HTM: (0.000000 0.000000 1.000000 0.017986 0.017986)
  - 9 - CH05.HTM: (0.000000 0.000000 0.997527 1.000000 0.017986)

Las matrices de similitud entre los documentos están en las tablas 7.3.3 (7.34) y 7.3.3 (7.35).

Nuevamente, en el caso de estos dos conjuntos de documentos, se observa que la distancia entre documentos en un mismo subconjunto temático es mucho menor a la distancia entre documentos de subconjuntos distintos. Así, esta cualidad del conjunto de datos es lo que permite que se pueda clasificar a los mismos con un algoritmo de *machine learning*.

En este caso, como en los anteriores casos tratados con esta técnica, se realizaron pruebas con y sin stemming.

	0	1	2	3	4	5	6	7	8	9
0	0.000000	0.000000	0.000000	0.000000	0.017986	1.728846	1.943143	1.415803	1.414442	1.730718
1	0.000000	0.000000	0.000000	0.000000	0.017986	1.728846	1.943143	1.415803	1.414442	1.730718
2	0.000000	0.000000	0.000000	0.000000	0.017986	1.728846	1.943143	1.415803	1.414442	1.730718
3	0.000000	0.000000	0.000000	0.000000	0.017986	1.728846	1.943143	1.415803	1.414442	1.730718
4	0.017986	0.017986	0.017986	0.017986	0.000000	1.728940	1.943226	1.415917	1.414557	1.730811
5	1.728846	1.728846	1.728846	1.728846	1.728940	0.000000	0.833398	0.945881	0.975765	1.363325
6	1.943143	1.943143	1.943143	1.943143	1.943226	0.833398	0.000000	1.265664	1.307208	0.989225
7	1.415803	1.415803	1.415803	1.415803	1.415917	0.945881	1.265664	0.000000	0.041634	0.953032
8	1.414442	1.414442	1.414442	1.414442	1.414557	0.975765	1.307208	0.041634	0.000000	0.982017
9	1.730718	1.730718	1.730718	1.730718	1.730811	1.363325	0.989225	0.953032	0.982017	0.000000

Tabla 7.34: Matriz de Similitud Breitling vs Java sin stemming (MSIM-BJ-SS).

	0	1	2	3	4	5	6	7	8	9
0	0.000000	0.000000	0.000000	0.000000	0.017986	1.732301	1.999626	1.666749	1.414442	1.730718
1	0.000000	0.000000	0.000000	0.000000	0.017986	1.732301	1.999626	1.666749	1.414442	1.730718
2	0.000000	0.000000	0.000000	0.000000	0.017986	1.732301	1.999626	1.666749	1.414442	1.730718
3	0.000000	0.000000	0.000000	0.000000	0.017986	1.732301	1.999626	1.666749	1.414442	1.730718
4	0.017986	0.017986	0.017986	0.017986	0.000000	1.732395	1.999545	1.666846	1.414557	1.730811
5	1.732301	1.732301	1.732301	1.732301	1.732395	0.000000	0.880095	1.214379	0.980559	1.314177
6	1.999626	1.999626	1.999626	1.999626	1.999545	0.880095	0.000000	0.960059	1.388250	0.982182
7	1.666749	1.666749	1.666749	1.666749	1.666846	1.214379	0.960059	0.000000	0.863313	0.122809
8	1.414442	1.414442	1.414442	1.414442	1.414557	0.980559	1.388250	0.863313	0.000000	0.982017
9	1.730718	1.730718	1.730718	1.730718	1.730811	1.314177	0.982182	0.122809	0.982017	0.000000

Tabla 7.35: Matriz de Similitud Breitling vs Java con stemming (MSIM-BJ-CS).

Pruebas con K-Medias

En el caso del algoritmo de las k-medias las pruebas dieron exitosas, con los documentos separados en dos clases.

1. KM1-BJ-SS: Los resultados de las mediciones realizadas sin stemming están en las tablas 7.3.3 (7.36) y 7.3.3 (7.37) (la bitácora completa se halla en el archivo KM1-BJ-SS.HTM).

   En el caso de la prueba sin stemming los centroides representantes de las clases fueron: Centroide 0 = (0.000000 0.000000 0.999505 0.398727 0.415341) y Centroide 1 = (0.003597 1.000000 0.000000 0.000000 0.000000). Por otro lado, en el caso de la prueba con stemming los centroides fueron: Centroide 0 = (0.003597 0.000000 0.999505 0.603415 0.415341) y Centroide 1 = (0.003597 1.000000 0.000000 0.000000 0.000000).

   Además, la distancia entre centroides fue 1.526603.

2. KM2-BJ-CS: Por otro lado, los resultados de las mediciones realizadas con stemming está e las tablas 7.3.3 (7.38) y 7.3.3 (7.39) (la bitácora completa de la medición se halla en el archivo KM2-BJ-CS.HTM).

   En este caso, la distancia entre centroides fue 1.592366.

Documento	*Clase*	*Distancia con centroide*
http://localhost/tesis/testpool/dosclases/2/Breitling Produits.htm	1	0.003597
http://localhost/tesis/testpool/dosclases/2/Breitling Produits2.htm	1	0.003597
http://localhost/tesis/testpool/dosclases/2/Breitling Produits3.htm	1	0.003597
http://localhost/tesis/testpool/dosclases/2/Breitling Produits4.htm	1	0.003597
http://localhost/tesis/testpool/dosclases/2/Breitling Produits5.htm	1	0.014389
http://localhost/test/paginas/CH01.HTM	0	0.676356
http://localhost/test/paginas/CH02.HTM	0	0.757772
http://localhost/test/paginas/CH03.HTM	0	0.508699
http://localhost/test/paginas/CH04.HTM	0	0.550322
http://localhost/test/paginas/CH05.HTM	0	0.720711

Tabla 7.36: Clasificación KM1-BJ-SS.

Clase	*Dispersión*
0	0.422546
1	0.000052

Tabla 7.37: Dispersión en clases KM1-BJ-SS.

Documento	*Clase*	*Distancia con centroide*
http://localhost/tesis/testpool/dosclases/2/Breitling Produits.htm	1	0.003597
http://localhost/tesis/testpool/dosclases/2/Breitling Produits2.htm	1	0.003597
http://localhost/tesis/testpool/dosclases/2/Breitling Produits3.htm	1	0.003597
http://localhost/tesis/testpool/dosclases/2/Breitling Produits4.htm	1	0.003597
http://localhost/tesis/testpool/dosclases/2/Breitling Produits5.htm	1	0.014389
http://localhost/test/paginas/CH01.HTM	0	0.754002
http://localhost/test/paginas/CH02.HTM	0	0.706110
http://localhost/test/paginas/CH03.HTM	0	0.460777
http://localhost/test/paginas/CH04.HTM	0	0.707553
http://localhost/test/paginas/CH05.HTM	0	0.561416

Tabla 7.38: Clasificación KM2-BJ-CS.

Clase	*Dispersión*
0	0.419049
1	0.000052

Tabla 7.39: Dispersión en clases en KM2-BJ-CS.

Documento	*Clase*	*Distancia con centroide*
http://localhost/tesis/testpool/dosclases/2/Breitling Produits.htm	0	0.003357
http://localhost/tesis/testpool/dosclases/2/Breitling Produits2.htm	0	0.003357
http://localhost/tesis/testpool/dosclases/2/Breitling Produits3.htm	0	0.003357
http://localhost/tesis/testpool/dosclases/2/Breitling Produits4.htm	0	0.003357
http://localhost/tesis/testpool/dosclases/2/Breitling Produits5.htm	0	0.014627
http://localhost/test/paginas/CH01.HTM	1	0.508855
http://localhost/test/paginas/CH02.HTM	1	0.476727
http://localhost/test/paginas/CH03.HTM	1	0.393786
http://localhost/test/paginas/CH04.HTM	1	0.425942
http://localhost/test/paginas/CH05.HTM	1	0.525895

Tabla 7.40: Clasificación CPN1-BJ-SS.

En estas mediciones se advierte también que la distancia del documento más lejano de un cluster es menor a la distancia media entre centroides. Nuevamente, esta cualidad de los datos es lo que permite separar los documentos en forma exitosa.

Pruebas con CPN

En el caso de la red de contrapropagación los resultados de las pruebas realizadas también dieron exitosos. Se utilizaron dos clases, velocidad de aprendizaje igual a 0.01.

1. CPN1-BJ-SS: Los resultados de la prueba sin stemming están en el archivo CPN1-BJ-SS.HTM y en la tabla 7.3.3 (7.40). Los centroides obtenidos en el caso sin stemming fueron: Centroide 0 = (0.003357 0.999970 0.000000 0.000000 0.000000) y Centroide 1 = (0.000015 0.081058 0.738243 0.247008 0.250478). La distancia entre los centroides de las clases fue de 1.230108.

2. CPN2-BJ-CS: Por otro lado, los resultados de las pruebas con stemming están en el archivo CPN2-BJ-CS.HTM y en la tabla 7.3.3 (7.41). Los centroides obtenidos en la prueba con stemming fueron: Centroide 0 = (0.003357 0.999970 0.000000 0.000000 0.000000) y Centroide 1: (0.001904 0.081058 0.688187 0.377610 0.243805). La distancia entre los centroides fue de 1.232896.

En los dos casos los resultados dieron en forma exitosa ya que los documentos fueron separados correctamente. Nuevamente, la distancia del documento más lejano de cualquier cluster es menor a la mitad de la distancia entre centroides; por ello, el algoritmo es capaz de separar los documentos correctamente.

Pruebas con FART

En el caso de la red de la FART, una vez más demostró que es capaz de clasificar a los documentos presentados una única vez y en cualquier orden. Como siempre, un valor muy alto del parámetro de vigilancia ρ hace que aparezcan clases de más; sin embargo, los documentos de clases distintas nunca son mezclados.

Los resultados de las pruebas sin y con stemming se hallan a continuación indicando la tabla son se detallan y el archivo correspondiente donde están grabadas las bitácoras de los experimentos. Todas las pruebas se hicieron con $\alpha = 1$, y aprendizaje rápido ($\beta = 1.0$).

Documento	*Clase*	*Distancia con centroide*
http://localhost/tesis/testpool/dosclases/2/Breitling Produits.htm	0	0.003357
http://localhost/tesis/testpool/dosclases/2/Breitling Produits2.htm	0	0.003357
http://localhost/tesis/testpool/dosclases/2/Breitling Produits3.htm	0	0.003357
http://localhost/tesis/testpool/dosclases/2/Breitling Produits4.htm	0	0.003357
http://localhost/tesis/testpool/dosclases/2/Breitling Produits5.htm	0	0.014627
http://localhost/test/paginas/CH01.HTM	1	0.550563
http://localhost/test/paginas/CH02.HTM	1	0.412296
http://localhost/test/paginas/CH03.HTM	1	0.365793
http://localhost/test/paginas/CH04.HTM	1	0.532851
http://localhost/test/paginas/CH05.HTM	1	0.411582

Tabla 7.41: Clasificación CPN2-BJ-CS.

1. FART1-BJ-09-SS: $\rho = 0.9$, sin stemming, archivo fart1-bj-09-ss.htm, tabla 7.3.3 (7.42). En el caso de esta medición se puede observar que los documentos de la serie *Breitling* fueron agrupados en una misma clase; sin embargo, los documentos de la serie *Java* fueron particionados en varias subclases. A pesar de esto, los documentos de clases de contenido distinta no fueron mezclados, lo que es correcto.

2. FART2-BJ-09-CS: $\rho = 0.90$, sin stemming, archivo fart2-bj-09-cs.htm, tabla 7.3.3 (7.43). En este caso ocurrió lo mismo que en el caso anterior.

3. FART3-BJ-08-SS: $\alpha = 0.80$, sin stemming, archivo fart3-bj-08-ss.htm, tabla 7.3.3 (7.44). Nuevamente, ocurrió lo mismo que en los casos anteriores.

4. FART4-BJ-08-CS: $\alpha = 0.80$, sin stemming, archivo fart4-bj-08-cs.htm, tabla 7.3.3 (7.45). También, en este caso, la separación de los documentos Java fue excesiva.

5. FART5-BJ-07-SS: $\alpha = 0.70$, sin stemming, archivo fart5-bj-07-ss.htm, tabla 7.3.3 (7.46). Esta vez, ocurrió lo mismo.

6. FART6-BJ-07-CS: $\alpha = 0.70$, sin stemming, archivo fart6-bj-07-cs.htm, tabla 7.3.3 (7.47). También en este caso, los documentos de la serie Java fueron sobre separados.

7. FART7-BJ-06-SS: $\alpha = 0.60$, sin stemming, archivo fart7-bj-06-ss.htm, tabla 7.3.3 (7.48). Finalmente, con este nivel de vigilancia, los documentos pudieron separarse en sólo dos clases.

8. FART8-BJ-06-CS: $\alpha = 0.60$, sin stemming, archivo fart8-bj-06-cs.htm, tabla 7.3.3 (7.49). En este caso, también los documentos fueron separados en dos clases.

9. FART9-BJ-06-SS: $\alpha = 0.60$, sin stemming, archivo fart9-bj-06-ss.htm, tabla 7.3.3 (7.50). Los documentos presentados en forma arbitraria fueron clasificados correctamente.

10. FART10-BJ-06-CS: $\alpha = 0.60$, sin stemming, archivo fart10-bj-06-cs.htm, tabla 7.3.3 (7.51). Nuevamente, los documentos presentados en forma arbitraria fueron clasificados correctamente.

Documento	*Clase*
http://localhost/tesis/testpool/dosclases/2/Breitling Produits.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits2.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits3.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits4.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits5.htm	0
http://localhost/test/paginas/CH01.HTM	1
http://localhost/test/paginas/CH02.HTM	2
http://localhost/test/paginas/CH03.HTM	3
http://localhost/test/paginas/CH04.HTM	3
http://localhost/test/paginas/CH05.HTM	4

Tabla 7.42: Clasificación FART1-BJ-09-SS.

Documento	*Clase*
http://localhost/tesis/testpool/dosclases/2/Breitling Produits.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits2.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits3.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits4.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits5.htm	0
http://localhost/test/paginas/CH01.HTM	1
http://localhost/test/paginas/CH02.HTM	2
http://localhost/test/paginas/CH03.HTM	3
http://localhost/test/paginas/CH04.HTM	4
http://localhost/test/paginas/CH05.HTM	3

Tabla 7.43: Clasificación FART2-BJ-09-CS.

Documento	*Clase*
http://localhost/tesis/testpool/dosclases/2/Breitling Produits.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits2.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits3.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits4.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits5.htm	0
http://localhost/test/paginas/CH01.HTM	1
http://localhost/test/paginas/CH02.HTM	1
http://localhost/test/paginas/CH03.HTM	2
http://localhost/test/paginas/CH04.HTM	2
http://localhost/test/paginas/CH05.HTM	3

Tabla 7.44: Clasificación FART3-BJ-08-SS.

Documento	*Clase*
http://localhost/tesis/testpool/dosclases/2/Breitling Produits.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits2.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits3.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits4.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits5.htm	0
http://localhost/test/paginas/CH01.HTM	1
http://localhost/test/paginas/CH02.HTM	1
http://localhost/test/paginas/CH03.HTM	2
http://localhost/test/paginas/CH04.HTM	2
http://localhost/test/paginas/CH05.HTM	3

Tabla 7.45: Clasificación FART4-BJ-08-CS.

Documento	*Clase*
http://localhost/tesis/testpool/dosclases/2/Breitling Produits.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits2.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits3.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits4.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits5.htm	0
http://localhost/test/paginas/CH01.HTM	1
http://localhost/test/paginas/CH02.HTM	1
http://localhost/test/paginas/CH03.HTM	2
http://localhost/test/paginas/CH04.HTM	2
http://localhost/test/paginas/CH05.HTM	2

Tabla 7.46: Clasificación FART5-BJ-07-SS.

Documento	*Clase*
http://localhost/tesis/testpool/dosclases/2/Breitling Produits.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits2.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits3.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits4.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits5.htm	0
http://localhost/test/paginas/CH01.HTM	1
http://localhost/test/paginas/CH02.HTM	1
http://localhost/test/paginas/CH03.HTM	2
http://localhost/test/paginas/CH04.HTM	2
http://localhost/test/paginas/CH05.HTM	2

Tabla 7.47: Clasificación FART6-BJ-07-CS.

Documento	*Clase*
http://localhost/tesis/testpool/dosclases/2/Breitling Produits.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits2.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits3.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits4.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits5.htm	0
http://localhost/test/paginas/CH01.HTM	1
http://localhost/test/paginas/CH02.HTM	1
http://localhost/test/paginas/CH03.HTM	1
http://localhost/test/paginas/CH04.HTM	1
http://localhost/test/paginas/CH05.HTM	1

Tabla 7.48: Clasificación FART7-BJ-06-SS.

Documento	*Clase*
http://localhost/tesis/testpool/dosclases/2/Breitling Produits.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits2.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits3.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits4.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits5.htm	0
http://localhost/test/paginas/CH01.HTM	1
http://localhost/test/paginas/CH02.HTM	1
http://localhost/test/paginas/CH03.HTM	1
http://localhost/test/paginas/CH04.HTM	1
http://localhost/test/paginas/CH05.HTM	1

Tabla 7.49: Clasificación FART8-BJ-06-CS.

Documento	*Clase*
http://localhost/tesis/testpool/dosclases/2/Breitling Produits.htm	0
http://localhost/test/paginas/CH02.HTM	1
http://localhost/tesis/testpool/dosclases/2/Breitling Produits2.htm	0
http://localhost/test/paginas/CH03.HTM	1
http://localhost/tesis/testpool/dosclases/2/Breitling Produits3.htm	0
http://localhost/test/paginas/CH04.HTM	1
http://localhost/tesis/testpool/dosclases/2/Breitling Produits4.htm	0
http://localhost/test/paginas/CH05.HTM	1
http://localhost/tesis/testpool/dosclases/2/Breitling Produits5.htm	0
http://localhost/test/paginas/CH01.HTM	1

Tabla 7.50: Clasificación FART9-BJ-06-SS.

Documento	*Clase*
http://localhost/tesis/testpool/dosclases/2/Breitling Produits.htm	0
http://localhost/test/paginas/CH02.HTM	1
http://localhost/tesis/testpool/dosclases/2/Breitling Produits2.htm	0
http://localhost/test/paginas/CH03.HTM	1
http://localhost/tesis/testpool/dosclases/2/Breitling Produits3.htm	0
http://localhost/test/paginas/CH04.HTM	1
http://localhost/tesis/testpool/dosclases/2/Breitling Produits4.htm	0
http://localhost/test/paginas/CH05.HTM	1
http://localhost/tesis/testpool/dosclases/2/Breitling Produits5.htm	0
http://localhost/test/paginas/CH01.HTM	1

Tabla 7.51: Clasificación FART10-BJ-06-CS.

7.4 Mediciones con Varias Clases de Documentos

En esta sección se muestran los resultados de las mediciones con varias clases de documentos para determinar la performance de los algoritmos de clustering y en particular de la teoría de la resonancia adaptativa difusa usando el método de claves para representar documentos.

Se utilizaron cuatro clases conceptuales de documentos:

1. *Relojes Breitling:*
   (a) http://localhost/tesis/testpool/dosclases/2/Breitling Produits.htm
   (b) http://localhost/tesis/testpool/dosclases/2/Breitling Produits2.htm
   (c) http://localhost/tesis/testpool/dosclases/2/Breitling Produits3.htm
   (d) http://localhost/tesis/testpool/dosclases/2/Breitling Produits4.htm
   (e) http://localhost/tesis/testpool/dosclases/2/Breitling Produits5.htm
   (f) http://localhost/tesis/testpool/dosclases/2/Breitling Produits6.htm
   (g) http://localhost/tesis/testpool/dosclases/2/Breitling Produits7.htm
   (h) http://localhost/tesis/testpool/dosclases/2/Breitling Produits8.htm
2. *Bases de Datos en C++:*
   (a) http://localhost/Tesis/TestPool/DosClases/1/vcg01.htm
   (b) http://localhost/Tesis/TestPool/DosClases/1/vcg02.htm
   (c) http://localhost/Tesis/TestPool/DosClases/1/vcg03.htm
   (d) http://localhost/Tesis/TestPool/DosClases/1/vcg04.htm
   (e) http://localhost/Tesis/TestPool/DosClases/1/vcg05.htm
   (f) http://localhost/Tesis/TestPool/DosClases/1/vcg06.htm
   (g) http://localhost/Tesis/TestPool/DosClases/1/vcg07.htm
   (h) http://localhost/Tesis/TestPool/DosClases/1/vcg08.htm
   (i) http://localhost/Tesis/TestPool/DosClases/1/vcg09.htm
   (j) http://localhost/Tesis/TestPool/DosClases/1/vcg10.htm

(k) http://localhost/Tesis/TestPool/DosClases/1/vcg11.htm

(l) http://localhost/Tesis/TestPool/DosClases/1/vcg12.htm

(m) http://localhost/Tesis/TestPool/DosClases/1/vcg13.htm

(n) http://localhost/Tesis/TestPool/DosClases/1/vcg14.htm

3. *Asistente Personal Palm Pilot:*

   (a) http://localhost/Tesis/TestPool/DosClases/1/palm1.htm

   (b) http://localhost/Tesis/TestPool/DosClases/1/palm2.htm

   (c) http://localhost/Tesis/TestPool/DosClases/1/palm3.htm

   (d) http://localhost/Tesis/TestPool/DosClases/1/palm4.htm

   (e) http://localhost/Tesis/TestPool/DosClases/1/palm5.htm

   (f) http://localhost/Tesis/TestPool/DosClases/1/palm6.htm

4. *Lenguaje de Programación Java:*

   (a) http://localhost/test/paginas/CH01.HTM

   (b) http://localhost/test/paginas/CH02.HTM

   (c) http://localhost/test/paginas/CH03.HTM

   (d) http://localhost/test/paginas/CH04.HTM

   (e) http://localhost/test/paginas/CH05.HTM

   (f) http://localhost/test/paginas/CH06.HTM

   (g) http://localhost/test/paginas/CH07.HTM

   (h) http://localhost/test/paginas/CH08.HTM

   (i) http://localhost/test/paginas/CH09.HTM

   (j) http://localhost/test/paginas/CH10.HTM

Las claves de filtrado que se usaron para conformar los vectores de características fueron: *watch, Breitling, java, programming, language, palm, pilot, handheld, c, database, sql, assistant, PDA, aeronautics.* Donde *watch, Breitling* y *aeronautics* sirven para representar los documentos de la serie *Breitling*; *java, programming, language* para los documentos de la serie *Java*; *c, database, sql* para la serie de *Bases de datos en C++* y *palm, pilot, handheld, assistant, PDA* para la serie *Palm Pilot.*

7.4.1 Pruebas con K-Medias

Las mediciones exitosas logradas con el algoritmo de las kmedias fueron las siguientes:

- KM6-VARIOS-OK-CS: Las claves usadas fueron: *watch, Breitling, java, programming, language, palm, pilot, handheld, c, database, sql, assistant, PDA, aeronautics.* Se usó stemming y stop list y el número de clases especificado fue 4. Los resultados están en las tablas 7.4.1 (7.52), 7.4.1 (7.53) y 7.4.1 (7.54). Los resultados dieron en forma correcta (si bien hubieron otras mediciones del k-medias que no lo hicieron).

  Centroides de las clases obtenidos:

- Centroide 0: (0.000000 0.000000 0.000000 0.000000 0.000000 0.978866 0.039762 0.000000 0.002998 0.002998 0.000000 0.002998 0.000000 0.000000)
- Centroide 1: (0.000000 0.000000 0.000000 0.853437 0.246683 0.000000 0.000000 0.000000 0.002569 0.947716 0.727472 0.016586 0.000000 0.000000)
- Centroide 2: (0.001799 0.000000 0.999740 0.614444 0.374252 0.000000 0.000000 0.000000 0.000000 0.025639 0.076703 0.003597 0.000000 0.000000)
- Centroide 3: (0.004497 1.000000 0.000000 0.000000 0.000000 0.000000 0.014105 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000)

- KM7-VARIOS-SS-OK: Para esta medición se usaron 4 clases, no se usó stop list ni stemming. Los resultados están en las tablas 7.4.1 (7.55), 7.4.1 (7.56) 7.4.1 (7.57), los cuales dieron en forma exitosa.

  Los centroides de las clases obtenidos fueron:

  - Centroide 0: (0.004497 1.000000 0.000000 0.000000 0.000000 0.000000 0.002248 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000)
  - Centroide 1: (0.000000 0.000000 0.999740 0.219624 0.374252 0.000000 0.000000 0.000000 0.886729 0.025639 0.076703 0.000000 0.000000 0.000000)
  - Centroide 2: (0.000000 0.000000 0.000000 0.428960 0.246683 0.000000 0.000000 0.000000 0.998237 0.947716 0.727472 0.000000 0.000000 0.000000)
  - Centroide 3: (0.000000 0.000000 0.000000 0.000000 0.000000 0.978866 0.039762 0.000000 0.007904 0.002998 0.000000 0.000000 0.000000 0.000000)

7.4.2 Pruebas con CPN

Las mediciones realizadas con la red de contrapropagación fueron las siguientes:

- CPN1-VARIOS-SS-OK: Se utilizaron 4 clases, velocidad de aprendizaje 0.01 e inicialización de los centroides con documentos patrones para cada una de las clases buscadas. En este caso no se usaron ni stop list ni stemming. Los resultados están en las tablas 7.4.2 (7.58) y 7.4.2 (7.59).

  Los centroides de clases obtenidos con el entrenamiento fueron:

  - Centroide 0: (0.004358 0.999941 0.000000 0.000000 0.000000 0.000000 0.002212 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000)
  - Centroide 1: (0.000000 0.000000 0.657603 0.129628 0.219541 0.000000 0.000000 0.000000 0.579233 0.014556 0.035878 0.000000 0.000000 0.000000)
  - Centroide 2: (0.000000 0.000000 0.000000 0.000000 0.000000 0.997849 0.046693 0.000000 0.007295 0.003050 0.000000 0.000000 0.000000 0.000000)
  - Centroide 3: (0.000000 0.000000 0.000000 0.234903 0.111798 0.000000 0.000000 0.000000 0.581976 0.540561 0.385632 0.000000 0.000000 0.000000)

Documento	*Clase*	*Distancia con centroide*
http://localhost/tesis/testpool/dosclases/2/Breitling Produits.htm	3	0.033623
http://localhost/tesis/testpool/dosclases/2/Breitling Produits2.htm	3	0.014804
http://localhost/tesis/testpool/dosclases/2/Breitling Produits3.htm	3	0.014804
http://localhost/tesis/testpool/dosclases/2/Breitling Produits4.htm	3	0.033623
http://localhost/tesis/testpool/dosclases/2/Breitling Produits5.htm	3	0.019517
http://localhost/tesis/testpool/dosclases/2/Breitling Produits6.htm	3	0.019517
http://localhost/tesis/testpool/dosclases/2/Breitling Produits7.htm	3	0.005940
http://localhost/tesis/testpool/dosclases/2/Breitling Produits8.htm	3	0.014804
http://localhost/test/paginas/CH01.HTM	2	0.800445
http://localhost/test/paginas/CH02.HTM	2	0.988391
http://localhost/test/paginas/CH03.HTM	2	0.429546
http://localhost/test/paginas/CH04.HTM	2	0.699460
http://localhost/test/paginas/CH05.HTM	2	0.531169
http://localhost/test/paginas/CH06.HTM	2	0.380323
http://localhost/test/paginas/CH07.HTM	2	0.640232
http://localhost/test/paginas/CH08.HTM	2	0.699460
http://localhost/test/paginas/CH09.HTM	2	0.383361
http://localhost/test/paginas/CH10.HTM	2	0.451472
http://localhost/Tesis/TestPool/DosClases/1/palm1.htm	0	0.083667
http://localhost/Tesis/TestPool/DosClases/1/palm2.htm	0	0.101658
http://localhost/Tesis/TestPool/DosClases/1/palm3.htm	0	0.030420
http://localhost/Tesis/TestPool/DosClases/1/palm4.htm	0	0.023073
http://localhost/Tesis/TestPool/DosClases/1/palm5.htm	0	0.030787
http://localhost/Tesis/TestPool/DosClases/1/palm6.htm	0	0.030169
http://localhost/Tesis/TestPool/DosClases/1/vcg01.htm	1	0.816218
http://localhost/Tesis/TestPool/DosClases/1/vcg02.htm	1	0.399137
http://localhost/Tesis/TestPool/DosClases/1/vcg03.htm	1	0.360644
http://localhost/Tesis/TestPool/DosClases/1/vcg04.htm	1	0.792183
http://localhost/Tesis/TestPool/DosClases/1/vcg05.htm	1	0.815804
http://localhost/Tesis/TestPool/DosClases/1/vcg06.htm	1	0.339141
http://localhost/Tesis/TestPool/DosClases/1/vcg07.htm	1	0.280233
http://localhost/Tesis/TestPool/DosClases/1/vcg08.htm	1	0.902756
http://localhost/Tesis/TestPool/DosClases/1/vcg09.htm	1	0.385542
http://localhost/Tesis/TestPool/DosClases/1/vcg10.htm	1	0.766812
http://localhost/Tesis/TestPool/DosClases/1/vcg11.htm	1	1.035957
http://localhost/Tesis/TestPool/DosClases/1/vcg12.htm	1	0.675216
http://localhost/Tesis/TestPool/DosClases/1/vcg13.htm	1	0.388494
http://localhost/Tesis/TestPool/DosClases/1/vcg14.htm	1	0.931647

Tabla 7.52: Clasificación KM6-VARIOS-CS-OK.

Clase	*Dispersión*
0	0.003442
1	0.467275
2	0.396810
3	0.000464

Tabla 7.53: Dispersión en clases KM6-VARIOS-CS-OK.

	0	1	2	3
0	0.000000	1.780691	1.575835	1.399601
1	1.780691	0.000000	1.531920	1.793625
2	1.575835	1.531920	0.000000	1.588662
3	1.399601	1.793625	1.588662	0.000000

Tabla 7.54: Distancias entre centroides KM6-VARIOS-CS-OK.

Documento	*Clase*	*Distancia con centroide*
http://localhost/tesis/testpool/dosclases/2/Breitling Produits.htm	0	0.005027
http://localhost/tesis/testpool/dosclases/2/Breitling Produits2.htm	0	0.005027
http://localhost/tesis/testpool/dosclases/2/Breitling Produits3.htm	0	0.005027
http://localhost/tesis/testpool/dosclases/2/Breitling Produits4.htm	0	0.005027
http://localhost/tesis/testpool/dosclases/2/Breitling Produits5.htm	0	0.013676
http://localhost/tesis/testpool/dosclases/2/Breitling Produits6.htm	0	0.013676
http://localhost/tesis/testpool/dosclases/2/Breitling Produits7.htm	0	0.016368
http://localhost/tesis/testpool/dosclases/2/Breitling Produits8.htm	0	0.005027
http://localhost/test/paginas/CH01.HTM	1	0.655301
http://localhost/test/paginas/CH02.HTM	1	1.130046
http://localhost/test/paginas/CH03.HTM	1	0.394764
http://localhost/test/paginas/CH04.HTM	1	0.432382
http://localhost/test/paginas/CH05.HTM	1	1.202866
http://localhost/test/paginas/CH06.HTM	1	0.414395
http://localhost/test/paginas/CH07.HTM	1	0.534830
http://localhost/test/paginas/CH08.HTM	1	0.439386
http://localhost/test/paginas/CH09.HTM	1	0.419922
http://localhost/test/paginas/CH10.HTM	1	0.431657
http://localhost/Tesis/TestPool/DosClases/1/palm1.htm	3	0.082637
http://localhost/Tesis/TestPool/DosClases/1/palm2.htm	3	0.101876
http://localhost/Tesis/TestPool/DosClases/1/palm3.htm	3	0.027463
http://localhost/Tesis/TestPool/DosClases/1/palm4.htm	3	0.024018
http://localhost/Tesis/TestPool/DosClases/1/palm5.htm	3	0.049918
http://localhost/Tesis/TestPool/DosClases/1/palm6.htm	3	0.030897
http://localhost/Tesis/TestPool/DosClases/1/vcg01.htm	2	0.985181
http://localhost/Tesis/TestPool/DosClases/1/vcg02.htm	2	0.553859
http://localhost/Tesis/TestPool/DosClases/1/vcg03.htm	2	0.474615
http://localhost/Tesis/TestPool/DosClases/1/vcg04.htm	2	0.842012
http://localhost/Tesis/TestPool/DosClases/1/vcg05.htm	2	0.958471
http://localhost/Tesis/TestPool/DosClases/1/vcg06.htm	2	0.429561
http://localhost/Tesis/TestPool/DosClases/1/vcg07.htm	2	0.321106
http://localhost/Tesis/TestPool/DosClases/1/vcg08.htm	2	0.534669
http://localhost/Tesis/TestPool/DosClases/1/vcg09.htm	2	0.532380
http://localhost/Tesis/TestPool/DosClases/1/vcg10.htm	2	0.769783
http://localhost/Tesis/TestPool/DosClases/1/vcg11.htm	2	1.068677
http://localhost/Tesis/TestPool/DosClases/1/vcg12.htm	2	0.841224
http://localhost/Tesis/TestPool/DosClases/1/vcg13.htm	2	0.474614
http://localhost/Tesis/TestPool/DosClases/1/vcg14.htm	2	0.819538

Tabla 7.55: Clasificación KM7-VARIOS-SS-OK.

Clase	*Dispersión*
0	0.000096
1	0.450959
2	0.523313
3	0.003664

Tabla 7.56: Dispersión en clases KM7-VARIOS-SS-OK.

	0	1	2	3
0	0.000000	1.726451	1.915396	1.399885
1	1.726451	0.000000	1.531580	1.710634
2	1.915396	1.531580	0.000000	1.899233
3	1.399885	1.710634	1.899233	0.000000

Tabla 7.57: Distancias entre centroides KM7-VARIOS-SS-OK.

- CPN2-VARIOS-CS-OK: Se utilizaron 4 clases, velocidad de aprendizaje 0.01 e inicialización de los centroides con documentos patrones para cada una de las clases buscadas. En este caso sí se usaron y stop list ni stemming. Los resultados están en las tablas 7.4.2 (7.60) y 7.4.2 (7.61).

  Los centroides obtenidos en el entrenamiento fueron:

  - Centroide 0: (0.004358 0.999632 0.000000 0.000000 0.000000 0.000000 0.015252 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000)
  - Centroide 1: (0.000905 0.000000 0.759767 0.419445 0.247915 0.000000 0.000000 0.000000 0.000000 0.016769 0.039633 0.002196 0.000000 0.000000)
  - Centroide 2: (0.000000 0.000000 0.000000 0.000000 0.000000 0.997959 0.046691 0.000000 0.002732 0.003050 0.000000 0.004242 0.000000 0.000000)
  - Centroide 3: (0.000000 0.000000 0.000000 0.534489 0.122595 0.000000 0.000000 0.000000 0.001385 0.599388 0.413697 0.011339 0.000000 0.000000)

7.4.3 Pruebas con FART

Con la red de la teoría de la resonancia adaptativa difusa se hicieron las siguientes mediciones:

- FART1-VARIOS-06-CS-MAL: Parámetros de la red: $\alpha = 0.0$, $\beta = 1.0$, $\rho = 0.60$. Se usó stemming y stop list. Archivo de log: fart1-varios-06-cs-mal.htm. El parámetro de vigilancia ρ resultó ser muy bajo, los documentos resultaron mezclados (ver tabla 7.4.3 (7.62)).

- FART2-VARIOS-06-SS-MAL: Parámetros de la red: $\alpha = 0.0$, $\beta = 1.0$, $\rho = 0.60$. No se usó stemming y stop list. Archivo de log: fart2-varios-06-ss-mal.htm. También, el parámetro de vigilancia ρ resultó ser muy bajo, los documentos resultaron mezclados (ver tabla 7.4.3 (7.63)).

Documento	*Clase*	*Distancia con centroide*
http://localhost/tesis/testpool/dosclases/2/Breitling Produits.htm	0	0.004888
http://localhost/test/paginas/CH01.HTM	1	0.388768
http://localhost/Tesis/TestPool/DosClases/1/palm1.htm	2	0.072272
http://localhost/Tesis/TestPool/DosClases/1/vcg02.htm	3	0.317844
http://localhost/tesis/testpool/dosclases/2/Breitling Produits2.htm	0	0.004888
http://localhost/tesis/testpool/dosclases/2/Breitling Produits3.htm	0	0.004888
http://localhost/tesis/testpool/dosclases/2/Breitling Produits4.htm	0	0.004888
http://localhost/tesis/testpool/dosclases/2/Breitling Produits5.htm	0	0.013804
http://localhost/tesis/testpool/dosclases/2/Breitling Produits6.htm	0	0.013804
http://localhost/tesis/testpool/dosclases/2/Breitling Produits7.htm	0	0.016363
http://localhost/tesis/testpool/dosclases/2/Breitling Produits8.htm	0	0.004888
http://localhost/test/paginas/CH02.HTM	1	0.546288
http://localhost/test/paginas/CH03.HTM	1	0.252734
http://localhost/test/paginas/CH04.HTM	1	0.276958
http://localhost/test/paginas/CH05.HTM	1	0.823603
http://localhost/test/paginas/CH06.HTM	1	0.268110
http://localhost/test/paginas/CH07.HTM	1	0.321669
http://localhost/test/paginas/CH08.HTM	1	0.281923
http://localhost/test/paginas/CH09.HTM	1	0.268444
http://localhost/test/paginas/CH10.HTM	1	0.276544
http://localhost/Tesis/TestPool/DosClases/1/palm2.htm	2	0.032378
http://localhost/Tesis/TestPool/DosClases/1/palm3.htm	2	0.029728
http://localhost/Tesis/TestPool/DosClases/1/palm4.htm	2	0.008002
http://localhost/Tesis/TestPool/DosClases/1/palm5.htm	2	0.049407
http://localhost/Tesis/TestPool/DosClases/1/palm6.htm	2	0.029829
http://localhost/Tesis/TestPool/DosClases/1/vcg01.htm	3	0.433873
http://localhost/Tesis/TestPool/DosClases/1/vcg03.htm	3	0.274665
http://localhost/Tesis/TestPool/DosClases/1/vcg04.htm	3	0.399672
http://localhost/Tesis/TestPool/DosClases/1/vcg05.htm	3	0.427462
http://localhost/Tesis/TestPool/DosClases/1/vcg06.htm	3	0.222739
http://localhost/Tesis/TestPool/DosClases/1/vcg07.htm	3	0.202915
http://localhost/Tesis/TestPool/DosClases/1/vcg08.htm	3	0.311802
http://localhost/Tesis/TestPool/DosClases/1/vcg09.htm	3	0.305945
http://localhost/Tesis/TestPool/DosClases/1/vcg10.htm	3	0.436415
http://localhost/Tesis/TestPool/DosClases/1/vcg11.htm	3	0.654766
http://localhost/Tesis/TestPool/DosClases/1/vcg12.htm	3	0.467207
http://localhost/Tesis/TestPool/DosClases/1/vcg13.htm	3	0.274664
http://localhost/Tesis/TestPool/DosClases/1/vcg14.htm	3	0.469120

Tabla 7.58: Clasificación CPN1-VARIOS-SS-OK.

	0	1	2	3
0	0.000000	1.354385	1.413380	1.359117
1	1.354385	0.000000	1.350503	0.924202
2	1.413380	1.350503	0.000000	1.354050
3	1.359117	0.924202	1.354050	0.000000

Tabla 7.59: Distancias entre centroides CPN1-VARIOS-SS-OK.

Documento	*Clase*	*Distancia con centroide*
http://localhost/tesis/testpool/dosclases/2/Breitling Produits.htm	0	0.032423
http://localhost/test/paginas/CH01.HTM	1	0.570049
http://localhost/Tesis/TestPool/DosClases/1/palm1.htm	2	0.073233
http://localhost/Tesis/TestPool/DosClases/1/vcg02.htm	3	0.210068
http://localhost/tesis/testpool/dosclases/2/Breitling Produits2.htm	0	0.015867
http://localhost/tesis/testpool/dosclases/2/Breitling Produits3.htm	0	0.015867
http://localhost/tesis/testpool/dosclases/2/Breitling Produits4.htm	0	0.032423
http://localhost/tesis/testpool/dosclases/2/Breitling Produits5.htm	0	0.020453
http://localhost/tesis/testpool/dosclases/2/Breitling Produits6.htm	0	0.020453
http://localhost/tesis/testpool/dosclases/2/Breitling Produits7.htm	0	0.005147
http://localhost/tesis/testpool/dosclases/2/Breitling Produits8.htm	0	0.015867
http://localhost/test/paginas/CH02.HTM	1	0.518207
http://localhost/test/paginas/CH03.HTM	1	0.324458
http://localhost/test/paginas/CH04.HTM	1	0.522932
http://localhost/test/paginas/CH05.HTM	1	0.378498
http://localhost/test/paginas/CH06.HTM	1	0.294513
http://localhost/test/paginas/CH07.HTM	1	0.372243
http://localhost/test/paginas/CH08.HTM	1	0.522932
http://localhost/test/paginas/CH09.HTM	1	0.305995
http://localhost/test/paginas/CH10.HTM	1	0.339008
http://localhost/Tesis/TestPool/DosClases/1/palm2.htm	2	0.031939
http://localhost/Tesis/TestPool/DosClases/1/palm3.htm	2	0.032921
http://localhost/Tesis/TestPool/DosClases/1/palm4.htm	2	0.006006
http://localhost/Tesis/TestPool/DosClases/1/palm5.htm	2	0.029367
http://localhost/Tesis/TestPool/DosClases/1/palm6.htm	2	0.029351
http://localhost/Tesis/TestPool/DosClases/1/vcg01.htm	3	0.401232
http://localhost/Tesis/TestPool/DosClases/1/vcg03.htm	3	0.217484
http://localhost/Tesis/TestPool/DosClases/1/vcg04.htm	3	0.401886
http://localhost/Tesis/TestPool/DosClases/1/vcg05.htm	3	0.401710
http://localhost/Tesis/TestPool/DosClases/1/vcg06.htm	3	0.177898
http://localhost/Tesis/TestPool/DosClases/1/vcg07.htm	3	0.183502
http://localhost/Tesis/TestPool/DosClases/1/vcg08.htm	3	0.614531
http://localhost/Tesis/TestPool/DosClases/1/vcg09.htm	3	0.214061
http://localhost/Tesis/TestPool/DosClases/1/vcg10.htm	3	0.466034
http://localhost/Tesis/TestPool/DosClases/1/vcg11.htm	3	0.698539
http://localhost/Tesis/TestPool/DosClases/1/vcg12.htm	3	0.405050
http://localhost/Tesis/TestPool/DosClases/1/vcg13.htm	3	0.204330
http://localhost/Tesis/TestPool/DosClases/1/vcg14.htm	3	0.599692

Tabla 7.60: Clasificación CPN2-VARIOS-CS-OK.

	0	1	2	3
0	0.000000	1.347593	1.412879	1.353057
1	1.347593	0.000000	1.347040	1.041941
2	1.412879	1.347040	0.000000	1.351157
3	1.353057	1.041941	1.351157	0.000000

Tabla 7.61: Distancias entre centroides CPN2-VARIOS-CS-OK.

- FART3-VARIOS-07-MAL: Parámetros de la red: $\alpha = 0.0$, $\beta = 1.0$, $\rho = 0.70$. No se usó stemming y stop list. Archivo de log: fart3-varios-07-mal.htm. También los documentos fueron mal distribuidos porque el parámetro de vigilancia estaba muy bajo (tabla 7.4.3 (7.64)).

- FART4-VARIOS-08-MAL: Idem anterior pero con $\rho = 0.80$. Archivo de log: fart4-varios-08-mal.htm. También los archivos resultan mal separados por la misma razón (tabla 7.4.3 (7.65)).

- FART5-VARIOS-09-OK: En este caso se usó parámetro de vigilancia $\rho = 0.9$. Además, no se usó eliminación de stop words ni stemming. Archivo de log: fart5-varios-09-ok.htm. Los resultados dieron en forma correcta aunque una misma clase conceptual es particionada en subclases por el algoritmo (tabla 7.4.3 (7.66)).

- FART6-VARIOS-09-OK: Idem anterior pero usando stop list y stemming. Archivo de log: fart6-varios-09-ok.htm. Ver tabla 7.4.3 (7.67).

7.5 Discusión

Hay dos puntos para discutir: 1) el efecto del cambio de representación y 2) los resultados de los métodos de clasificación.

El cambio de representación tiene varias consecuencias:

- El tamaño del vector de características se ha reducido en una cantidad muy importante. Antes, en el método de los trigramas, el vector tenía $27^3 = 19683$ componentes; ahora, la cantidad de componentes del vector de características es igual a la cantidad de claves de filtrado ingresadas por el usuario, las que generalmente no llegan a los extremos de 19000 valores. Debido a esto, la eficiencia del algoritmo se ha incrementado, ya sea en tiempo de ejecución y por la dificultad disminuida en lograr una clasificación adecuada de los datos.

- El ingreso de claves por parte del usuario ha hecho que la representación se haya vuelto más dependiente del usuario y no autocontenida (a diferencia de lo que ocurre en los trabajos relacionados, ver secciones 9.3.8 y 9.3.13).

- Además, esta representación puede verse como un caso particular de IREP de sistemas de recuperación de información tradicionales (capítulo 3).

Con respecto a los resultados de los algoritmos de clasificación, se puede decir que los mismos han dado en forma satisfactoria.

La explicación de la efectividad de los algoritmos de clustering reside en el hecho de que los documentos de un mismo cluster se encuentran cerca entre sí y a una distancia mucho menor que la distancia entre los centroides de los clusters.

El algoritmo de las k-medias funciona bien salvo en contadas ocasiones en las que mezcla documentos en clases que no están claramente separadas; tal es el caso de los documentos de *bases de datos en C++* y *Java*. Queda claro que la correctitud de la clasificación es totalmente dependiente de las claves que especifique el usuario.

Documento	*Clase*
http://localhost/tesis/testpool/dosclases/2/Breitling Produits.htm	0
http://localhost/test/paginas/CH01.HTM	0
http://localhost/Tesis/TestPool/DosClases/1/palm1.htm	0
http://localhost/Tesis/TestPool/DosClases/1/vcg02.htm	1
http://localhost/tesis/testpool/dosclases/2/Breitling Produits2.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits3.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits4.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits5.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits6.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits7.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits8.htm	0
http://localhost/test/paginas/CH02.HTM	1
http://localhost/test/paginas/CH03.HTM	0
http://localhost/test/paginas/CH04.HTM	0
http://localhost/test/paginas/CH05.HTM	0
http://localhost/test/paginas/CH06.HTM	0
http://localhost/test/paginas/CH07.HTM	0
http://localhost/test/paginas/CH08.HTM	0
http://localhost/test/paginas/CH09.HTM	0
http://localhost/test/paginas/CH10.HTM	0
http://localhost/Tesis/TestPool/DosClases/1/palm2.htm	0
http://localhost/Tesis/TestPool/DosClases/1/palm3.htm	0
http://localhost/Tesis/TestPool/DosClases/1/palm4.htm	0
http://localhost/Tesis/TestPool/DosClases/1/palm5.htm	0
http://localhost/Tesis/TestPool/DosClases/1/palm6.htm	0
http://localhost/Tesis/TestPool/DosClases/1/vcg01.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg03.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg04.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg05.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg06.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg07.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg08.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg09.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg10.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg11.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg12.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg13.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg14.htm	1

Tabla 7.62: Clasificación FART1-VARIOS-06-CS-MAL.

Documento	*Clase*
http://localhost/tesis/testpool/dosclases/2/Breitling Produits.htm	0
http://localhost/test/paginas/CH01.HTM	0
http://localhost/Tesis/TestPool/DosClases/1/palm1.htm	0
http://localhost/Tesis/TestPool/DosClases/1/vcg02.htm	1
http://localhost/tesis/testpool/dosclases/2/Breitling Produits2.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits3.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits4.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits5.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits6.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits7.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits8.htm	0
http://localhost/test/paginas/CH02.HTM	1
http://localhost/test/paginas/CH03.HTM	0
http://localhost/test/paginas/CH04.HTM	0
http://localhost/test/paginas/CH05.HTM	2
http://localhost/test/paginas/CH06.HTM	0
http://localhost/test/paginas/CH07.HTM	0
http://localhost/test/paginas/CH08.HTM	0
http://localhost/test/paginas/CH09.HTM	0
http://localhost/test/paginas/CH10.HTM	0
http://localhost/Tesis/TestPool/DosClases/1/palm2.htm	0
http://localhost/Tesis/TestPool/DosClases/1/palm3.htm	0
http://localhost/Tesis/TestPool/DosClases/1/palm4.htm	0
http://localhost/Tesis/TestPool/DosClases/1/palm5.htm	0
http://localhost/Tesis/TestPool/DosClases/1/palm6.htm	0
http://localhost/Tesis/TestPool/DosClases/1/vcg01.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg03.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg04.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg05.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg06.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg07.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg08.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg09.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg10.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg11.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg12.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg13.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg14.htm	1

Tabla 7.63: Clasificación FART2-VARIOS-06-SS-MAL.

Documento	*Clase*
http://localhost/tesis/testpool/dosclases/2/Breitling Produits.htm	0
http://localhost/test/paginas/CH01.HTM	0
http://localhost/Tesis/TestPool/DosClases/1/palm1.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg02.htm	1
http://localhost/tesis/testpool/dosclases/2/Breitling Produits2.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits3.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits4.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits5.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits6.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits7.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits8.htm	0
http://localhost/test/paginas/CH02.HTM	2
http://localhost/test/paginas/CH03.HTM	0
http://localhost/test/paginas/CH04.HTM	0
http://localhost/test/paginas/CH05.HTM	2
http://localhost/test/paginas/CH06.HTM	0
http://localhost/test/paginas/CH07.HTM	2
http://localhost/test/paginas/CH08.HTM	0
http://localhost/test/paginas/CH09.HTM	0
http://localhost/test/paginas/CH10.HTM	0
http://localhost/Tesis/TestPool/DosClases/1/palm2.htm	1
http://localhost/Tesis/TestPool/DosClases/1/palm3.htm	1
http://localhost/Tesis/TestPool/DosClases/1/palm4.htm	1
http://localhost/Tesis/TestPool/DosClases/1/palm5.htm	1
http://localhost/Tesis/TestPool/DosClases/1/palm6.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg01.htm	2
http://localhost/Tesis/TestPool/DosClases/1/vcg03.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg04.htm	2
http://localhost/Tesis/TestPool/DosClases/1/vcg05.htm	2
http://localhost/Tesis/TestPool/DosClases/1/vcg06.htm	2
http://localhost/Tesis/TestPool/DosClases/1/vcg07.htm	2
http://localhost/Tesis/TestPool/DosClases/1/vcg08.htm	3
http://localhost/Tesis/TestPool/DosClases/1/vcg09.htm	3
http://localhost/Tesis/TestPool/DosClases/1/vcg10.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg11.htm	3
http://localhost/Tesis/TestPool/DosClases/1/vcg12.htm	3
http://localhost/Tesis/TestPool/DosClases/1/vcg13.htm	1
http://localhost/Tesis/TestPool/DosClases/1/vcg14.htm	1

Tabla 7.64: Clasificación FART3-VARIOS-MAL.

Documento	*Clase*
http://localhost/tesis/testpool/dosclases/2/Breitling Produits.htm	0
http://localhost/test/paginas/CH01.HTM	1
http://localhost/Tesis/TestPool/DosClases/1/palm1.htm	0
http://localhost/Tesis/TestPool/DosClases/1/vcg02.htm	2
http://localhost/tesis/testpool/dosclases/2/Breitling Produits2.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits3.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits4.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits5.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits6.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits7.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits8.htm	0
http://localhost/test/paginas/CH02.HTM	1
http://localhost/test/paginas/CH03.HTM	1
http://localhost/test/paginas/CH04.HTM	3
http://localhost/test/paginas/CH05.HTM	1
http://localhost/test/paginas/CH06.HTM	1
http://localhost/test/paginas/CH07.HTM	1
http://localhost/test/paginas/CH08.HTM	3
http://localhost/test/paginas/CH09.HTM	1
http://localhost/test/paginas/CH10.HTM	1
http://localhost/Tesis/TestPool/DosClases/1/palm2.htm	0
http://localhost/Tesis/TestPool/DosClases/1/palm3.htm	0
http://localhost/Tesis/TestPool/DosClases/1/palm4.htm	0
http://localhost/Tesis/TestPool/DosClases/1/palm5.htm	0
http://localhost/Tesis/TestPool/DosClases/1/palm6.htm	0
http://localhost/Tesis/TestPool/DosClases/1/vcg01.htm	2
http://localhost/Tesis/TestPool/DosClases/1/vcg03.htm	2
http://localhost/Tesis/TestPool/DosClases/1/vcg04.htm	2
http://localhost/Tesis/TestPool/DosClases/1/vcg05.htm	2
http://localhost/Tesis/TestPool/DosClases/1/vcg06.htm	2
http://localhost/Tesis/TestPool/DosClases/1/vcg07.htm	2
http://localhost/Tesis/TestPool/DosClases/1/vcg08.htm	2
http://localhost/Tesis/TestPool/DosClases/1/vcg09.htm	2
http://localhost/Tesis/TestPool/DosClases/1/vcg10.htm	4
http://localhost/Tesis/TestPool/DosClases/1/vcg11.htm	4
http://localhost/Tesis/TestPool/DosClases/1/vcg12.htm	4
http://localhost/Tesis/TestPool/DosClases/1/vcg13.htm	2
http://localhost/Tesis/TestPool/DosClases/1/vcg14.htm	4

Tabla 7.65: Clasificación FART4-VARIOS-08-MAL.

Documento	*Clase*
http://localhost/tesis/testpool/dosclases/2/Breitling Produits.htm	0
http://localhost/test/paginas/CH01.HTM	1
http://localhost/Tesis/TestPool/DosClases/1/palm1.htm	2
http://localhost/Tesis/TestPool/DosClases/1/vcg02.htm	3
http://localhost/tesis/testpool/dosclases/2/Breitling Produits2.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits3.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits4.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits5.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits6.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits7.htm	0
http://localhost/tesis/testpool/dosclases/2/Breitling Produits8.htm	0
http://localhost/test/paginas/CH02.HTM	4
http://localhost/test/paginas/CH03.HTM	5
http://localhost/test/paginas/CH04.HTM	5
http://localhost/test/paginas/CH05.HTM	5
http://localhost/test/paginas/CH06.HTM	5
http://localhost/test/paginas/CH07.HTM	4
http://localhost/test/paginas/CH08.HTM	5
http://localhost/test/paginas/CH09.HTM	1
http://localhost/test/paginas/CH10.HTM	5
http://localhost/Tesis/TestPool/DosClases/1/palm2.htm	2
http://localhost/Tesis/TestPool/DosClases/1/palm3.htm	2
http://localhost/Tesis/TestPool/DosClases/1/palm4.htm	2
http://localhost/Tesis/TestPool/DosClases/1/palm5.htm	2
http://localhost/Tesis/TestPool/DosClases/1/palm6.htm	2
http://localhost/Tesis/TestPool/DosClases/1/vcg01.htm	3
http://localhost/Tesis/TestPool/DosClases/1/vcg03.htm	3
http://localhost/Tesis/TestPool/DosClases/1/vcg04.htm	3
http://localhost/Tesis/TestPool/DosClases/1/vcg05.htm	3
http://localhost/Tesis/TestPool/DosClases/1/vcg06.htm	3
http://localhost/Tesis/TestPool/DosClases/1/vcg07.htm	3
http://localhost/Tesis/TestPool/DosClases/1/vcg08.htm	6
http://localhost/Tesis/TestPool/DosClases/1/vcg09.htm	3
http://localhost/Tesis/TestPool/DosClases/1/vcg10.htm	7
http://localhost/Tesis/TestPool/DosClases/1/vcg11.htm	7
http://localhost/Tesis/TestPool/DosClases/1/vcg12.htm	7
http://localhost/Tesis/TestPool/DosClases/1/vcg13.htm	3
http://localhost/Tesis/TestPool/DosClases/1/vcg14.htm	6

Tabla 7.66: Clasificación FART5-VARIOS-OK.

Documento	*Clase*
http://localhost/test/paginas/CH04.HTM	0
http://localhost/test/paginas/CH05.HTM	0
http://localhost/test/paginas/CH06.HTM	0
http://localhost/test/paginas/CH07.HTM	1
http://localhost/tesis/testpool/dosclases/2/Breitling Produits.htm	2
http://localhost/test/paginas/CH01.HTM	1
http://localhost/Tesis/TestPool/DosClases/1/palm1.htm	3
http://localhost/Tesis/TestPool/DosClases/1/vcg02.htm	4
http://localhost/tesis/testpool/dosclases/2/Breitling Produits2.htm	2
http://localhost/tesis/testpool/dosclases/2/Breitling Produits3.htm	2
http://localhost/tesis/testpool/dosclases/2/Breitling Produits7.htm	2
http://localhost/tesis/testpool/dosclases/2/Breitling Produits8.htm	2
http://localhost/test/paginas/CH02.HTM	5
http://localhost/test/paginas/CH03.HTM	0
http://localhost/test/paginas/CH08.HTM	0
http://localhost/Tesis/TestPool/DosClases/1/vcg09.htm	4
http://localhost/Tesis/TestPool/DosClases/1/vcg10.htm	4
http://localhost/Tesis/TestPool/DosClases/1/vcg11.htm	6
http://localhost/Tesis/TestPool/DosClases/1/vcg12.htm	4
http://localhost/test/paginas/CH09.HTM	1
http://localhost/test/paginas/CH10.HTM	0
http://localhost/Tesis/TestPool/DosClases/1/palm2.htm	3
http://localhost/Tesis/TestPool/DosClases/1/palm3.htm	3
http://localhost/Tesis/TestPool/DosClases/1/palm4.htm	3
http://localhost/Tesis/TestPool/DosClases/1/palm5.htm	3
http://localhost/Tesis/TestPool/DosClases/1/palm6.htm	3
http://localhost/Tesis/TestPool/DosClases/1/vcg01.htm	7
http://localhost/Tesis/TestPool/DosClases/1/vcg03.htm	4
http://localhost/Tesis/TestPool/DosClases/1/vcg04.htm	7
http://localhost/Tesis/TestPool/DosClases/1/vcg05.htm	7
http://localhost/Tesis/TestPool/DosClases/1/vcg06.htm	4
http://localhost/Tesis/TestPool/DosClases/1/vcg07.htm	7
http://localhost/Tesis/TestPool/DosClases/1/vcg08.htm	8
http://localhost/Tesis/TestPool/DosClases/1/vcg13.htm	4
http://localhost/Tesis/TestPool/DosClases/1/vcg14.htm	8
http://localhost/tesis/testpool/dosclases/2/Breitling Produits4.htm	2
http://localhost/tesis/testpool/dosclases/2/Breitling Produits5.htm	2
http://localhost/tesis/testpool/dosclases/2/Breitling Produits6.htm	2

Tabla 7.67: Clasificación FART6-VARIOS-OK.

La red de contrapropagación trabajó bien en los casos en que se consideraron sólo dos clases de documentos. En el caso de las mediciones con varias clases de documentos, hubo que "ayudar" al algoritmo inicializando los clusters con representantes de las clases elegidos a "mano".

La red de la teoría de la resonancia adaptativa difusa mostró que puede separar un conjunto de documentos presentado únicamente una vez en forma arbitraria. La correctitud de la clasificación dependerá del acierto en la elección del parámetro de vigilancia: si éste es muy chico, se aparearán documentos no relacionados; si es muy alto, una misma clase se particionará en varias menores (pero siempre sin aparear documentos no relacionados). Por lo tanto, siempre se puede encontrar un nivel del parámetro de vigilancia ρ tal que los documentos puedan clasificarse correctamente con una única pasada sobre ellos.

Además, los documentos que no tengan relación alguna con el perfil codificado, tienen un vector de características nulo (ya que no contienen ninguna de las claves de filtrado).

7.6 Resumen

La representación de documentos mediante una fuzzyficación de la cantidad de apariciones de términos de filtrado permite la clasificación de los mismos con algoritmos de clustering tradicionales de varias pasadas como el de las k-medias y la red de contrapropagación. Las mediciones también demostraron que siempre es posible encontrar un conjunto de parámetros para la red neuronal de la teoría de la resonancia adaptativa difusa de tal manera que, en una única pasada, clasifique correctamente los documentos de entrada.

Capítulo 8

Internet

Por un lado, los agentes descriptos en capítulos posteriores tienen su lugar de acción en Internet. Por otro, el agente *Querando!* filtra páginas HTML las que obtiene de la web. Además, el agente está implementado con una interfaz basada en browser (usando HTML, CGI y guiones JavaScript). Por lo tanto, se explica qué es la WWW y cómo funcionan las aplicaciones basadas en formularios HTML y CGI. Por último, se analizan las variantes a CGI para implementar aplicaciones basadas en Web.

8.1 Historia de Internet

En esta sección, se hace un breve resumen de la historia de las redes en los Estados Unidos que llevaron al desarrollo de lo que hoy se conoce como Internet.

La carestía de las computadoras de antaño, en cuanto a espacio en almacenamiento secundario y costo pecuniario, hizo surgir las redes de computadoras como una manera de compartir recursos y de esta manera disminuir los costos asociados a su funcionamiento.

Por otro lado, en los años sesenta, el recrudecimiento de la guerra fría entre los Estados Unidos y el bloque de países orientales alineados con la entonces Unión de Repúblicas Socialistas Soviéticas planteó la necesidad de mantener un sistema de comunicaciones alternativo a las líneas telefónicas en el caso del surgimiento de una guerra termonuclear entre dichas potencias. Las redes de computadoras que usaban comunicación telefónica se consideraban muy frágiles para dicho propósito [Cheong96, Gach96].

Como siempre, la necesidad es la madre de la inventiva. Así, aparece el concepto de transporte de datos por *packet switching*, en donde los datos se envían particionados en paquetes, y la pérdida de un paquete, por falla del medio, no implicaba el reenvío de todo el mensaje entero sino sólamente del paquete perdido. Además, se pasaba de la comunicación punto a punto entre computadoras a usar una red donde los paquetes, para llegar de una máquina a otra, pasaban por las máquinas intermedias hasta llegar a destino, donde eran reensamblados [Cheong96].

Para 1970, Internet era una red formada por cuatro supercomputadoras en cuatro universidades norteamericanas comisionadas por la Agencia de Proyectos de Investigación Avanzados del Departamento de Defensa (DARPA); esta primigenia red se llamó ARPAnet. Para 1973, la ARPAnet tenía quince nodos. En esa época, ARPAnet involucraba a cuatro subredes (incluyendo una de satélites).

Hacia fines de los años 70, surgieron varias redes comunitarias: CSNET, BITNET, USENET, FidoNet, IBM VNET. Estas redes unían centros de investigación, máquinas

IBM, sitios UNIX, PC con MS-DOS via modem, respectivamente. Luego, por iniciativa de Vinton Cerf se unieron la ARPAnet y la CSNET usando protocolos TCP/IP. Lo más importante es que las empresas y los particulares empiezan a participar de la red además de los militares y los científicos.

En los ochenta aparecen otras redes como BITNET y UseNet. En 1989, se unen CSNET y BITNET. Esta época está relatada en el libro de Bruce Sterling (*The Hacker Crackdown*, 1994).

En 1992, Internet se hace comercial y allí comienza su importante crecimiento tanto en servidores como en usuarios.

8.2 World Wide Web

La *World Wide Web* (WWW o Web) es una red de recursos de información. La Web se basa en tres mecanismos para hacer que dichos recursos estén disponibles [Raggett98]:

1. Un esquema de uniforme de nombres para localizar recursos en la Web (por ej., URIs).

2. Protocolos, para acceder a recursos con nombre en la Web (por ej., HTTP).

3. Hipertexto, para tener navegación entre los recursos (por ej., HTML).

La WWW ha hecho que el público general acceda a Internet gracias a la introducción de interfaces gráficas, permitiendo a los usuarios experimentar vistas y sonidos en un estilo intuitivo de navegación.

8.2.1 Desarrollo de la World Wide Web

El precursor de la World Wide Web fue un pequeño sistema de hipertexto desarrollado en el CERN, el Laboratorio Europeo para Física de Partículas, para hacer seguimientos de información personal en un proyecto distribuido. La experiencia positiva preparó el desarrollo de lo que se volvería la WWW. En marzo de 1989, Tim Bernes-Lee en el CERN comenzó a hacer circular una propuesta para construir un sistema de hipertexto para compartir información entre grupos de investigadores separados geográficamente en la comunidad de Física de Alta Energía.

En octubre de 1990, comenzó a desarrollarse el proyecto de la Web. En navidad de 1990, el acceso a archivos de hipertextos y grupos de noticias se demostró con los *browsers* gráficos NeXT Step. Para fines de 1991, la publicación de un reporte de CERN anunció al mundo a la Web. Otros *browsers* para la WWW incluían a *Viola* (Pei Wei, UC Berkeley), *Mosaic* (Marc Andreesen, Illinois NSCA), *Cello* (Thomas Bruce, Cornell University) y *Lynx* en pantalla de texto (Lou Montulli, Universidad de Kansas) [Cheong96].

8.2.2 Crecimiento de la Web

Con el tiempo, la Web se volvió inmensamente popular en parte porque los *browsers* que hicieron fácil para cualquiera en Internet, recorrer (*surf*) el espacio de información de la Web. En abril de 1993, había 62 servidores de Web registrados en Internet. Para abril de 1994, el número de servidores registrados era de 829. Para mayo 1994, el número creció

a 1248 [Cheong96]. El crecimiento del tráfico de la WWW es igualmente impresionante [Cheong96].

Para julio de 1994, la Web había sobrepasado la habilidad del CERN para lidiar con ella. Así, CERN comenzó a transferir el proyecto de la Web a un nuevo grupo llamado Organización W3, una sociedad entre CERN y MIT con base en Cambridge, Massachussets, EE.UU. En 1995, esta sociedad floreció en el Consorcio de la WWW (*WWW Consortium*).

8.2.3 Diseminación de Información en la Web

La Web fue originalmente concebida como una manera conveniente de diseminar información en una organización. La Web se comporta como un repositorio de información, que acumula conocimiento, permitiendo compartir ideas como aspectos de un proyecto común a colaboradores en sitios remotos [Cheong96].

Como una herramienta para distribución de información, la Web se ha vuelto muy popular debido a su capacidad de proveer un medio de interacción entre sus usuarios flexible y extensible.

La información residente en la Web puede cambiar de forma mediante la alteración de los enlaces de hipertexto para modificar el conocimiento representado en ella de una manera dinámica.

Además, el diseño escalable de la Web no requiere ninguna administración centralizada de la información. Estas propiedades han ayudado a la Web a expandirse desde sus orígenes del CERN a la Internet independientemente de límites de naciones y de disciplinas.

La WWW organiza, transmite y recupera información de todo tipo usando una combinación de hipertexto, gráficos y tecnología *multimedia*, unificados en un conjunto de convenciones de nombrado, protocolos de redes y formatos de documentos llevados a cabo usando una arquitectura cliente/servidor.

8.3 Protocolos de Red

En esta sección, se describirán los distintos protocolos de redes relevantes a la aplicación. Si bien la parte importante es desde el punto de vista del programador, el tema se enfoca desde un punto de vista general.

TCP/IP es un conjunto de protocolos para interconectar los distintos sistemas en la Internet. TCP/IP provee una interfaz común de programación para hardware heterogéneo; también, TCP/IP soporta la unión de redes físicas separadas implementadas en medios distintos [Weber97].

Los protocolos de redes se estudian a través del uso de modelos, lo que hace que el objeto de estudio pueda observarse con distintos niveles de abstracción.

8.3.1 Modelo de Referencia OSI

La arquitectura de protocolos de red conocida como *Modelo de Referencia de Interconexión de Sistemas Abiertos*[1] (OSI) se usa para describir sistemas de redes. El esquema de OSI

[1] Open Systems Interconnect Reference Model.

es parte de los proyectos de la Organización Internacional para la Standarización[2] (ISO) [Weber97].

El modelo consiste de siete capas[3] que proveen funcionalidad específica. Cada capa posee características definidas y todas juntas permiten la comunicación en red. La implementación en software de tal modelo de capas se llama, apropiadamente, una *pila de protocolo*.

Las aplicaciones de usuario información en una capa y cada una encapsula los datos hasta que la última es alcanzada. Las capas tienen roles específicos, cada una cuidándose de no invadir el dominio de las otras, y, a su vez, dependiendo de ellas. Las capas son las siguientes (mostradas de mayor a menor nivel) [Weber97]:

- *Capa de Aplicación:* Contiene las aplicaciones de red con las cuales la gente interactúa, como *e-mail*, transferencia de archivos y *login* remoto.
- *Capa de Presentación:* Crea estructura de datos comunes.
- *Capa de Sesión:* Maneja las conexiones entre aplicaciones de red.
- *Capa de Transporte:* Asegura que los datos sean recibidos exactamente como se mandaron.
- *Capa de Red:* Enruta los datos a través de varias redes físicas mientras viajan a un *host* conocido.
- *Capa de Enlace de Datos:* Transmite y recibe paquetes de información en forma confiable a través de una red física uniforme.
- *Capa Física:* Define las propiedades físicas de la red, como los niveles de voltaje, tipos de cable y *pins* de interfaces.

8.3.2 Modelo de Red TCP/IP

Al verse como un modelo de capas, TCP/IP se ve como un modelo compuesto de cuatro capas (de mayor a menor nivel): de aplicación, de transporte, de red y de enlace. Los intentos de mapear estas capas al modelo OSI son inexactos. Como en el modelo OSI, cada capa TCP/IP tiene un rol específico. Las capas del modelo son [Weber97]:

- *Capa de Aplicación:* Las aplicaciones de red dependen de la definición de un diálogo claro. En un sistema cliente/servidor, la aplicación cliente sabe como requerir servicios y el servidor sabe como responder apropiadamente. Los protocolos que implementan esta capa incluyen HTTP, FTP y Telnet [Weber97].
- *Capa de Transporte:* La capa de transporte permite a las aplicaciones de red obtener mensajes sobre canales claramente definidos y con características específicas. Los dos protocolos dentro de TCP/IP que implementan esta capa son el *Protocolo de Control de Transmisión*[4] (TCP) y el *Protocolo de Datagramas de Usuario*[5] (UDP).

[2]International Organization for Standarization.
[3]Las capas se conocen como *layers*.
[4]Transmission Control Protocol.
[5]User Datagram Protocol.

- *Capa de Red:* La capa de red permite que la información sea transmitida a cualquier máquina en la red contigua TCP/IP, independientemente de las diferentes redes físicas intervinientes. El *Protocolo de Internet*[6] (IP) es el mecanismo para transmitir datos en esta capa.

- *Capa de Enlace:* La capa de enlace consiste de protocolos de bajo nivel usados para transmitir datos a máquinas en la misma red física. Protocolos, que no son parte la *suite* TCP/IP como Ethernet, Token Ring, FDDI y ATM, implementan esta capa.

Los datos dentro de estas capas son usualmente encapsulados con un mecanismo común: los protocolos tienen un encabezado o *header*, identificando metainformación como la fuente, el destino y otros atributos, y una porción de datos que contiene la información real. Los protocolos de los niveles superiores se encapsulan en la porción de datos de los niveles inferiores. Cuando los datos viajan por la pila del protocolo, la información se reconstruye a medida que los datos se envían a cada capa.

8.3.3 Protocolo TCP/IP

Internet Protocol

El *Protocolo de Internet* (IP)[7] es la pieza clave de la suite TCP/IP. Todos los datos en la Internet fluyen a través de *paquetes IP*, las unidades básicas de las transmisiones IP. IP se denomina un protocolo *sin conexión* y *no confiable*. Como protocolo sin conexión, IP no intercambia información de control antes de transmitir datos a un sistema remoto –los paquetes se mandan a destino con la esperanza de que sean tratados apropiadamente. IP es no confiable porque no retransmite paquetes perdidos o detecta datos corruptos. Estas tareas deben implementarse por protocolos de más alto nivel como TCP [Weber97, p. 421].

IP define un esquema de direccionamiento universal llamado direcciones IP. Una *dirección IP* es un número de 32 bits único en Internet. Dado un paquete IP, la información puede rutearse a destino en base a la dirección IP en el header. Las direcciones IP generalmente se escriben como cuatro números, entre 0 y 255, separados por puntos (por ejemplo, `123.23.234.67`) [Weber97].

Mientras que un número de 32 bits es una manera apropiada de sistemas de direcciones para computadoras, los humanos naturalmente tienen dificultades para recordarlas. Así, se desarrolló un sistema llamado *Sistema de Nombres de Dominio* (DNS)[8] se desarrolló para traducir direcciones IP a identificadores más intuitivos y viceversa. Por ejemplo, se puede usar `www.netspace.org` en vez de `128.148.157.6` [Weber97].

Transmission Control Protocol

Muchas aplicaciones de Internet usan el *Protocolo de Control de Transmisión* (TCP) para implementar la capa de transporte. TCP es un protocolo con las siguientes cualidades [Weber97]:

[6] Internet Protocol
[7] Internet Protocol.
[8] Por *Domain Name System*.

- *Confiable*: Cuando los segmentos TCP, las unidades más pequeñas de las transmisiones, se pierden o corrompen, la implementación TCP va a detectarlo y retransmitirá los segmentos necesarios.

- *Orientado a la conexión*: TCP establece una conexión con un sistema remoto transmitiendo información de control, generalmente conocido como *handshake*, antes de comenzar una comunicación. Al final de la conexión, un handshake de cierre termina la transmisión.

- *De Flujo constante*: TCP provee un medio de comunicación que permite que un número arbitrario de bytes se envíen y reciban suavemente; una vez que se estableció una conexión, los segmentos TCP le proveen a la capa de aplicación la apariencia de un flujo continuo de datos.

Por estas características, las aplicaciones basadas en TCP no deben preocuparse sobre cómo están codificados los datos o hacer correcciones de errores. El precio a pagar es la sobrecarga de procesamiento.

Un concepto importante es el de puerto. Los puertos o *ports* separan varios flujos de comunicación TCP que están ocurriendo concurrentemente en un sistema. Para aplicaciones servidor, las cuales esperan que clientes TCP inicien contacto, se puede establecer un puerto específico desde donde las comunicaciones van a originarse. Estos conceptos se unen en una abstracción de programación conocida como *socket* [Weber97].

8.3.4 Network News Transfer Protocol

El *Protocolo de Transmisión de Noticias de Red* es el protocolo subyacente para acceder a los servidores de noticias de Internet. Los servidores de noticias escuchan en el puerto 119.

Los comandos básicos incluyen [Bigus98]:

1. *GROUP*: para seleccionar un grupo de noticias (*newsgroup*);

2. *STAT*: para ubicar el cursor en un artículo determinado;

3. *NEXT*: para caminar a través de los artículos de un newsgroup, y,

4. *HEAD* y *BODY*: para recuperar datos concernientes a los artículos individuales.

El NNTP es la base de la implementación del filtrador de grupos de noticias descripto en [Bigus98] (sección 9.3.2).

8.3.5 HyperText Transfer Protocol

En esta sección se describirá el Protocolo de Tranferencia de Hipertextos (HTTP) que se usa en la capa de aplicación del protocolo TCP/IP.

Esta sección está basada en el trabajo de Cheong [Cheong96, p. 125–151], si bien las especificaciones del protocolo pueden hallarse en la Web[9].

[9] `http://ietf.cnri.reston.va.us/internet-drafts/draft-ietf-http-v10-spec-03.txt`

Modo de Operación de HTTP

HTTP está basado en el paradigma de interacción *requerimiento/respuesta*, esto quiere decir que un programa llamado *cliente* hace un requerimiento a otro programa llamado *servidor*[10]; el tiempo entre el requerimiento y la respuesta se llama *conexión*. HTTP se dice *stateless* porque no guarda información de estado entre conexiones distintas [Cheong96].

Mensajes con HTTP

Los mensajes en HTTP deben ir precedidos por un encabezado; hay cuatro tipos de encabezados [Cheong96]: generales, de requerimiento, de respuesta y de entidad[11].

1. *Campos de Encabezados Generales de Mensaje:*

   Los encabezados generales contienen información respecto a la fecha del mensaje, URI[12] de qué proxy server "forwardeó" el mensaje, identificadores de mensajes, y, si el mensaje cumple con la norma MIME[13][Cheong96, p. 130].

2. *Mensajes de Requerimiento:*

   Un cliente WWW hace requerimientos a un servidor WWW para comenzar una operación. Un mensaje de requerimiento de un cliente a un servidor incluye la siguiente información en la primera línea[14]: el método a aplicarle al recurso requerido, el identificador del recurso, y, la versión del protocolo usada [Cheong96, p. 130].

   El método indica el método a realizar en el recurso identificado por el URI del requerimiento. La siguiente es la lista de los métodos permitidos [Cheong96, p. 131]:

   - *GET*: El método GET recupera información (en la forma de una entidad) identificada por un URI. Si el URI referencia a un proceso, lo que se retorna es la salida del proceso, como en el caso de CGI (sección 8.8). El método GET se vuelve un GET condicional cuando el mensaje de requerimiento incluye un encabezado *If-Modified-Since*[15]. Un GET condicional transfiere la entidad sólo si ésta fue modificada después de la fecha especificada, en caso contrario el servidor devuelve un código 304 (ver tabla B); ésto se usa para ahorrar ancho de banda en el caso de haber servidores proxies (ver sección 8.5.3).
   - *HEAD*: El método HEAD es equivalente al GET salvo no retorna una identidad sino la metainformación asociada a ella. Se usa para testear la validez de los enlaces.

[10] Cualquier programa puede ser cliente o servidor, el nombre sólo hace referencia al rol jugado por el programa en la transacción considerada.

[11] Una *entidad* es una representación de un recurso (una página HTML, una foto, un texto, etc.) que puede enmarcarse en un mensaje de requerimiento o de respuesta. Una entidad consiste de metainformación (en la forma de encabezados de entidad) y contenido (en la forma de cuerpo de una entidad).

[12] La explicación del concepto de URI y URL se da en la sección 8.6.

[13] MIME: Multi-purpose Internet Mail Extensions es una especificación que permite intercambiar texto en lenguajes con conjuntos de caracteres diferentes y correo electrónico multimedia [Cheong96, p. 126].

[14] Llamada *línea de requerimiento.*.

[15] Si-Modificado-Desde.

- *POST*: El método POST se usa para requerir que el servidor de destino acepte la entidad encerrada en el requerimiento como un nuevo subordinado del recurso identificado por el URI del requerimiento. POST está designado para cubrir las siguientes funciones: anotar recursos existentes, mandar un mensaje a un BBS, grupo de noticias, lista de correo, etc., proveer un block de datos (usualmente un formulario –ver sección 8.8– a un proceso, y, extender una base de datos a través de una operación de *append*.

  La función realizada por el método POST es determinada por el servidor y es usualmente dependiente de la URI del requerimiento. No es necesario que este método cree una entidad (con una URI) en el servidor, aunque sí puede hacerlo. En el primer caso, se puede devolver un código 200 o 204 (ver tabla B); mientras que en el segundo, la respuesta debería ser un código 201.

- *PUT*: El método PUT requiere que la entidad encerrada sea guardada bajo el URI entregado en el requerimiento; si el URI hace referencia a un objeto ya existente, el nuevo debe considerarse una actualización del viejo. Si el requerimiento se completa satisfactoriamente, debe retornarse un código 200 (ver tabla B).

  La diferencia entre POST y PUT es que en POST el método identifica al proceso que va a manejar la entidad mientras que en el PUT es al revés.

- *DELETE*: El método DELETE requiere que el servidor borre el recurso identificado por el URI. Una respuesta satisfactoria debería ser una de: 200, 202 ó, 204.

- *LINK*: El método LINK establece una o más relaciones de enlace entre el recurso identificado por el URI y otros recursos existentes. La diferencia entre el LINK y los otros métodos es que no permite el envío de ninguna entidad ni la creación de nuevos recursos.

- *UNLINK*: El método UNLINK remueve uno o más enlaces.

Si se hace un requerimiento con un método no soportado por el servidor, éste retornará un código 501 (ver tabla B).

3. *Mensajes de Respuesta HTTP:*

   Después de recibir e interpretar un mensaje de requerimiento, un servidor responde en la forma de un *mensaje de respuesta HTTP*.

   Si un cliente envía un requerimiento HTTP 1.0 y recibe una respuesta que no comienza con una línea de estado, debería asumir que la respuesta es simple y "parsearla" acordemente. Luego el servidor cerrará la conexión.

   La línea de estado consiste de tres partes: la versión del protocolo, un código númerico y una frase textual asociada. Las clases de códigos de respuesta HTTP están resumidas en la tabla B, mientras que la descripción de los códigos de respuesta de éxito, de redirección, de error del cliente y de error del servidor están explicitadas en las tablas B, B, B y B respectivamente.

8.3.6 File Transfer Protocol

FTP quiere decir *File Transfer Protocol*, como su nombre lo indica es un protocolo usado para transferir archivos de una computadora a otra sobre una red TCP/IP [Dwight97].

Los lenguajes de programación C y Java poseen primitivas para poder realizar dicha transferencia de archivos [Kruglinski97, Weber97].

8.3.7 Otros Protocolos

Existen otros protocolos importantes que no se tratan aquí debido a que no se requieren para comprender la implementación del agente *Querando!*. Entre ellos podemos hallar a POP, SMTP, UDP, etc. [Dwight97, Weber97].

8.4 Otros Sistemas de Información de Internet

Además de la Web, hay otros sistemas de información como *Gopher* y *WAIS* que también usan una arquitectura cliente/servidor similar. Estos sistemas, sin embargo, juegan roles diferentes. *Gopher*, que es una especie de Web pero sin capacidades completas de hipertextos, usa un sistema de menúes que permiten que la información se organice en una jerarquía de directorios. *WAIS* (*Wide Area Information System*) no provee ninguna facilidad de navegación y usa indexación exclusivamente para llevar a sus usuarios a la información requerida del espacio de información.

Usando la analogía de un libro como un espacio de información, *Gopher* generalmente se describe como su tabla de contenidos, *WAIS* como el índice alfabético y la WWW como el cuerpo de hipertexto donde reside el grueso de la información [Cheong96].

8.4.1 Gopher

Gopher es un sistema de recuperación de documentos cliente/servidor que empezó como Sistema de Información de Campus en la Universidad de Minnesota. Gopher está definido en la RFC 1436 [Dwight97].

Gopher organiza la información en dos maneras: árbol y cristal (grafo) [Gach96].

Su interfaz básica es un menú, el cual está estructurado en forma de árbol, lo cual representa una jerarquía de directorios y subdirectorios, etc.

El bosque de todos los árboles gopher se llama "gopherespacio". Un servidor gopher apunta a otro servidor gopher y así sucesivamente.

La estructura cristalina de Gopher es una versión primitiva del hipertexto de documentos HTML de la WWW.

Gopher posee un buscador llamado *Veronica*.

En particular, hoy en día los browsers devolverán la estructura de un directorio Gopher como una página HTML, usando la transparencia provista por un servidor HTTP.

8.4.2 WAIS

WAIS (Wide Area Information Systems) o Sistemas de Información de Área Ancha[16] es un estándar abierto para construir un índice del texto completo de sus documentos y, de esta

[16]En otras fuentes se entiende como Servidor de Información de Área Ancha [Gach96].

manera, crear una bases de datos WAIS. Entonces, en esta base de datos se puede buscar desde programas CGI especiales que saben cómo leer la misma [Dwight97, Cheong96].

Algunos servidores proveen soporte propio para búsquedas WAIS, eliminando de esta manera la necesidad de guiones CGI externos.

WAIS soporta consultas por claves de búsqueda, es decir, se especifican un conjunto de términos y se devuelve una lista rankeada (de mejor a peor) de los documentos que las contienen. Una búsqueda WAIS usa tres criterios simultáneos: cuán frecuente ocurre la palabra, cuán cerca ocurre la palabra de ella misma y cuán cerca está la palabra del principio del texto [Gach96].

Los tipos de documentos soportados por los servidores WAIS son: bibtex, dash, dvi, gif, html, para, pict, ps, text, piff [Dwight97].

Existen productos como *freeWAIS*, para instalar búsquedas WAIS en un servidor propio [Dwight97].

8.5 Clientes, Servidores y Proxies

Un cuerpo de software comprende a la Web en una forma concreta. Esta arquitectura de software se compone de los siguientes componentes para interoperar en la Internet [Cheong96]:

- *Clientes* que permiten a los usuarios navegar la Web o aún interactuar con el servidor en maneras interesantes.
- *Servidores* que permiten a los sitios de Internet publicar información o exportar datos al mundo.
- *Proxies* que facilitan las comunicaciones y proveen control de acceso para sitios que deben confiar en *hosts* intermedios para comunicación con la Internet (por ejemplo, sitios detrás de un *firewall*).

8.5.1 Clientes Web

Un programa cliente WWW corre en una computadora de escritorio y es capaz de acceder a diferentes servidores Web distribuidos a lo largo de Internet. Con un *browser* de Web cliente, los usuarios pueden ver documentos hipermedia siguiendo los enlaces de información. Además de proveer funciones básicas de *browsing*, los clientes web pueden solicitar información a los usuarios a través de formularios (secciones 8.7.3 y 8.8), *Java applets* (sección 8.10.5) o controles *VBScript* o *JavaScript* (sección 8.10.6).

8.5.2 Servidores Web

Un programa servidor de WWW usualmente corre en una estación de trabajo con el poder de cómputo suficiente para manejar múltiples requerimientos de clientes de toda la Internet. El requerimiento más común es pedir (*GET*) una página Web para mostrarla en un *browser*. Entre los productos que hay para servidores tenemos: Windows NT o Apache Server [Dwight97].

Además de servir requerimientos de documentos de hipertexto, un servidor de Web actúa también como un *gateway* a otro software o fuentes de información como por ejemplo una base de datos. Usando CGI, el servidor de Web invoca un programa llamado *script* o guión de *gateway* que toma la información provista por el cliente (usualmente un formulario) que aparece en el *browser* cliente) y lo procesa de acuerdo a las instrucciones del guión y retorna como resultado una página Web al cliente.

8.5.3 *Web Proxies*

Un *proxy* es un servidor de Web que usualmente corre en una máquina *firewall*, es decir, una máquina que actúa de barrera de seguridad entre la Internet y la LAN (red de área local) que lo contuviera, por ejemplo en una organización [Cheong96, Dwight97, Gach96]. El *proxy* actúa de intermediario entre los clientes de Web detrás del *firewall* y los servidores de Web en la Internet.

Cuando el *proxy* recibe un requerimiento de una de las máquinas internas detrás del *firewall*, envía el requerimiento a dicho servidor y cuando éste llega, recién es mandado a la máquina que originó el pedido.

Cachés

Un proxy también puede usarse para cachear documentos Web, lo cual es útil cuando múltiples clientes dentro de una LAN (no necesariamente detrás de un *firewall*) hacen pedidos de las mismas páginas Web. El *proxy* guardará los resultados de dichos pedidos y simplemente devolverá dichas páginas guardadas cuando ocurrieran los próximos pedidos de la misma, reduciendo de esta manera los tiempos de respuesta para los clientes [Cheong96].

8.6 URI y URL: Identificadores y Ubicadores de Recursos Universales

Mientras que las direcciones IP identifican unívocamente sistemas en Internet y los puertos identifican servicios TCP o UDP en un sistema, los URIs (Universal Resource Identifier)[17] proveen un esquema universal de identificación en el nivel de aplicación. Los URIs se desarrollaron para crear un forma común de identificar recursos en la Web, pero fueron diseñados para ser lo suficientemente generales para cubrir las aplicaciones que están en Internet desde hace décadas así como para acomodarse a protocolos futuros [Weber97].

8.6.1 Sintaxis de las URLs

La clasificación primaria para las URL es el *esquema*, el cual usualmente responde a un protocolo de aplicación. Los esquemas incluyen *http*, *ftp*, *telnet* y *gopher*. El resto de la sintaxis de la URL está en un formato que depende del esquema. Estas dos porciones de información están separadas por un carácter dos puntos:

[17]Por *Identificador de Recurso Universal*, aunque el término más conocido es URL (*Universal Resource Locator* o *Ubicador de Recurso Universal*, que es una forma más restringida de URI [Raggett98]. Por ello, en este trabajo los términos URI y URL se usarán indistintamente.

```
nombre-de-esquema:información-de-esquema
```

Así, mientras que la URL `mailto:jperez@pepe.com.ar` significa "mande un mail al usuario `jperez` en la máquina `pepe.com.ar`", la URL `ftp://jperez@pepe.com.ar` significa "abra una conexión FTP a `pepe.com.ar` y *loguéese* como el usuario `jperez`".

8.6.2 Formato General de las URLs

Muchas de las URLs se adaptan al siguiente patrón [Weber97][18]:

```
nombre-de-esquema://host:port/file-info#internal-reference
```

`Nombre-de-esquema` es un esquema de URL como HTTP, FTP o Gopher. `Host` es el nombre de dominio o dirección IP del sistema remoto. `Port` es el número de puerto donde el servicio está escuchando; como muchos protocolos de aplicación definen un puerto estándar, a menos que se use un puerto no estándar, el puerto y los dos puntos que lo separan del host se omiten. `File-info` es el recurso requerido en el sistema remoto, el cual, generalmente, es un archivo. Sin embargo, la porción de archivo (file-info) puede en realidad ejecutar un programa servidor y usualmente incluye un *path* a un archivo específico en el sistema. La parte `internal-reference` es usualmente el identificador de un ancla dentro de una página HTML; en el contexto de [Raggett98], las referencias internas se conocen como *indentificadores de fragmento*.

URLS Absolutas y Relativas

Las URLs pueden ser absolutas o relativas [Raggett98]. Una *URL absoluta* tiene el formato general, es decir, especifica un nombre de esquema. Por otro lado, una *URL relativa* no contiene información de esquema. Su path generalmente se refiere a un recurso en la misma máquina en la que está el documento corriente. Las URLs relativas pueden contener componentes de camino relativas (por ejemplo, ".." significa un nivel más arriba en la jerarquía definida para el camino). Las *URLs relativas* se resuelven en URLs absolutas usando una URL base.

Por ejemplo, considere un documento HTML con URL

http://www.unsitio.com/dir1/dir2/paginas/pepe.htm

y el tag para una imagen `<IMG SRC='../iconos/logo.gif'>`, entonces la URL absoluta correspondiente será *http://www.unsitio.com/dir1/dir2/iconos/logo.gif*. Análogamente, para el ancla de hipertexto `<A HREF='juan.htm'>` con la URL relativa *juan.htm*, la URL absoluta correspondiente será *http://www.unsitio.com/dir1/dir2/paginas/juan.htm*.

Resolución de URLs Relativas en *Querando!*

En la implementación de *Querando!*, se necesitó programar la clase *URLSplitter*, que, justamente, realiza la resolución de URLs relativas en URLs absolutas. Esta necesidad surge del hecho de que las páginas filtradas se mandan del servidor donde reside *Querando!*; por lo tanto, las referencias relativas dentro de las páginas deben resolverse al sitio de origen para que sean correctas.

[18] Una descripción más detallada se puede hallar en *http://www.netscape.org/users/dwb/url-guide.html*.

Esto se resuelve modificando el código interno de las páginas HTML que pasan a través del filtrador. La modificación requiere cambiar todos los tags HTML que contengan referencias a URIs. Dichos tags HTML se detallan en la tabla 10.2.1 y están basados en [Gulbransen98].

8.7 HyperText Markup Language

En capítulos previos, se dio un panorama inicial del lenguaje de descripción de hipertextos HTML (sección 4.1). Como, por un lado, la interfaz de *Querando!* está basada en formularios HTML, se esplicarán los conceptos de HTML relacionados con esta tarea.

8.7.1 Anclas de Hipertexto

En HTML, las anclas de hipertexto se denotan con el tag `A` [Raggett98, Scharf96]. Por ejemplo, el código

```
<A HREF='http://www.unservidor.com/otraPagina.htm'>Otra Pagina</A>
```

tiene como efecto que el texto *Otra Pagina* aparezca subrayado en el documento HTML indicando que es un salto de hipertexto, mientras que el enlace es hacia el documento *http://www.unservidor.com/otraPagina.htm*.

Los enlaces pueden indicar una URL absoluta o relativa al sitio de origen. Una URL absoluta especifica la URL completa de una entidad web. Una URL relativa especifica la ubicación de un objeto en forma relativa a la URL del documento que se está mostrando en el browser.

8.7.2 Frames

Los *frames* (marcos) HTML les permiten a los autores presentar documentos en vistas múltiples, las cuales pueden ser ventanas independientes o subventanas [Raggett98]. Las vistas múltiples les permiten tener cierta información visible mientras que vistas son "scroleadas" o reemplazadas. Por ejemplo, dentro de una misma ventana, un marco podría mostrar un banner estático, otro menú de navegación y un tercero un documento principal para leer.

En la figura 8.7.2, se puede apreciar un ejemplo de un documento HTML con tres marcos, dos corresponden a otros documentos HTML y uno a una foto; En la figura 8.2 se ve el *layout* de la página resultante.

Los marcos son relevantes ya que dada una URL, cuando *Querando!* quiera obtener la representación la página asociada, si ésta posee marcos, deberá recuperar las páginas enmarcadas.

8.7.3 Formularios

Un *formulario* (form) HTML es una sección de un documento HTML con contenido normal y además elementos especiales llamados controles (checkboxes, radio botones, menués, áreas de texto, etc.) y etiquetas sobre dichos controles [Raggett98, Scharf96]. Los usuarios pueden completar un formulario modificando sus controles (ingresando texto,

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN"
"http://www.w3.org/TR/REC-html40/frameset.dtd">
<HTML>
<HEAD>
<TITLE>A simple frameset document</TITLE>
</HEAD>
<FRAMESET cols="20%, 80%">
<FRAMESET rows="100, 200">
<FRAME src="contenido_del_frame1.html">
<FRAME src="contenido_del_frame2.gif">
</FRAMESET>
<FRAME src="contenido_del_frame3.html">
<NOFRAMES>
<P>Este documento contiene::
<UL>
<LI><A href="contenido_del_frame1.html">Unos contenidos</A>
<LI><IMG src="contenido_del_frame2.gif" alt="Una imagen">
<LI><A href="contenido_del_frame3.html">Otros contenidos</A>
</UL>
</NOFRAMES>
</FRAMESET>
</HTML>
```

Figura 8.1: Ejemplo de página con frames.

Dos frames corresponden a otros documentos HTML y uno a una foto. Para los browsers que no soportan marcos, existe el tag *NOFRAMES*.

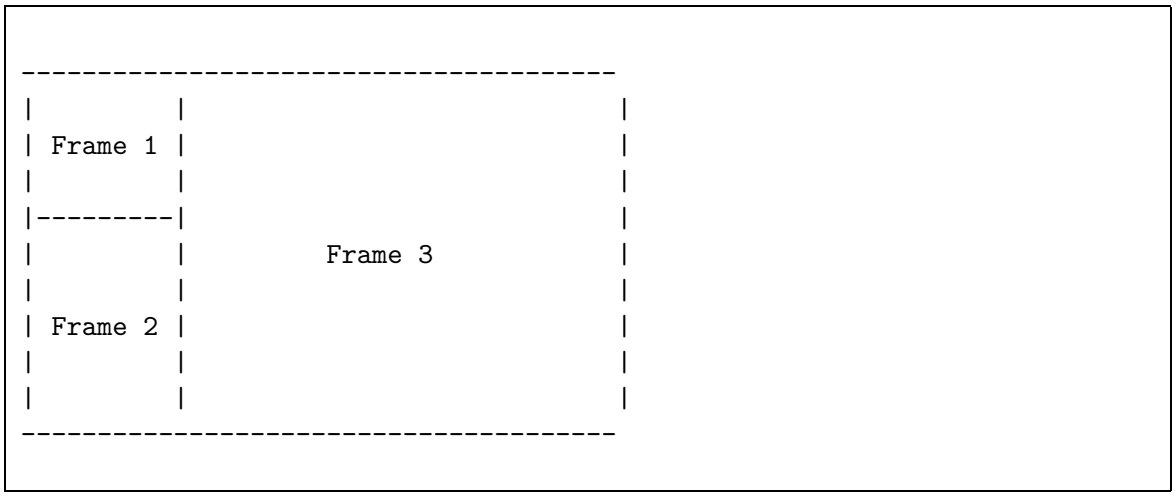

Figura 8.2: Forma en que se ve la página de la figura 8.7.2.

seleccionando ítems de menúes, etc.), antes de enviar los datos a un agente para que los procese (por ej., un servidor web, un servidor de correo electrónico, etc.).

En [Dwight97, Raggett98, Scharf96] se puede hallar una descripción completa de los controles disponibles en los formularios HTML. Además, el uso de formularios se profundizará en la sección 8.8.

8.7.4 Scripts

A través de los *scripts* (guiones), los autores pueden crear páginas Web dinámicas –por ej., "formularios inteligentes" que reaccionan cuando los usuarios los llenan (secciones 8.7.3 y 8.10.6)– y usar HTML como un medio para construir aplicaciones basadas en red [JS99, Gulbransen98, Raggett98]. Los mecanismos provistos por HTML para incluir scripts en un documento HTML son independientes del lenguaje de scripts.

Un script del lado del cliente (*client-side*) es un programa que puede acompañar un documento HTML o puede estar embebido directamente en él. El programa se ejecuta directamente en la máquina cliente cuando el documento se carga o en algún otro momento, por ejemplo, cuando se activa un enlace.

Los scripts ofrecen un medio para extender los documentos HTML en maneras interactivas. Por ejemplo [JS99, Raggett98, Gulbransen98]:

- Los scripts pueden evaluarse cuando un documento se carga para modificar los contenidos de un documento dinámicamente.
- Los scripts pueden acompañar un formulario para procesar la entrada a medida que ésta se ingresa. Los diseñadores pueden dinámicamente llenar parte de un formulario en base a los valores de otros campos. También pueden asegurar que los campos sean mutuamente consistentes, que estén en un rango determinado de valores, etc.
- Los scripts pueden dispararse por eventos que afecten al documento, como la carga, descarga, el foco en un elemento, el movimiento del ratón, etc.
- Los scripts pueden enlazarse a controles en un formulario (por ej., botones) para producir elementos de interfaz gráfica de usuario.

Los scripts van entre las cláusulas HTML `<SCRIPT>` y `</SCRIPT>`.

8.7.5 Hojas de Estilo en Cascada

Las *Hojas de Estilo en Cascada* (CSS) son simplemente un conjunto de definiciones acerca de la forma en que es preciso visualizar cada uno de los elementos HTML de una página, o para indicar la forma en la que tienen que aparecer para el usuario en la ventana del navegador [Bert98, Gulbransen98].

En HTML, la especificación del estilo se hace a través del atributo `STYLE` de un marcador, por ejemplo `<P style='text-indent:30'>` hará que un párrafo aparezca sangrado. La otra forma, relevante a la caracterización de las páginas HTML (sección 4.2) es el código entre dos marcadores `<STYLE>` y `</STYLE>`.

```
<HTML>
   <HEAD>
      <TITLE>Titulo de la Pagina
   </HEAD>
   <BODY>
      <FORM>
         ... Aqui va el codigo HTML para controles varios...
      </FORM>
   </BODY>
</HTML>
```

Figura 8.3: Estructura general de un formulario HTML.

8.8 Common Gateway Interface

La interfaz de usuario del agente *Querando!* está basada en formularios HTML que comunican datos entre programas CGI. Por ello, en esta sección se explica cómo funciona esta técnica.

8.8.1 Definición

La *Interfaz común de Gateway* (CGI) es un protocolo de comunicación que define cómo la información recolectada por un usuario a través de un formulario (*form*) en un *Web browser*, será pasada a través de un servidor de Web a un programa ejecutable, y como los resultados serán enviados de vuelta par ser mostrados al usuario por el *Web browser* [Duncan96, Dwight97].

La estructura general de un formulario HTML aparece en la figura 8.3. El texto del formulario y las etiquetas (*tags*) HTML para los controles del formulario -como cajas de entrada de texto (*text entry boxes*), menúes desplegables, radio botones, casillas de verificación (*check boxes*) y botones de *Submit* (Enviar) y *Reset* (Cancelar)- van ubicados entre los tags `<FORM>` y `</FORM>`.

La etiqueta `<FORM>` tiene dos atributos o modificadores: `METHOD` y `ACTION`. El atributo `METHOD` puede tomar uno de dos valores, `GET` o `POST`, que especifican la forma en que la información del formulario va a ser presentada al programa que la procese. El modificador `ACTION` es usado para especificar la URL (ver Sección 8.6) de un programa ejecutable o *script* que va procesar la información.

Consideremos el esqueleto de un formulario HTML en la figura 8.4. En este caso, cuando el usuario clickee el botón con la etiqueta *Aceptar Formulario*, el *Web browser* recolectará la información entre las etiquetas `<FORM>` y `</FORM>` y la enviará al servidor de Web que le mandó el formulario. El servidor de Web a su vez activará el programa especificado por el modificador `ACTION`, conocido como el programa CGI o programa de *gateway*, y le pasará los datos del formulario en la manera especificada por el modificador `METHOD`.

```
<HTML>
  <HEAD>
    <TITLE>Titulo de la Pagina</TITLE>
  </HEAD>
  <BODY>
    <FORM METHOD="POST"
     ACTION="http://www.pepedom.com/cgi-bin/mi_programa.exe">
      ... Codigo HTML para controles va aqui...
    <INPUT TYPE="submit" VALUE="Enviar Datos del Formulario">
    <INPUT TYPE="reset"  VALUE="Resetear Datos">
    </FORM>
  </BODY>
</HTML>
```

Figura 8.4: Esqueleto de un formulario HTML.

Es importante notar que tanto el cliente Web, como servidor y el programa CGI pueden estar ubicados en cualquier parte de la red, es decir, que podrían estar en la misma máquina o en tres máquinas diferentes. Típicamente, el servidor de Web y el programa CGI residen en la misma máquina por razones de seguridad. De hecho, en muchos servidores, los programas CGI pueden ser ejecutados sólo si residen en un directorio autorizado particular (llamado por convención *cgi-bin*).

8.8.2 Forma de Trabajo de CGI

El programa CGI encuentra la información que le pasó el servidor de Web en uno de dos lugares. Si el programa CGI es invocado por una etiqueta `<FORM>` con el método `GET`, la información del formulario será accesible en una variable de ambiente llamada `QUERY_STRING` [Dwight97]. Si es invocado con el método `POST`, la información del formulario aparecerá como una cadena de caracteres (*string*) en el dispositivo de entrada standard [Dwight97], como si la información hubiera sido tipeada en el teclado de la misma máquina o hubiera sido pasada al programa, vía redirección de entrada desde un pipe o un archivo con los operadores `|` o `<` desde la línea de comandos. Cualquier otra información útil es pasada al programa CGI a través de un conjunto de variables de ambiente predefinidas, como se ve en la tabla B.

Con respecto al pasaje del contenido de la información del formulario al programa CGI, cada nombre de control y su valor son apareados, separados por un signo igual. Cada par de nombres de control y valores son concatenados en una cadena de caracteres simple separados por ampersands (`&`). Los caracteres espacio, de puntuación y control son mapeados a sus equivalentes hexadecimales, con el signo porciento (`%`) como carácter de escape. Abstractamente, `var1=val1&var2=val2&...&varN=valN`. Por ejemplo supongamos que el formulario tuviera un área de texto (*text-box*) de control llamado nombre_usuario que contuviera "`Jorge Luis Borges`" y otra área de texto llamada ocupación que contuviera "`bibliotecario, escritor, profesor`". Cuando el botón *submit* de este formulario

sea *clickleado*, la cadena de caracteres siguiente:

nombre_usuario=Jorge%20Luis%20Borges
&ocupacion=bibliotecario%2C%20escritor%2C%20profesor

será enviada al programa CGI via el dispositivo de entrada standard o la variable de ambiente `QUERY_STRING`[19].

El programa CGI es responsable de parsear esta cadena de caracteres y recuperar los nombres de control originales y sus valores provistos por el usuario del *Web browser*, traducir combinaciones de caracteres de escape en los caracteres originales, procesar los datos del formulario y retornar el resultado de la transacción CGI al usuario. Lo último se realiza simplemente escribiendo etiquetas HTML al dispositivo de salida standard luego de escribir el encabezado HTTP:

```
Content-type: text/html\n\n
```

El *stream* de salida es entonces capturado por el servidor de Web y pasado de vuelta al cliente Web que envió el formulario en primer lugar. En otras palabras, la página Web vista por el usuario como resultado del envío del formulario no existe necesariamente como un documento HTML. Puede ser generada en el momento por el programa CGI.

8.8.3 Escritura de Programas CGI

En virtud de que un programa CGI obtiene sus datos de variables de ambiente y del dispositivo de entrada standard y envía los resultados de su procesamiento al dispositivo de salida standard, se puede escribir un programa CGI en casi cualquier lenguaje, desde Visual Basic a C++ [Kernigham85]. Por esa razón, un programa CGI es libre de comunicarse con tantos programas como sea necesario para llevar a cabo su tarea, incluyendo programas y servidores de bases de datos corriendo en otras máquinas, usando los mecanismos de comunicación entre procesos soportados por el sistema operativo huésped. De esta manera, la naturaleza espartana de la interfase CGI confiere una flexibilidad tremenda.

En máquinas Unix, los programas CGI usualmente se implementan en Perl, un lenguaje interpretado ideado originalmente para automatizar tareas de administrador de servidor [Dwight97].

8.9 Personal Web Server

El *Servidor Personal de Web*[20] (PWS) del sistema operativo Microsoft Windows 98 es un servidor de Internet/intranet que viene empaquetado con Microsoft FrontPage. En Windows NT, el mismo producto se conoce como Microsoft Internet Information Server (IIS); éste es más completo que el PWS, toma ventaja de las características de NT como puertos de completación de E/S, *file-caching handling*[21] y escalado de CPU para hilos[22] [Kruglinski97]. En resumen, tanto como PWS como IIS, simulan un servidor de Web en

[19] Dependiendo de si se usó `POST` o `GET`, respectivamente.
[20] Personal Web Server.
[21] Manejo de cacheo de archivos.
[22] Threads.

una máquina que en realidad no está conectada a la red, lo que permite desarrollar sitios Web en aislamiento y poder hacer el debugging sin comprometer el funcionamiento del sitio verdadero.

8.9.1 Servicios Disponibles

El PWS brinda los servicios de HTTP, mientras que IIS brinda además gopher y WWW.

8.9.2 Directorios PWS e IIS

La raiz del sitio Web del PWS será el directorio `C:/Inetpub/wwwroot/` (si está instalado en el disco `C:/`). También permite mapear directorios del disco rígido al sitio Web (llamados *directorios virtuales*. También permite configurar la seguridad de los directorios, para habilitarlos para lectura y/o ejecución [Kruglinski97, WinHelp98].

8.10 Alternativas y Complementos a CGI

En esta sección se discuten las alternativas a los guiones CGI para construir aplicaciones basadas en la Web.

8.10.1 ISAPI

ISAPI (Internet Server Application Programming Interface) es un standard que tiene más ventajas sobre los CGI [Dwight97].

La sobrecarga de disparar tareas CGI externas es demasiado alto, el tiempo de respuesta demasiado lento, y coordinar las tareas complica al servidor. En vez de usar ejecutables o scripts interpretados, se propone usar DLLs (dynamic link libraries). Las DLLs tienen las siguientes ventajas [Dwight97]:

- Viven en el espacio de procesos del servidor. Esto hace el intercambio más eficiente que los intercambios por los pipes stdin/stdout.
- Pueden cargarse en memoria hasta que no se necesiten, acelerando la velocidad de ejecución.
- En vez de pasar toda la información para el caso en que un script CGI la necesite, la especificación provee una API para que el programa CGI pida la información.

La desventaja que tiene es que cada vez que hay que recompilar la DLL, hay que reiniciar el servidor. Una opción que no es práctica en tiempo de desarrollo.

8.10.2 FastCGI

FastCGI extiende las capacidades de CGI removiendo la sobrecarga asociadas a los programas CGI. Como CGI, FastCGI es un sistema no propietario en que corre en *background* manejando requerimientos bajo demanda [Dwight97].

Como CGI, FastCGI es independiente del lenguaje, es decir, los scripts FastCGI se pueden escribir en cualquier lenguaje. Además, FastCGI hace uso de programación distribuida, permitiendo que, en vez de servir documentos y ejecutar scripts CGI en una sóla máquina, se pueda distribuir la carga entre varias máquinas.

8.10.3 NSAPI

NSAPI (Netscape Server Application Programming Interface) se creó para aliviar algunas de las limitaciones del CGI. Para aliviar la sobrecarga generada por la creación de procesos CGI, Netscape produjo una interfaz entre el servidor Netscape y las aplicaciones back-end usando linkedición dinámica y objetos compartidos [Dwight97].

8.10.4 HTML Dinámico

HTML dinámico no es un lenguaje distinto de HTML *per se*, sino el lenguaje que se obtiene al agregar a HTML la capacidad de scripting con JavaScript o Visual Basic Script, junto con las hojas de estilo en cascada que hacen que el contenido de los documentos HTML deje de ser estático para convertirse justamente en dinámico. De esta manera, el contenido de un documento HTML cambiará en respuesta a eventos del usuario y/o condiciones internas en el estado de sus controles [Gulbransen98].

Enlazado Dinámico de Datos

Una característica interesante del HTML dinámico es lo que constituye el *enlazado dinámico de datos*. Esta característica, que fue aprovechada en alguna versión prototipo de *Querando!*, se explica en esta sección.

El principal problema del CGI (sección 8.8) está relacionado con su "falta de estado" (*stateleness*), lo cual hace que se tengan que usar elementos (input hidden) para simular variables de estado entre distintas sesiones. Además, como al usuario se le deben presentar los datos en forma parcial, para evitar el "information glut", con cgi se deben hacer muchos viajes de datos entre el cliente y el servidor, lo cual consume recursos computacionales en el servidor (lo cual presenta un problema de escalabilidad de las soluciones) y ancho de banda, y, además, hace esperar al usuario.

Las extensiones de HTML dinámico constan de tres atributos que hacen posible especificar dónde se obtendrán los datos (`DATASRC`), qué columna de datos es la que se considera (`DATAFLD`) y el formato de los datos que se van a recuperar (`DATAFORMATAS`) [Gulbransen98, p. 245–253].

Entonces, la mayoría de los problemas asociados al CGI (falta de escalabilidad, transacciones de ida y vuelta en el servidor, y complejidad creciente en el desarrollo de los guiones) se pueden solucionar con el enlazado de datos.

El enlazado de datos por parte del cliente nos permite seleccionar una fuente de datos que se situarán en una página de la red, y entonces se sitúan automáticamente los datos en la página. Así, se logra una mayor simplicidad ya que no hay que usar al servidor para enviar contenidos.

El enlazado de datos con HTML tiene, sin embargo, un requisito importante, además de ser utilizado con un cliente que lo admita. El cual es que los datos estén en formato

tabular [Gulbransen98, p. 254]. La *forma tabular* significa que los datos pueden representarse como filas y columnas, y que se puede acceder a cada columna individualmente.

El enlazado de datos, entonces, permite, con muy poco código HTML, mostrar el contenido de tablas en archivos de texto y de bases de datos conectadas con ODBC [Gulbransen98, pp. 255–287].

8.10.5 Java

Java es un lenguaje orientado a objetos desarrollado por Sun Microsystems [Bigus98, Weber97]. Originalmente, Java se diseño para sistemas embebidos (hornos a microondas, teléfonos celulares, video cassetteras, etc.); sin embargo, se lo redirigió para Internet por cuestiones comerciales.

Java es independiente de la arquitectura, le permite a un usuario recibir software desde un sistema remoto y ejecutarlo en su máquina local independientemente del hardware y sistema operativo que posea.

A diferencia de los lenguajes tradicionales, el código fuente no se traduce a instrucciones de máquina para una plataforma en particular, sino que se traduce a una forma intermedia llamada *bytecodes*. Estos bytecodes pueden ejecutarse en cualquier máquina que implemente una Máquina Virtual Java (JVM). Esta portabilidad es quizá la característica más atrayente de Java.

La naturaleza portable e interpretada de Java impacta en su performance. Mientras que la performance del código Java interpretado es mejor que la de los lenguajes interpretados y suficientemente rápida para aplicaciones interactivas, es más lenta que los lenguajes tradicionales donde el código fuente se traduce directamente a código de máquina para una plataforma en particular. Para mejorar la performance han surgido los compiladores *Just-In-Time* (JIT); un compilador JIT corre concurrentemente con la JVM y determina cuáles son las porciones de código más ejecutadas, las que son compiladas *on the fly* a código de máquina.

La portabilidad de los bytecodes es lo que permite a Java ser transportado a través de una red y ejecutarse en el browser. Un *applet Java* es un pequeño programa Java diseñado para incluirse en un documento HTML [Weber97]. Los tags HTML especifican el nombre del applet Java y su URL. Cuando un web browser java-enabled muestra un documento web conteniendo un tag applet, los bytecodes Java se bajan del sitio especificado y la JVM ejecuta los bytecodes. Los applet Java, de esta manera, permiten que las páginas web contengan gráficos animados y contenido interactivo.

Java, como lenguaje orientado a objetos, toma muchos rasgos de C++ y Smalltalk. Está caracterizado como un C++ más seguro, ya que no maneja punteros y la administración de la memoria no es hecha por el programador sino por el lenguaje mismo.

8.10.6 JavaScript

JavaScript (JS) es un lenguaje de scripting (de guiones) desarrollado en Netscape. JS es admitido tanto por IE 4.0 como por Communicator y provee una forma de hacer guiones para páginas de HTML estático y para controlar elementos de HTML dinámico.

La figura 8.10.6 muestra un guión rudimentario en JS cuyo único función es imprimir un cartel en la ventana del browser.

```
<SCRIPT language = "JavaScript">
        document.write( "Hola, mundo!" );
</SCRIPT>
```

Figura 8.5: El código JavaScript va entre los tags `<SCRIPT language='JavaScript'>` y `</SCRIPT>`.

El código JS se inserta en el código HTML de una página entre los tags `<SCRIPT language='JavaScript'>` y `</SCRIPT>`; el atributo "language" especifica que el lenguaje en el que se escribirá el guión es JS, otra opción es usar el lenguaje de scripting de Microsoft, Visual Basic Script (ver sección 8.10.7). Cuando la página se carga el código JS es interpretado por el browser ya sea debido a la generación de un evento de usuario o a la presencia de un programa secuencial. El caso de un programa secuencial es el de la figura 8.10.6 1, donde dicho programa se ejecutará cuando se cargue la página que lo contuviera. Por otro lado, un evento corresponderá a una acción del usuario sobre algún elemento de la página, como ser un movimiento del ratón, la edición de algún elemento de un formulario, etc. Esto último es lo que más su usó en este trabajo, es decir, usar JS para validar campos en formularios [Gulbransen98].

Además, JS junto con HTML dinámico permite tener elementos dinámicos en la interfaz de usuario; por ejemplo, tener elementos de un formulario que se hacen o no visibles bajo demanda.

El lenguaje JS posee los siguientes elementos [JS99]: sentencias, bloques, comentarios, expresiones y variables (de tipo string, numérico, booleano, objeto o null; JS es un lenguaje de tipado dinámico), funciones, objetos (los cuales pueden ser built-in –para denotar propiedades de la página– o definidos por el usuario), matrices, eventos y estructuras de control (secuencia, if, if-else, while, for. do-while).

En este trabajo se generaron, mediante CGI, guiones de JS en forma dinámica, ya que el contenido de éstos iba a depender de los datos que se manejaran en dicho momento. La ventaja de este enfoque permite aligerar la carga en el servidor, ya que parte del procesamiento de la interfaz de usuario se realiza en el cliente. Por otro lado, la desventaja de este enfoque reside en que, para ejecutar esta aplicación, hace falta poseer un browser de versión relativamente reciente (IE4.0 o superior).

También, existe una extensión de Microsoft al lenguaje JavaScript llamada *JScript*.

8.10.7 VBScript

Visual Basic Script es el equivalente de JScript. VBScript es un lenguaje propietario de Microsoft. La funcionalidad y características de VBScript son equivalentes a las de JScript. La diferencia de VBScript con JavaScript es que sólo es soportado en los browsers de Microsoft (a saber, el Internet Explorer) [Gulbransen98].

```
<html><head><title>Example</title>
<body>
<?php echo "Hi, I'm a PHP script!"; ?>
</body></html>
```

Figura 8.6: Un ejemplo de un script PHP.

8.10.8 PHP

PHP, que quiere decir "PHP: Hypertext Preprocessor", es un lenguaje de scripting embebido en HTML [Bakken99]. Mucha de su sintaxis es tomada de C, Java y Perl con algunas características propias. El objetivo del lenguaje es permitirle a los desarrolladores web escribir rápidamente páginas generadas dinámicamente.

Un ejemplo muy simple de código PHP, tomado de [Bakken99], se ve en la figura 8.10.8.

En PHP, a diferencia de C o Perl, en vez de escribir un programa con montones de comandos para imprimir en la salida código HTML, se escribe un script HTML con código embebido (en el caso de la figura 8.10.8, para imprimir un texto). El código PHP se enmarca en tags especiales de principio y fin que permiten entrar al browser en *modo PHP*. Lo que distingue a PHP de algo client-side como JavaScript es que el código es ejecutado en el servidor. Si usted fuera a tener un script similar al de la figura 8.10.8 en su servidor, el cliente recibiría los resultados de ejecutar tal script, sin ninguna manera de determinar cuál sería dicho código. En el nivel más básico, PHP puede hacer cualquier cosa que un programa CGI pueda hacer, como colectar datos de un formulario, generar contenido de páginas dinámico y enviar y recibir cookies; además, PHP tiene soporte para un amplio rango de bases de datos.

Con respecto a las plataformas que soportan PHP, éste se instala en el servidor, el cual puede ser Unix o Windows 95/98/NT; PHP se puede instalar también sobre IIS y/o PWS [Bakken99, p. 57].

PHP tiene tipos como numéricos, string y lógicos, además de tener objetos y las estructuras de control habituales (secuencia, selección, bucles y excepciones) y funciones.

8.10.9 Comparación entre Opciones

La ventaja de CGI es que los guiones se pueden programar en cualquier lenguaje de programación; por lo tanto, se puede aprovechar el soporte de bases de datos, objetos, etc del lenguaje que se usara.

De las opciones analizadas, tal vez Java sea la más flexible ya que permite tener procesamiento en el cliente (por medio del uso de un applet) sumado al procesamiento en el servidor (por medio de una aplicación stand-alone) y a ambas conectadas mediante un socket [Weber97].

En cuanto a VBScript, este lenguaje es tan poderoso como JavaScript aunque tiene portabilidad nula a otros browsers distintos a Internet Explorer [Gulbransen98].

PHP es muy poderoso pero hace procesamiento server-side, lo que hace que aumente

la carga de procesamiento sobre el servidor. Es comparable a CGI, sólo que el código se pone en la página HTML.

8.11 Resumen

En este capítulo se describió qué es Internet, la World Wide Web, los protocolos de Internet relacionados con los agentes estudiados y con el agente implementado. También, se describieron las tecnologías para construir aplicaciones basadas en la Web.

Capítulo 9

Motores de Búsqueda y Trabajos Relacionados

En este capítulo se hace una descripción de la interfaz con servicios de búsqueda de la web. Además, se enumeran los trabajos relacionados; en particular, se discutem los agentes de filtrado y manejo de información.

9.1 Motores de Búsqueda

Para obtener información en la Web es más fácil usar los recursos de los "Web Crawlers" ya existentes en la Red. A continuación se explica como es la interfaz usando CGI con los "Web crawlers" Altavista y Lycos.

9.1.1 Altavista

El servicio de búsqueda *Altavista* ayuda al usuario a encontrar documentos en World Wide Web. Su funcionamiento es el siguiente: el usuario le indica al servicio de búsqueda qué es lo que está buscando por medio del tipeo de palabras claves, frases o preguntas en la caja de búsqueda. El servicio de búsqueda le responde al usuario dándole una lista de todas las páginas de la Web en su índice relacionada con dichos tópicos. Los contenidos más relevantes aparecerán primero entre los resultados. De acuerdo al Help del servicio de búsqueda, los índices del Altavista son una gran colección creciente y ordenada de páginas de la Web y páginas de grupos de discusión de todo el mundo. Dicho índice crece día a día gracias a que Altavista tiene "arañas" (*crawlers*) que recorren la Web buscando nuevas páginas.

Existen dos formas de especificar una búsqueda: simple y avanzada.

Búsqueda Simple

El usuario sólamente especifica una serie de palabras, una combinación de letras y números, separadas por caracteres tabuladores y espacios. Si se desea encontrar una frase exacta se deben usar comillas delimitando la frase que se ingresa en la caja de búsqueda. Por ejemplo, si se quisiera encontrar una canción de Miguel Mateos, se podría poner: *"sólo busco lo que vos querés"*.

También se pueden crear frases usando puntuación o caracteres especiales como guiones, subrayados, comas, barras o puntos. Por ejemplo, *1-800-999-9999* en vez de *1 800 999 9999*. El efecto de los guiones es unir a los números como una frase.

Otra característica es que cada vez que el usuario no esté seguro de lo que busca use letras minúsculas para especificar las claves de búsqueda; ya que, cuando se usan letras minúsculas, el servicio de búsqueda encuentra resultados en minúsculas tanto como en mayúsculas, mientras que cuando se utilizan letras mayúsculas, el servicio de búsqueda sólo encuentra claves en mayúscula. Por ejemplo, si el usuario especifica *paris* hallará *Paris*, *paris* y *Paris* en sus páginas resultado; sin embargo, cuando se tipea *Paris* sólo se hallará *Paris* en los resultados. La recomendación es que cada vez que el usuario tenga dudas debe usar minúsculas.

Otra característica es la inclusión o exclusión obligada de palabras claves en la búsqueda. Para asegurar que una palabra específica siempre será incluida en la búsqueda, se debe poner el símbolo más (+) delante de la clave en la caja de búsqueda; por otro lado, para asegurar que una palabra específica siempre será excluida del tópico de búsqueda, se debe poner un signo menos (−) delante de la clave en la caja de búsqueda. Por ejemplo, para encontrar recetas con *oatmeal* pero sin *raisins*, poner *recipe cookie +oatmeal -raisin*.

También, se puede expandir la búsqueda usando asteriscos. Tipeando al final de una clave de búsqueda, se puede buscar una palabra con múltiples terminaciones. Por ejemplo, poner *wish**, para hallar *wish*, *wishes*, *wishful*, *wishbone* y *wishy-washy*. Además, Altavista no sólo busca texto, en la siguiente tabla se detallan otras maneras en las que se puede buscar en la Web.

Búsqueda Avanzada

La búsqueda avanzada es para búsquedas muy específicas y no para búsquedas generales. De acuerdo a la ayuda de Altavista, casi cualquier cosa que el usuario necesite buscar puede ser encontrada rápidamente y con mejores resultados usando la caja standard de búsqueda, donde el servicio de búsqueda de AltaVista ordena los resultados ubicando los de contenido más relevante primero. Sin embargo, si el usuario necesitara en un cierto intervalo de fechas o si tuviera que hacer una búsqueda Booleana compleja se podría usar esta herramienta.

Las palabras claves de búsqueda trabajan de la misma manera en tanto en la búsqueda standard como en la avanzada. También se puede elegir buscar en Usenet o refinar la búsqueda usando cualquiera de las dos herramientas. Las características de inclusión (+) y exclusión (-) no están disponibles en la búsqueda avanzada. Sin embargo, se pueden usar los comandos Booleanos para personalizar la búsqueda. Otra diferencia es que se puede elegir ver los resultados sin el ordenamiento por ranking.

El usuario puede crear relaciones específicas entre las palabras o frases de búsqueda usando los comandos *OR*, *NOT*, *AND*, *NEAR* en la sección de operadores booleanos debajo de la caja de búsqueda.

La tabla 9.1 indica la sintaxis y semántica de los operadores búsqueda.

Interfaz con Altavista

Altavista[1] provee un índice de sitios Web y grupos de noticias.

[1] `http://www.altavista.com/`

1. *AND* (&): Encuentra sólo documentos conteniendo palabras o frases específicas. Por ejemplo, *Mary AND lamb* hallará documentos con las palabras *Mary* y *lamb*.
2. *OR* (—): Encuentra documentos con al menos una de las palabras o frases especificadas. *Mary OR lamb* encuentra documentos conteniendo *Mary o lamb*. Los documentos pueden contener una de las palabras o ambas.
3. *NOT* (!): Excluye los documentos conteniendo la palabra o frase especificada. *Mary AND NOT lamb* encuentra documentos con *Mary* pero no conteniendo *lamb*.
4. *NEAR* (˜): Encuentra documentos conteniendo las dos palabras o frases especificadas pero a lo sumo 10 palabras de distancia una de otra. *Mary NEAR lamb* encontraría *the nursery rhyme* pero problamente no encontraría documentos religiosos o navideños.

Tabla 9.1: Resumen de comandos avanzados de Altavista.

```
<H1>Search Altavista</H1>
<FORM method = get
      action = "http://www.altavista.digital.com/cgi-bin/query">
<INPUT Type=hidden name=pg value=q>

<B>Search
<SELECT name=what>
<OPTION value=web SELECTED>The Web
<OPTION value=news>Usenet
</SELECT>
and Display the Results
<SELECT NAME=fmt>
<OPTION value="." SELECTED>in Standard Form
<OPTION value=c>in Compact Form
<OPTION value=d>in Detailed Form
</SELECT>
</B>

<INPUT type=text name=q size=55 maxlength=200 value="">
<input type=submit value="  Submit  ">
<BR>
</FORM>
```

Figura 9.1: Formulario Genérico de Búsqueda a Altavista.

La figura 9.1.1 de código HTML muestra como agregar la opción de búsqueda en una página HTML [Dwight97, p. 388].

Se puede observar que como el método de invocación del script de CGI correspondiente es GET, y si se ingresan una búsqueda en la Web en formato compacto, el llamado tendría el siguiente aspecto:

```
what=web&fmt=c&pg=g&q=<string de búsqueda&stq=<número de resultados>
```
(9.1)

Donde dicho string es un conjunto de cinco parámetros separados por ampersands. `what=web` le dice a Altavista que busque en su índice de páginas Web (la otra opción posible es `what=news` para que busque en los grupos de noticias). `fmt=c` dice que la salida sea producida en formato compacto, para facilitar el "parsing" (las otras dos opciones son `fmt=.`, para la forma standard y `fmt=d`, para el formato en forma detallada). `pq=q` indica que se está haciendo una consulta usando la sintaxis básica. `q=` identifica el string de búsqueda, el cual debe estar codificado en el formato URL, codificando espacios y otros caracteres especiales. Finalmente, `sqt=` le dice a Altavista a partir de cuál item del resultado de la consulta debe comenzar al retornar sus datos; Altavista retorna diez resultados por vez, y `sqt=` permite obtener los resultados después de los primeros diez [Weber97, p. 431].

La página HTML devuelta por una consulta simple Altavista contiene los resultados como saltos entre los marcadores HTML `<PRE>` y `</PRE>` [Weber97]. Una página tal se puede apreciar en la figura 9.1.1. Allí, se aprecia la siguiente información:

1. El buscador devuelve 10 resultados por vez (la figura está editada por razones de espacio).
2. Se aprecia el número de resultado en negrita (marcadores `<B>` y `</B>`) del resultado i-ésimo.
3. La URL correspondiente al enlace del resultado i-ésimo (marcador `<A>`).
4. El texto correspondiente al enlace (hasta el marcador `</A>`).
5. Un pequeño resumen del contenido del enlace.

9.1.2 Deja

Deja (`http://deja.com/`) es un servicio de búsqueda de la WWW para grupos de noticias. El usuario, como en todos los casos, especifica palabras claves de búsqueda y se le devuelve un conjunto de enlaces.

La interfaz con este buscador es la siguiente:

`http://deja.com/dnquery.xp?QRY=<consulta>`.

Por ejemplo, la consulta típica es `http://deja.com/dnquery.xp?QRY=highlander`.

El formato para recuperar páginas más allá de las primeras respuestas no parece responder a un formato no volátil. Por ejemplo, para la segunda página del query anterior la línea de comandos del explorador fue:

```
http://x69.deja.com/dnquery.xp?search=next&
LNG=english&IS=highlander&ST=QS&offsets=db2000p8x%0225&
CONTEXT=975428334.1680605209
```

```
<pre>

<b>1.</b>
<a href="http://www.highlanderweb.co.uk/">Highlander Web Magazine  </a>
3-Feb-2000 What's New. Tell a Friend.

<b>2.</b>
<a href="http://www.seventh-dimension.simplenet.com/">Seventh Dimension -- High</a>
1-Feb-2000 Archive for fiction writte

<b>3.</b>
<a href="http://members.aol.com/VFairbanks/index3.html">Highlander the series and</a>
20-Feb-2000 Highlander the Series star

                              . . . . . .

<b>9.</b>
<a href="http://www.geocities.com/Heartland/Estates/2217/ravenring.html">Join the Highlander The R</a>
24-Jun-1999 Welcome to the Highlander:

<b>10.</b>
<a href="http://www.geocities.com/Area51/9198/links/highland1.html">The Ultimate Highlander L</a>
3-Feb-2000 Info --&gt; The Ultimate List

</pre>
```

Figura 9.2: Resultado de una consulta a Altavista.

9.1.3 Excite

Excite (`http://search.excite.com/`) es otro servicio de búsqueda para páginas web. La sintaxis para la búsqueda simple es

`http://search.excite.com/search.gw?c=web&search=<consulta>`.

Entonces, para el caso de la consulta *highlander*, la URL de consulta es:

`http://search.excite.com/search.gw?c=web&search=highlander`.

Para obtener páginas más allá de las primeras diez respuestas, el formato es el siguiente:

```
http://search.excite.com/search.gw?
s=highlander&c=web&showSummary=true&start=10
&perPage=10&lang=en&next=Next+Results
```

Otro formato de consulta que devuelve los resultados ordenados por sitio es:

```
search.gw?s=highlander&c=web&showSummary=false&start=0&perPage=10&sort=site
```

Donde `s=` indica cuál es la consulta (en este caso *highlander*, `c=` donde vamos a buscar (*la web*), `showSummary` indica si se quiere una descripción de cada una de las respuestas halladas (*true*) o no se quiere (*false*). `start=` indica a partir de que hit se quieren los resultados. `perPage=` indica cuantos resultados por página se desean. `lang=` indica el idioma. Y `next=`, si los resultados son los siguientes o los previos.

En el caso de Excite, existe una opción para que los resultados sean devueltos en formato resumido, sin descripción de párrafo. En este caso, la lista de resultados de la consulta comienzan a partir del marcador HTML `<UL>` (indicando lista); a su vez, cada item está indicado con el marcador HTML `<LI>`.

9.1.4 Google

Google es un prototipo de un motor de búsqueda de gran escala que hace un uso exhaustivo de la estructura presente en el hipertexto [Brin2000]. Google está diseñado para recorrer e indexar la Web eficientemente y producir resultados de búsqueda en forma eficiente.

Google es un prototipo con una base de datos de texto completo y los enlaces de por lo menos 24 millones de páginas que está disponible en la URL `http://www.google.com/`.

En Google, el método para darle pesos a las páginas web está basado en el principio de las citas académicas (donde un artículo es más importante cuanto más referencias a él se puedan hallar en la literatura). Google, cuenta la cantidad de enlaces de otras páginas a una página dada; esto le da una aproximación de la importancia de la página o de la calidad de la misma [Brin2000].

La interfaz con Google se dedujo a partir del código HTML de la página devuelta por su portal (figura 9.1.4)

En base a los mismos argumentos que con la interfaz a Altavista, la consulta a Google hay que hacerla a la URL `http://www.google.com/intl/en/search` y hay que usar la siguiente línea de comandos:

```
?q=<consulta>
```

ya que las opciones de idioma parecen ser accesorias.

Para recuperar resultados más allá de los diez primeros hay que usar el formato:

```
?q=<consulta>&start=10&sa=N
```

donde `start=10` indica que se quieren los resultados del 11 al 20. Análogamente,

```
?q=<consulta>&start=20&sa=N
```

recuperará los resultados del 21 al 30.

Los ejemplos con la consulta *highlander* son los siguientes:

1. http://www.google.com/search?q=highlander
2. http://www.google.com/search?q=highlander&hl=es&lr=&safe=off&start=10&sa=N
3. http://www.google.com/search?q=highlander&hl=es&lr=&safe=off&start=20&sa=N

9.1.5 Hotbot

Hotbot es otro servicio de búsqueda disponible en la dirección `http://hotbot.lycos.com/`.

La interfaz para hacer una búsqueda con Hotbot es la siguiente:

```
http://hotbot.lycos.com/?MT=<consulta>
```

Un ejemplo de búsqueda con la clave *highlander* es:

```
http://hotbot.lycos.com/?MT=highlander
```

```
<html>
<head>
<title>Google</title>

<FORM action="http://www.google.com/intl/en/search" method=get name=f>

<select name=meta>
    <option selected value="lr=&amp;hl=es">Todos los idiomas
    <option value="lr=lang_de&amp;hl=de">Aleman
    <option value="lr=lang_zh-CN&amp;hl=zh-CN">Chino (simplificado)
    <option value="lr=lang_zh-TW&amp;hl=zh-TW">Chino (tradicional)
    <option value="lr=lang_ko&amp;hl=ko">Coreano
    <option value="lr=lang_da&amp;hl=da">Danes
    <option value="lr=lang_es&amp;hl=es">Espanol
    <option value="lr=lang_fi&amp;hl=fi">Finlandes
    <option value="lr=lang_fr&amp;hl=fr">Frances
    <option value="lr=lang_nl&amp;hl=nl">Holandes
    <option value="lr=lang_en&amp;hl=en">Ingles
    <option value="lr=lang_it&amp;hl=it">Italiano
    <option value="lr=lang_ja&amp;hl=ja">Japones
    <option value="lr=lang_no&amp;hl=no">Noruego
    <option value="lr=lang_pt&amp;hl=pt">Portugues
    <option value="lr=lang_sv&amp;hl=sv">Sueco
    <option value="">Personalizado
</select>

<INPUT name=btnG type=submit value="Busqueda en Google">

<input type=text value="" framewidth=4 name=q size=55>
</FORM>

</BODY>
</HTML>
```

Figura 9.3: Formulario Google

```
<FORM ACTION=http://www.lycos.com/srch/ METHOD=GET>

<INPUT TYPE=hidden NAME=lpv VALUE=1>
<INPUT TYPE=hidden NAME=loc VALUE=searchhp>
<INPUT TYPE=text NAME=query size=25>

<input type=submit value="Enviar consulta">
```

Figura 9.4: Formulario Lycos

9.1.6 Infoseek

Infoseek es otro servicio de búsqueda disponible en la URL `http://www.go.com/`.

La interfaz para usar este servicio de búsqueda es:

```
http://www.go.com/Split?pat=go&col=WW&qt=<consulta>
```

donde `col=WW` indica que se quiere buscar en la WWW y `qt=<consulta>` corresponde a la especificación de los términos de búsqueda.

El ejemplo con el término de búsqueda *highlander* es:

```
http://www.go.com/Split?pat=go&col=WW&qt=highlander
```

9.1.7 Lycos

Lycos es otro servicio de búsqueda que permite hacer búsquedas por clave en la World Wide Web. La URL de Lycos es `http://www.lycos.com/`.

La interfaz con Lycos fue deducida a partir del código HTML de la página devuelta por su portal (figura 9.1.7).

En base a los mismos argumentos que con la interfaz a Altavista, la consulta a Lycos hay que hacerla a la URL `http://www.lycos.com/srch` y hay que usar la siguiente línea de comandos:

```
?lpv=1&loc=searchhp&query=<consulta>
```

9.1.8 Yahoo

Yahoo es otro motor de búsqueda de la Web y está disponible en la URL `http://www.yahoo.com/`.

El formato de la consulta a Yahoo es:

```
http://search.yahoo.com/bin/search?p=<consulta>
```

9.2 Robot Netiquette

La *Netiquette* consiste de la etiqueta de la red o, dicho de otra manera, de las buenas maneras de Internet [Gach96]. Cuando se habla de *robot netiquette* hablamos de las buenas maneras de los programas de Internet.

9.2.1 Robots WWW

Un *robot* es un programa que automáticamente recorre al estructuras de hipertexto de la Web recuperando recursivamente todos los documentos referenciados. Los web browsers normales no son robots porque están operados por humanos y no recuperan documentos referenciados (salvo imágenes) [Koster96a].

Otros nombres para los robots son *web wanderers* (vagabundos), *crawlers* o *spiders* (arañas) [Cheong96, Koster96a].

Otras funciones de los robots son: indexamiento de sitios, validación de HTML, validación de enlaces (búsqueda de enlaces muertos) y espejado de sitios.

9.2.2 Etiqueta

En esta sección se describen los recaudos que hay que tener a la hora de construir un agente robot que vaya a interactuar con otros agentes (programas) en la WWW, los cuales están basados en [Dwight97, Koster93, Koster96a, Koster96b].

1. Ir sólamente detrás de los archivos valiosos. Si no tiene sentido para el robot recuperar archivos de imágenes, películas o sonidos, entonces no hay que hacerlo.

2. Recuperar sólo la porción del archivo HTML que se necesita.

3. Configurar al robot para validar las URLs. Muchas URLs pueden no tener las barras al final para indicar que se trata de un directorio en vez de un archivo. Asegurarse que el robot no trate con scripts CGI.

4. No hacer que el robot tome todos los datos de una vez. Quizá muchos servidores de la web no escalen a la velocidad de pedidos de un programa; cuando un usuario navegando por la web pedirá una página cada dos minutos, un programa puede pedir cientas en el mismo lapso. Por ello, tratar de que el programa pida de a una página por minuto (o cada 30 segundos) por servidor. Sin embargo, a veces esto puede ser demasiado lento o demasiado rápido dependiendo del servidor.

5. Hay que identificar el nombre del robot en el campo *User-agent* de HTTP.

6. Hay que anunciar que se va a liberar el robot mandando un mensaje a `comp.infosystems.www.providers`.

7. Si un servidor tiene un archivo `/robots.txt`, tratar de someterse a él. El uso de este archivo es parte del estándar para exclusión de robots. En este archivo se especifican qué robots pueden acceder al sitio y en qué directorios. Primero, el robot de uno debe registrarse en `http://www.submit-it.com/`. Un ejemplo de un archivo tal se ve en la figura 9.5.

```
# /robots.txt file for http://webcrawler.com/
# mail webmaster@webcrawler.com for constructive criticism

User-agent: webcrawler
Disallow:

User-agent: lycra
Disallow: /

User-agent: *
Disallow: /tmp
Disallow: /log
```

Figura 9.5: Ejemplo de un archivo `/robots.txt`.

Las primeras dos líneas que empiezan con '#' son comentarios. El primer párrafo especifica que el robot llamado *webcrawler* puede ir donde quiera. El segundo párrafo indica que el robot llamado *lycra* no tiene permitidas aquellas URLs relativas empezando con '/'. Como todas las URLs relativas en un servidor comienzan con '/', esto significa que el sitio le está prohibido. El tercer párrafo indica que todos los robots no deberían visitar las URLs comenzando con emph/tmp o */log*. Nótese que '*' es un carácter especial, significando *cualquier otro agente de usuario*.

9.3 Trabajos Relacionados

En esta sección, se describirán agentes (o aplicaciones) de manejo de información, en especial acceso y filtrado, en Internet que se hallaron en la literatura.

9.3.1 Filtro de Correo Electrónico

El enfoque actual en las interfaces de usuario se denomina metáfora de *manipulación directa*, lo que requiere que el usuario inicie todas las tareas explícitamente y monitoree todos los eventos. Una variante consiste en usar a los agentes como entidades de software con poder delegado para realizar tareas en favor de sus usuarios [Cheong96, p. 6]. Un ejemplo de esta visión está dada por el agente de filtrado de correo electrónico de Pattie Maes y cía. [Lashkari97].

En el artículo de [Lashkari97] se describe un agente de interfaz para el dominio de correo electrónico para la aplicación comercial Eudora.

La interfaz de cada usuario aprende por medio de continuamente "mirar sobre el hombro" del usuario a medida que éste realiza acciones. El agente de interfaz monitorea las acciones del usuario durante largos períodos de tiempo, encuentra patrones recurrentes y ofrece automatizarlos.

Cuando el agente se inserta en un ambiente multiagente es capaz de aprender no sólo por medio del entrenamiento de parte del usuario sino aprovechando el contacto con otros

agentes. Se identifican dos problemas: competencia y confianza. La *competencia* se refiere al hecho de cómo el agente obtiene el conocimiento para decidir cuándo y cómo ayudar al usuario. La *confianza* se refiere a cómo asegurar que el usuario se sienta confortable delegando tareas a un agente. Maes resuelve el problema con un enfoque de *machine learning* (lo que concuerda con Cheong [Cheong96, p. 6]), donde el agente aprende los hábitos del usuario a través de la interacción en el tiempo. El agente gradualmente adquiere su competencia por medio de los cuatro siguientes métodos:

- *Observar e imitar al usuario*: Esto se logra mediante un método de captura de los patrones de comportamiento del usuario llamado *Razonamiento Basado en Memoria* (sección 5.5), en el cual el agente de interfaz monitorea las acciones del usuario, buscando patrones recurrentes y buscando automatizarlos.

- *Recibir feedback positivo o negativo del usuario*: La evaluación por parte del usuario de la salida del robot, hace que éste vaya aprendiendo de sus errores y reforzando la conducta que lo llevó a un acierto.

- *Recibir instruccciones explícitas del usuario*: El enfoque de machine learning tiene la ventaja de poder hacer aprendizaje por medio de ejemplificación de la conducta deseada. Por otro lado, sufre del problema de tener una 'curva de aprendizaje' lenta; esto es, el agente requiere de un número suficientemente grande de ejemplos para hacer predicciones precisas. Esto lleva a tener que usar un enfoque de programación mediante reglas. En este caso, el usuario puede dar conocimiento inicial al agente programando un *script*[Lashkari97]. Esta solución tiene sus problemas, como ser que el usuario debe aprender un lenguaje formal; y, en el caso en que el agente no use machine learning, mantenerlas en el tiempo; este ha sido estudiado en la literatura y se conoce como *cuello de botella de adquisición de conocimiento*[2][Clancey93, Ford93].

- *Pedir consejo a otros agentes*: También en [Lashkari97], se describe una estrategia alternativa a la del ítem anterior. En este caso, el agente obtendrá su conocimiento inicial preguntando a otros agentes. Pattie Maes *et al.* identifican dos tipos de colaboración:

  - *Comunicación basada en la desesperación*: Un agente se comunica con otros cuando no tiene suficiente experiencia para hacer una predicción.
  - *Comunicación exploratoria*: Por otro lado, cuando el agente tiene experiencia, se comunica con otros agentes para buscar al mejor colaborador.

Los resultados obtenidos por [Lashkari97] son que el nivel de precisión en las predicciones del agente que colabora siempre es mayor o igual al del agente trabajando sólo.

9.3.2 Filtro de Noticias de Bigus

La aplicación *NewsFilter* desarrollada en [Bigus98] usa un agente inteligente basado en redes neuronales de backpropagation (sección 5.6.6) y mapa de Kohonen (sección 5.6.8) para filtrar artículos en servidores de noticias.

El agente está basado en el protocolo NNTP (sección 8.3.4). A su vez, este agente provee tres tipos de filtrado de artículos de newsgroups [Bigus98]:

[2]Knowledge acquisition bottleneck.

1. *Filtrado por clave*: El usuario especifica palabras de clave de búsqueda y los artículos son "rankeados" de acuerdo a la cantidad de *matches* de las mismas.

2. *Filtrado por clustering*: Un conjunto de artículos es clasificado mediante un mapa autoorganizante de Kohonen. Cuando se presenta un nuevo documento, el sistema determina a cuál clase pertenece éste y expresa un juicio sobre el mismo.

3. *Filtrado por feedback*: Usa una red de backpropagation para construir un modelo de predicción de para puntuar artículos basado en un criterio de relevancia.

9.3.3 Harvest

Harvest es un conjunto integrado de herramientas para recolectar, extraer, organizar, buscar, almacenar y replicar información relevante en Internet [Cheong96, Hardy96]. Los usuarios pueden personalizar a Harvest para procesar información en muchos formatos diferentes y también personalizar búsquedas en Internet.

Un objetivo clave de Harvest es proveer un sistema flexible que puede configurarse de varias maneras para crear índices, haciendo uso eficiente de los servidores de Internet, enlaces de redes y tamaño de los índices en los discos.

Las mediciones de estos autores indican que Harvest puede reducir la carga de los servidores en un factor de más de 6000, el tráfico en la red en un factor de 60, el requerimiento de espacio en disco en un factor de 40 cuando se construyen índices comparadas con otros sistemas como Archie, WAIS y el WWW Worm.

Harvest también le permite a los usuarios extraer información estructurada de muchos formatos de información diferentes y construir índices que permiten que estos atributos se referencien durante consultas (por ejemplo, buscar todos los documentos con una cierta expresión regular en el campo del título).

Harvest consiste de varios subsistemas. El subsistema Recolector *(Gatherer)* recolecta información de indexación (como claves, nombres de autor y títulos) de los recursos disponibles en sitios *proveedores* (como servidores FTP y HTTP). El subsistema *Broker* recupera información de uno o más recolectores, suprime información duplicada, incrementalmente indexa la información recolectada y provee una interfaz WWW (formulario CGI) para ella. El subsistema Replicador *(Replicator)* eficientemente replica brokers alrededor de Internet. Los usuarios pueden recuperar eficientemente información ubicada a través del subsistema de *Cache*. El Sistema de Registro de Harvest –*Harvest Server Registry (HSR)*– es un broker distinguido que mantiene información acerca de cada recolector, broker, cache y replicador en Internet.

9.3.4 Internet Learning Agent

El agente *Querando!* construido en este trabajo de grado hace consultas a buscadores en la Web, por lo tanto debe tener codificado el conocimiento para acceder a dichas fuente de información.

Mike Perkowitz y Oren Etzioni plantean que esta manera de trabajar no permiten que los sistemas basados en inteligencia artificial escalen al crecimiento de Internet [Perkowitz96]. En base a esto, estos autores plantean el sistema *ILA* (Agente de Aprendizaje de Internet) [Perkowitz96], el cual es capaz de aprender a modelar por si mismo el servicio de internet `whois`.

En sus experimentos, ILA comienza con conocimiento de los miembros locales de una facultad y es capaz de aprender modelos del personal de otras universidades en base a las relaciones de los primeros con los segundos.

9.3.5 Internet Softbot

Oren Etzioni y Daniel Weld han desarrollado una interfaz a la Internet que permite a los usuarios nuevos "localizar, monitorear y transmitir información en la red".

El *Internet Softbot*[3] [Cheong96, Etzioni97, págs. 10–11] usa un shell Unix [Wayar99] y la WWW para interactuar con rango amplio de recursos de Internet. Los robots físicos poseen sensores y efectores [Carter97]; en cambio, los sensores del Softbot son programas Unix, como archie, gopher, netfind, etc. Por otro lado, los efectores del robot son ftp, telnet, mail y comandos de manipulación de archivos.

El softbot es una interfaz que se comporta como un asistente personal (PDA). El usuario hará un requerimiento de alto nivel y el robot lo satisfará usando técnicas de *divide-and-conquer*. El requerimiento puede hacerse en forma de una sentencia de un lenguaje de primer orden o a través de un formulario en el caso de usuarios menos experimentados.

El cumplimiento del requerimiento del usuario se hace a través de un componente llamado *softbot planner*. Éste descompone los requerimientos de alto nivel en componentes más simples los cuales son comparados contra esquemas de acciones. A su vez, el planeador posee un modelo de los recursos de la Web y la manera de accederlos.

Según los autores, las ventajas de usar el softbot son [Etzioni97]:

- El agente provee una interfaz integrada y expresiva a la Internet.
- El robot elige, dinámicamente, qué facilidades usar para realizar su labor y en qué orden utilizarlas.
- El robot pasa de una facilidad a otra, dependiendo de la información recolectada, en tiempo de ejecución.

Entre las características de la interfaz, podemos hallar las siguientes [Etzioni97]:

- *Goal-oriented*: El requerimiento del usuario especifica *qué* quiere el usuario y el agente determina *cómo* satisfacerlo.
- *Charitable*: Un requerimiento no necesita ser especificado completamente; el agente descifra lo que falte.
- *Balanceada*: El agente balancea el costo de hallar la información por su cuenta contra la molestia de hacerle muchas preguntas al usuario.
- *Integrada*: El robot provee una interfaz uniforme a los servicios y recursos de la Internet.

[3] El término *Softbot* es un neologismo cuyo significado es *Robot de Software*

Ejemplo de Uso del Softbot

El ejemplo de uso de este agente que proponen sus autores es el siguiente: Supongamos, dicen, que encargamos al agente la tarea "enviar memos de presupuesto a Mitchell de CMU". El requerimiento se especifica de dos maneras, como una sentencia ó a través de la interfaz basada en un formulario; luego, debe determinar la dirección de mail de Mitchell (para lo que usara netfind); luego, debe determinar cuáles son los archivos que debe enviar (le preguntará al usuario); por último, enviará los archivos.

9.3.6 Inducción de Wrappers

Uno de los enfoques para lidiar con software de legado es la implementación de un *wrapper*, es decir, inyectarle código a un programa para que se comunique en un lenguaje dado. O de otra manera, poner una cáscara alrededor de un recurso para adaptarlo a una nueva interfaz [Genesereth97].

Los reservorios de enlaces HTML son una clase particular de recursos de información. Las páginas de contenido HTML –como catálogos, reservaciones de pasajes de aerolíneas, etc.– constituyen también recursos de información. La técnica de machine learning de *Querando!* es eficaz sólamente en el reconocimiento de documentos parecidos. Sin embargo, la traducción de dichos recursos a un formato más tradicional, digamos a algún formato de bases de datos (Microsoft Access, por poner el ejemplo más banal), surge como una alternativa.

En el artículo de Nicholas Kushmerick [Kushmerick98], se plantea la traducción automática de dichos recursos a KIF [Genesereth92, Labrow98].

9.3.7 InfoSleuth

En el trabajo de R. Bayardo Jr. *et al.* [Bayardo98] se describe el sistema multiagente *InfoSleuth*[4], como parte de un proyecto de MCC que tiene como objetivo *explotar y sintetizar nuevas tecnologías en un sistema unificado que recupere y procese información en una red de recursos cambiante.* Su origen es otro proyecto de MCC para integrar bases de información heterogéneas. Ahora, se pretenden adaptar a la WWW, donde hay más tipos y disponibilidad de fuentes de información. Su arquitectura de agentes está basada en una red de agentes que cooperan entre sí y se comunican por medio de imperativas KQML. El usuario especifica requerimientos y consultas sobre ontologías específicas via interfaces basadas en applets Java. Para representar dichas consultas se usan los dialectos del lenguaje de representación de conocimiento KIF y el lenguaje SQL. Las consultas se rutean por agentes de mediación y *brokerage* a agentes especializados en recuperación de datos de recursos distribuidos, y en la integración y análisis de los resultados. Los usuarios interactúan con esta red de agentes a través de *Java-enabled Web browsers* que se comunican con un agente inteligente y personalizado de usuario.

Los agentes promocionan sus servicios y procesan sus requerimientos basados en inferencias sobre conocomientos propios o rutean los requerimientos a otros agentes, basado en una ontología que describe cuál es el área de conocimiento de cada agente.

Así, la arquitectura del sistema multiagente está compuesta por los siguientes agentes [Bayardo98]:

[4]InfoSleuth quiere decir *sabueso de información.*

- *Agente de Usuario*: Constituye la interfaz del usuario con InfoSleuth. Usa conocimiento de las ontologías del sistema para ayudar al usuario a plantear consultas y mostrar sus resultados.

- *Agente de Ontología*: Provee conocimiento sobre las ontologías (modelos de dominio) y respuestas sobre las ontologías.

- *Agente Broker*: Recibe y almacena anuncios de todos los agentes de InfoSleuth y de sus habilidades Basado en esta información, responde a las consultas de los agentes así como rutea los requerimientos específicos.

- *Agente de Recursos*: Hace la traducción de las ontologías a los esquemas de bases de datos y lenguaje nativo a su recurso, incluyendo consultas.

- *Agente de Análisis de Datos*: Corresponde a los agentes de recursos especializados en métodos de análisis y minería de datos[5].

- *Agente de Ejecución de Tareas*: Coordina la ejecución de tareas para satisfacer las consultas. Identifica los recursos que las pudieran resolver, decide que parte resuelve cada uno y reemsambla los resultados.

- *Agente Monitor*: Controla las interacciones entre los agentes y provee una interfaz visual para mostrar la ejecución.

Los autores han aplicado su arquitectura a la solución de un problema de descubrimiento de conocimiento y una aplicación de cuidado de la salud.

En [Bayardo98] hacen crítica al modelo de KQML, el cual supone que las imperativas llegan en el orden en que fueron enviadas. De acuerdo a los autores, la naturaleza de TCP hace que no esto no sea cierto.

9.3.8 Lira

El trabajo de Marko Balabanović, Yoav Shoham y Yeogirl Yun [Balabanovic98] también tiene como objetivo reducir la sobrecarga de información en el usuario que produce la vastedad de la misma en la WWW. Su agente, llamado *Lira*, aprende a browsear la Internet en favor de un usuario. Cada día, el agente le presenta una selección de páginas Web interesantes. El usuario evalúa cada página, y dado este *feedback*, el sistema se adapta y trata de producir mejores páginas el día siguiente. De esta manera, el sistema es capaz de aprender un modelo de un usuario con un interés bien definido.

Los autores hicieron un experimento durante veinticuatro días, el sistema se comportó mejor que uno con selección al azar de páginas, y fue mejor que la elección humana en el cincuenta porciento de los casos.

Implementación

El sistema ha sido implementado en una mezcla de C++ [Brokken95, Kernigham85, Kruglinski97] y Python; corre en ciclos donde cada uno es de la forma:

[5]Minería de datos: Data mining.

1. Buscar en la Web, usando alguna heurística de búsqueda, tomando un cantidad de tiempo limitada.

2. Seleccionar las mejores p páginas para presentárselas al usuario, usando alguna heurística.

3. Para cada página presentada, recibir una evaluación del usuario.

4. Actualizar las heurísticas de búsqueda y selección de acuerdo a este *feedback*.

Extracción de Características

En Lira, la extracción de características de los documentos está basada en la representación vectorial (ref a IR). Cada documento se representa por un conjunto de palabras con su peso. Las palabras se obtienen luego de eliminar los tags HTML, eliminar stop words y aplicar stemming a las palabras restantes (usando el algoritmo de Porter [Frakes92b]). Los pesos de las palabras se obtienen utilizando un esquema TFIDF[6], donde el peso w_i de la palabra p_i en un documento D está dado por[7]:

$$w_i = \frac{(0.5 + 0.5\frac{tf(i)}{tf_{max}})(log\frac{n}{df(i)})}{\sqrt{\sum_{d_j \in T}((0.5 + 0.5\frac{tf(i)}{tf_{max}})^2(log\frac{n}{df(i)})^2)}} \tag{9.2}$$

donde $tf(i)$ es el número de veces que la palabra p_i aparece en el documento D (la *frecuencia de términos*), $df(i)$ es el número de documentos en la colección que contienen p_i (la *frecuencia de documento*), n es el número de documentos en la colección y tf_{max} es la máxima frecuencia de términos sobre todas la palabras en D.

Los autores calcularon la frecuencia de documento utilizando un diccionario fijo de palabras obtenidas de la Web y luego "stemmizadas" de 27000 ítems.

Como representación del documento, su prototipo utiliza sólo los 10 términos más pesados del documento.

La interpretación de la ecuación para obtener el peso V_i de las palabras del documento es la siguiente: La expresión del lado derecho es una fracción donde el denominador es un término de normalización. Por otro lado, el numerador es un producto de dos términos: El primero es directamente proporcional a la frecuencia de término $tf(i)$, es decir, cuantas más veces aparezca una palabra en un documento, ésta tendrá un valor más alto (se debe recordar que no hay "*stop words*"); el lector inferirá que el hecho de que una palabra aparezca muchas veces no la hace *per se* importante en la discriminación de un documento, al fin y al cabo, esta palabra podría llegar a ser una *stop word* que hubiéramos olvidado especificar. Aquí es donde empieza a jugar el segundo término del numerador, el cual es cual es el logaritmo de un número mayor a 1 –ya que $df(i)$ vale n en el peor de los casos (p_i sería tal vez una *stop word*)–; vemos que el segundo término es inversamente proporcional a la cantidad de documentos en los que aparece la palabra i-ésima, por lo tanto, si la palabra aparece en menos documentos, este hecho le da más puntos.

[6]Frecuencia de Términos × Inversa de la Frecuencia de Documento.

[7]Nota del autor: He modificado la notación original de los autores para hacerla más "castellana", salvo w_i que está anglizada (en obvia referencia a "peso" en inglés, *weight*).

Personalmente, creo que el uso de un diccionario para calcular la frecuencia de términos, si bien hace muy fácil la obtención de las características, adolece de los siguientes defectos: Primero, adolece del defecto de tener que actualizar el diccionario a medida que aparezcan palabras nuevas (bueno, los lenguajes son bastante estables, así que no sería tan grave). Segundo, cuando aparece una palabra que no figura en el diccionario, a ésta no se le puede calcular el peso, y, por lo tanto, ésta no va a aparecer en la representación del documento; en el caso en que la palabra sea crítica en la determinación de la relevancia del documento, el sistema fallará.

Inicialización de los Perfiles de Usuario

Los perfiles de usuario se inicializan a partir del *history file* del browser con lo que se va a inicializar un vector $\vec{M}$. Cada página $\vec{V_i}$ será vista por el usuario y recibirá una evaluación e_i (un entero en el rango [-5, +5]). Dada esta información, los pesos de $\vec{M}$ se actualizan de acuerdo a:

$$\vec{M} := \vec{M} + \sum_{i=1}^{p} e_i \vec{V_i} \tag{9.3}$$

En la literatura, esta fórmula se conoce como *feedback de relevancia* (sección 3.6).

Búsqueda

La postura de los autores [Balabanovic98] con respecto a la búsqueda de páginas Web es que la WWW es el espacio de búsqueda donde los documentos de texto y HTML representan los nodos mientras que los hipersaltos representan las aristas.

Los autores usan una heurística *best-first*[8] [Bigus98, Luger93, Rich94, Winston94] donde el puntaje de cada página es el inferido como la suma de los pesos de las características. Además, los autores expanden nodos con valores altos, y, también, tienen en cuenta el manejo de nodos ya visitados para no mostrarle dos veces la misma página al usuario.

Interfaz

La interfaz del agente es través de formularios CGI lo que permite usarlo desde cualquier punto de la Web.

9.3.9 Mantenimiento de Directorios de Recursos

En el artículo de [Cohen96b], se describe un método para mantener actualizados directorios de recursos. Un directorio de recursos es un documento HTML compuesto solamente de enlaces a sitios o páginas de interés en un tema dado. La motivación fundamental de este trabajo está basada en el hecho de que dichos directorios son difíciles de mantener mientras la lista de documentos on-line crece día a día.

En el trabajo mencionado, un directorio de recursos va a considerarse como una definición *extensional* (por extensión) de un concepto. Es decir, todos los documentos apuntados por el recurso van a considerarse como ejemplos del concepto mientras que el resto de los documentos van a ser casos negativos. Aquí, podemos observar un ejemplo de aplicación del concepto de hipótesis de mundo cerrado.

[8] Primero el mejor.

Al aprender dicho concepto por medio de técnicas de *machine learning*, será posible determinar si un documento cualquiera pertenece a dicho conjunto. Además, se trata de construir una definición *intencional* (por comprensión) a partir de los ejemplos (documentos enlazados) para usarla como una consulta a un buscador a fin de obtener más documentos que pertenezcan a este concepto. El problema puede considerarse como un caso particular de feedback de relevancia.

Estos autores han implementado dos sistemas: uno *batch* y otro interactivo.

Sistema Batch

Una de sus dos implementaciones es un script Perl que corre como un sistema batch –es decir, no requiere intervención del usuario. La entrada del sistema batch es una lista de URLs que correponden a los ejemplos positivos de un concepto desconocido. El sistema tiene dos salidas: una representación interna del concepto desconocido y una lista de documentos que incluye a todos los documentos positivos más una lista de documentos negativos.

A partir de los documentos el sistema construye un conjunto de cláusulas en forma normal disyuntiva de términos del documento; donde los términos que no se quiere que aparezcan se presentan negados.

Sistema Interactivo

El sistema interactivo está constituido por una interfaz basada en web browser. Al usuario se le presentan las páginas HTML modificadas con el agregado de enlaces especiales que aparecen al principio de las mismas. Dichos enlaces le permiten al usuario realizar operaciones como clasificar un documento o invocar al subsistema de aprendizaje.

Los algoritmos de aprendizaje usados por estos autores son *RIPPER* y el de los *Expertos Durmientes*.

Los resultados obtenidos por estos autores indican una precisión mayor al 90% con un *recall* también del 90%.

9.3.10 SenseMaker

SenseMaker [Wang97] es una interfaz de exploración de fuentes heterogéneas que soporta la evolución basada en contextos de los intereses de un usuario a través de: (1) una aproximación al contexto actual de información como la colección actual de referencias de información acumuladas, y (2) un conjunto unificado de acciones centradas en el usuario para examinar el contexto actual y para progresar hacia el siguiente.

9.3.11 SIFTER

SIFTER[9] [Mostafa97] es un sistema de filtrado de documentos implementado en base a un modelo propio de sus autores usando técnicas establecidas en recuperación de información e inteligencia artificial. Dichas técnicas incluyen representación de documentos por medio

[9]SIFTER es un acrónimo por *Smart Information Filtering Technology for Electronic Resources.*

del modelo de espacio vectorial, clasificación de documentos por aprendizaje no supervisado, y modelaje del usuario por aprendizaje basado en refuerzos. El sistema puede filtrar documentos basado en el contenido y en los intereses específicos de un usuario.

Modelo de Filtrado

En el modelo de filtrado de [Mostafa97] se consideran tres entidades importantes e independientes: la fuente de documentos, el filtro y el usuario.

Los documentos pueden existir en varios sitios y pueden ser recibidos por el usuario por medio de varios canales. La tarea de almacenar dichos documentos, antes del filtrado, es manejada por un componente que se llama *DAM* (aquisición y manejo de documentos). DAM es un componente separado del filtro y su diseño real puede variar de un ambiente a otro. Por ejemplo, DAM puede ser un *web crawler* que recupera documentos de sitios designados, un *daemon* que mantiene archivos indexados o aún un sofisticado DBMS. Cualquiera sea la factura del DAM, cuando se invoca, produce un flujo de documentos hacia el filtro.

El filtro mismo consiste de tres módulos: el representador, el clasificador y el manejador de perfiles.

Representación de Documentos

En SIFTER, la representación de documentos se hace usando la técnica TF*IDF (frecuencia de términos por la frecuencia inversa de documentos) para establecer el grado de importancia de cada concepto en un documento. Para aplicar esta técnica, se genera una tabla off-line conteniendo las frecuencias de todos los términos de thesaurus; dichas frecuencias se obtienen de una muestra de 2000 documentos elegidos aleatoriamente.

Clasificación de Documentos

En SIFTER los documentos sufren primero una etapa de clustering. Cada cluster de documentos C_i pasa a representarse por su centroide Z_i. Luego cada documento pasa a clasificarse en un cluster de acuerdo a un criterio de mínima distancia usando la medida de similitud del coseno entre documentos.

Resultados Experimentales

Los resultados de estos autores muestran que logran una precisión de 0.9 con un recall de 0.9 para un total de 35 sesiones de filtrado.

9.3.12 Sistemas de Soporte al NLP

Los agentes que hagan procesamiento de lenguaje natural (NLP) necesitarán soporte para la comprensión de los significados de las palabras que hallen en su camino. En esta sección, se describen los esfuerzos que la comunidad científica hace en este campo. Se describen por separado tres aplicaciones: CYC, WordNet y el Diccionario Electrónico EDR. Una comparación de sus fortalezas y desventajas puede hallarse en [Lenat95b].

CYC

Desde 1984, se está construyendo *CYC*, un esquema universal de 10^5 conceptos relacionados con la vida humana [Lenat95a]. Mucho de este tiempo se ha usado codificando dichos conceptos; aproximadamente 10^6 axiomas de sentido común se han sido introducidos a mano y millones han sido inferidos y almacenados por CYC. CYC puede pensarse como un sistema experto con un dominio que abarca objetos y acciones de todos los días.

Tal sustrato de sentido común puede servir como una ontología estándar subyacendo la World Wide Web y el comercio electrónico. Como un esquema universal, puede ayudar a estandarizar –y hacer más eficiente– la recuperación, integración y validación de la consistencia de la información.

La esperanza de los autores es que en diez años, las aplicaciones basadas en procesamiento de lenguaje natural y *machine learning* puedan escalar a la altura de la WWW, quizá ayudándose con un *framework* como CYC.

WordNet

Cualquier programa que quiera hacer procesamiento de lenguaje natural debe tener información acerca de las palabras y de sus significados (sección 3.10). *WordNet* es una base de datos lexicográfica que se diseñó para usarse bajo control de programa (a diferencia de los diccionarios tradicionales, que están orientados al usuario) [Miller95].

WordNet contiene información de thesaurus conteniendo la siguientes relaciones entre palabras [Miller95]:

- La *sinonimia* es la relación básica en WordNet, porque usa conjuntos de sinónimos[10] (*synsets*) para representar los significados de las palabras. La sinonimia es una relación simétrica entre las palabras.

- La *antonimia* (nombres opuestos) es también una relación semántica simétrica entre palabras, especialmente importante para organizar los significados de adjetivos y adverbios.

- La *hiponimia* (subnombre) y su inversa, la *hipernimia* (supernombre), son relaciones transitivas entre synsets. Como hay usualmente sólo un hipérnimo, esta relación semántica organiza los significados de los nombres en una estructura jerárquica.

- La *meronimia* (parte de) y su inversa, la *holonimia* (nombre entero), son relaciones semánticas complejas. WordNet distingue partes *componentes*, partes *sustantivas* y partes *miembro*.

- La *troponimia* (nombre de manera) es para los verbos lo que la hipernimia es para los nombres, a pesar de que las jerarquías resultantes son mucho menos profundas.

Diccionario Electrónico EDR

El *Diccionario Electrónico EDR* es un proyecto de nueve años financiado por el gobierno japonés para diseñar una gran base de datos de referencias cruzadas entre palabras, términos y oraciones japonesas e inglesas [Yokoi95]. Esta base de datos se espera que sea

[10] Dos sinónimos son dos palabras con el mismo significado.

la fundación de una aún mayor base de datos de referencias cruzadas del conocimiento mundial.

El diccionario incluye [Yokoi95]:

- El *Diccionario de Palabras*, el cual incluye las relaciones entre palabras y conceptos (equivalente al sentido de una palabra), como también las características gramaticales de las palabras y sus respectivos significados, permitiéndoles a las computadoras realizar procesamientos morfológico y sintáctico.

- El *Diccionario de Clasificación de Conceptos*, el cual clasifica todos los conceptos en super/sub relaciones y es similar a un thesaurus (sección 3.10), ayudando a las computadoras a encontrar conceptos similares o equivalentes y a calcular el grado de similitud entre los conceptos.

- El *Diccionario de Descripción de Conceptos*, el cual describe las co-ocurrencias semánticas entre conceptos, ayudando a las computadoras a juzgar correctitud semántica.

- El *Diccionario de Co-ocurrencias*, el cual describe las co-ocurrencias de palabras en un nivel superficial, ayudando a las computadoras a entender oraciones en lenguaje natural.

- El *Diccionario Bilingüe*, el cual describe los significados correspondientes entre palabras japonesas e inglesas, permitiendo a las computadoras hallar palabras equivalentes apropiadas.

- El *Cuerpo EDR*, el cual es una colección de datos usados en el desarrollo del diccionario y está basado en oraciones extraídas de texto al cual se le agregan los resultados de análisis morfológicos, sintácticos y semánticos.

- La *Base de Texto EDR* es una colección de grandes volúmenes de texto usados como datos en la investigación del procesamiento del lenguaje. Esta base de datos se creó a partir de texto legible por máquina de diarios y otros materiales publicados.

9.3.13 Syskill & Webert

Syskill & Webert es un agente inteligente que aprende un modelo de los intereses de un usuario en un tema específico, basado en páginas web calificadas [Billsus97, Pazzani97a, Pazzani97b]. El sistema está implementado como una extensión de un web browser y provee una interfaz que le permite a los usuarios decir si una página es *hot* (caliente) o *cold* (fría).

Una sesión típica de Syskill & Webert procede de la siguiente manera [Billsus97]: Primero, el usuario elige un tema arbitrario de interés. El usuario entonces puede navegar o buscar en la web y clasificar cualquier página que juzgue de interés como hot, y cualquiera que no sea de su interés como cold. Tan pronto como Syskill & Webert juntó un número mínimo de clasificaciones (en su implementación son 10), puede usar las páginas para inferir un modelo probabilístico de los intereses del usuario.

El modelo entonces puede usarse en dos maneras [Billsus97]: Primero, Syskill & Webert va a anotar los enlaces de cualquier página de la que considere que puede ser de los

gustos del usuario. Segundo, Syskill & Webert puede usar el modelo aprendido para construir una consulta a un motor de búsqueda, mandar la consulta, recolectar los resultados y finalmente evaluarlos con respecto al modelo de los gustos del usuario.

Método de Clasificación

El cálculo para determinar si una página va a ser o no del gusto del usuario se hace en base a la probabilidad de que la página en cuestión sea hot [Billsus97, Pazzani97a]. Dicho cálculo se hace con el método de clasificación supervisado llamado clasificador bayesiano (SBC) (sección 5.2). Como el SBC requiere que los documentos sean representados como vectores de características, estos autores convierten los documentos de texto en un conjunto vectores de características Booleanos, donde cada característica indica si una palabra está presente o ausente en un documento particular. Para hallar las características, seleccionan palabras que ocurren frecuentemente en páginas calificadas como interesantes y, a la vez, infrecuentemente en páginas no interesantes. Esto se hace hallando la ganancia de información esperada que la presencia o ausencia de una palabra da para la clasificación de elementos de un conjunto de documentos. Usando este enfoque, hallan las k (96 en realidad) palabras más informativas del conjunto corriente de páginas calificadas.

El uso del clasificador bayesiano deriva de que se puede usar para determinar la probabilidad de que un ejemplo j pertenezca a la clase C_i dados los valores de los atributos del ejemplo [Pazzani97a]:

$$P(C_i|A_1 = V_{1j} \& \ldots \& A_n = V_{nj}) \tag{9.4}$$

Si los valores de los atributos son independientes, esta probabilidad es proporcional a:

$$P(C_i) \prod_k P(A_k = V_{kj}|C_i) \tag{9.5}$$

$P(A_k = V_{kj}|C_i)$ (la probabilidad de que una página que contenga una palabra dada sea hot) y $P(C_i)$ (la probabilidad de que una página sea hot) pueden estimarse a partir de los datos de entrenamiento. Para determinar la clase más probable de un ejemplo (es decir, si es hot o cold), se computa la probabilidad de cada clase. El ejemplo se asigna entonces a la clase con mayor probabilidad.

Resultados Obtenidos

Estos autores realizaron experimentos con usuarios reales interesados en diversos temas y obtuvieron resultados con una precisión promedio entre el 62.9% y el 80% [Pazzani97a].

También, estos autores realizaron mediciones con otros métodos como el del vecino más cercano, perceptrón, red de backpropagation, y reformulación de consultas de Rocchio. Estos resultados pueden consultarse en [Pazzani97a].

9.4 Resumen

En este capítulo se describieron la tecnología de agentes, se explicó cómo interactuar con los servicios de búsqueda de la Web, los recaudos que es necesario tener a la hora de interactuar con servidores remotos y se describieron las aplicaciones de agentes halladas en la literatura.

Capítulo 10

Querando!

En este capítulo se describen los aspectos de implementación del agente de filtrado de páginas HTML llamado *Querando!*.

La exposición está estructurada como sigue: primero, se describe la interfaz de interacción con el usuario; segundo, se describe como se logra este comportamiento; tercero, se detallan resultados experimentales con un usuario real; por último, se compara esta implementación con los trabajos similares de la literatura.

10.1 Interfaz del Agente

El agente está implementado como un conjunto de documentos HTML con código JavaScript embebido y un conjunto de scripts CGI. De esta manera, el mismo es accesible desde cualquier punto del mundo usando una computadora conectada a Internet y con un browser instalado.

10.1.1 Log-In

Una sesión típica de Querando ocurre de la siguiente manera. Primero, el usuario se loguea al sistema con su nombre de usuario y su password usando la página `/querando/hello.html` (figura 10.1 izq.). Si el usuario no está registrado, puede hacerlo llenando un formulario con sus datos personales.

Una vez que se ingresó al sistema, se tiene un menú con tres opciones: iniciar una nueva sesión de filtrado, editar los perfiles de filtrado o continuar una sesión anterior (figura 10.1 der.).

10.1.2 Manejo de Perfiles

Cada perfil define un concepto que se va a usar para filtrar documentos HTML. Este enfoque ya se usó en la literatura de agentes de filtrado de información (veáse sección 9.3).

En la página de manejo de perfiles (figura 10.2), el usuario tiene la opción de definir nuevos perfiles de filtrado, borrarlos o modificar sus parámetros.

La creación de un perfil se hace dándole un identificador de perfil y definiendo los tres parámetros de la red FART, a saber, α, β y ρ (veáse sección 5.6.10).

En esta implementación los perfiles no se pueden compartir entre los usuarios ni con otras instancias del agente, como sí se hace en [Lashkari97] (veáse sección 9.3.1).

Figura 10.1: Entrada al agente (izq.) y menú principal (der.).

Figura 10.2: Manejo de perfiles de filtrado.

10.1.3 Sesión de Filtrado

Creación de una Sesión

Al elegir la creación de una nueva sesión de filtrado, se despliega la página HTML de la figura 10.3. Allí, se le piden al usuario el nombre de la sesión (el sistema provee uno formado por la fecha y la hora), la url del reservorio o las claves de búsqueda y los buscadores a los que hay consultar. Otras opciones son cómo se van a mostrar los resultados (filtrado batch o interactivo). Además, se permite especificar la profundidad con que se va a explorar el grafo de la WWW a partir de cada enlace. También, se puede especificar el tiempo entre hits a los servidores (cumpliendo con las reglas de etiqueta de la sección 9.2).

La obtención de las páginas HTML para filtrar se obtienen de una de tres maneras:

1. *Usando un reservorio de enlaces:* En este caso, los enlaces son parte de un documento de la WWW que contiene múltiples referencias a otros documentos o sitios con contenido en un tema particular. Si se toma esta opción, el usuario deberá especificar con qué profundidad se explorará cada enlace para formar el vector de características de los mismos; aquí, se creará un único vector por cada enlace (ver caso 3) con la suma de las características de todos los documentos explorados.

2. *Usando un buscador de la web:* En este caso, los documentos para filtrar se obtienen haciendo una consulta a uno de los buscadores de la WWW. Actualmente, el sistema soporta los siguientes buscadores: Altavista, Excite, Deja, Google, Hotbot, Infoseek, Lycos y Yahoo; se interactúa con éstos usando los protocolos definidos en la sección 9.1.

3. *Haciendo un Breadth-First Search desde una URL:* Finalmente, el usuario puede especificar una URL de partida y una profundidad. El sistema armará una página reservorio de enlaces formada por todos aquellos documentos que se recuperaran desde dicha URL recorriendo el grafo de la WWW hasta la profundidad especificada. Alternativamente, para acotar la cantidad potencialmente grande de documentos a recuperar, el usuario también puede especificar una cota para la cantidad de documentos a recuperar.

   En esta opción, cada documento será clasificado y aprendido por la red neuronal en forma independiente.

Para esta primera aproximación, se usará un reservorio de enlaces y usemos como dirección del mismo `http://localhost/tesis/paginas/testx/index.htm`. Ésta es una página bien simplificada, pues solamente posee cuatro enlaces a cuatro páginas distintas. Además, pondremos cero como profundidad del árbol de búsqueda y modo de uso interactivo.

El agente recupera de la Web a la página reservorio de enlaces y la modifica agregándole una lista de selección por cada enlace y unos controles HTML en la parte inferior de la página. Las listas de selección le permiten al usuario indicar sus gustos por las páginas referenciadas por los enlaces, a saber: *Buena* (relevante), *Mala* (irrelevante), *No Clasificado* (clasificar en esta sesión) y *No Clasificar* (no clasificar en esta sesión). Por otro lado, los controles le permiten despachar la clasificación al servidor, elegir si la red neuronal se

Figura 10.3: Creación de sesión de filtrado.

Figura 10.4: Reservorio original (izq.) y modificado (der.).

usa en modo de producción o en aprendizaje y si se usa la tabla de *Spool*[1] para obtener los documentos asociados a los enlaces.

En la figura 10.4, se observa la diferencia entre la página reservorio original (izquierda) y cómo la página es modificada por el agente (derecha). En secciones posteriores se explicará cómo se implementó de dicha modificación.

Desarrollo de la Sesión de Filtrado

El usuario entonces usará al agente para que este le sugiera qué páginas le son relevantes. Por ejemplo, si el usuario quiere saber sobre la relevancia o irrelevancia del segundo enlace seteará todos las listas de selección a "No Clasificar" con el botón correspondiente y luego

[1]La tabla de *Spool* surgió como una necesidad durante las etapas iniciales del desarrollo. Al no disponer de una conexión a Internet, el recurso de interponer una tabla cache entre el agente y la web permitió simular el comportamiento del agente en condiciones reales.

Figura 10.5: Clasificación de un enlace.

seteará la segunda lista de selección a "No Clasificado" para indicar que quiere que el agente evalúe dicho enlace. Se despachan los datos al servidor y, con suerte, el agente contestará sobre la relevancia de la página. En la figura 10.5 se observa esta situación.

El valor de clasificación de las páginas asociadas a los enlaces es expresado por el agente con un ícono junto al mismo. Las opciones posibles son:

1. La relevancia de una página se expresa con una $\uparrow$ verde.
2. La irrelevancia de una paǵina se expresa con una $\downarrow$ roja.
3. Cuando no se puede recuperar la página asociada se muestra un ícono $\sim \exists$ (*no existe*).
4. Cuando el agente no sabe qué contestar, se expresa con un ? rojo.
5. También se usan combinaciones de los cuatro símbolos anteriores para indicar un cambio de estado en los gustos del usuario o alguna condición anómala:
   (a) Cuando el usuario contestó que la página es irrelevante pero, sin embargo, el agente no pudo recuperarla, la situación se indica con la combinación $\downarrow\sim \exists$.
   (b) Cuando el usuario indicó que la página es irrelevante y el sistema no lo sabía, aprende este gusto y lo indica con la combinación $\downarrow$?.
   (c) Cuando el usuario indicó que la página es irrelevante y el sistema creía que ésta era relevante; se aprende el cambio y se lo indica con la combinación $\downarrow\uparrow$.
   (d) Cuando el usuario indica relevante y el sistema creía irrelevante, el cambio se anota y se lo indica con $\uparrow\downarrow$.
   (e) Cuando el usuario indica relevante y la página no se puede recuperar, se indica con $\uparrow\sim \exists$.
   (f) Cuando el usuario indica relevante y el sistema no sabía qué contestar, se toma nota y se indica con $\uparrow$?.

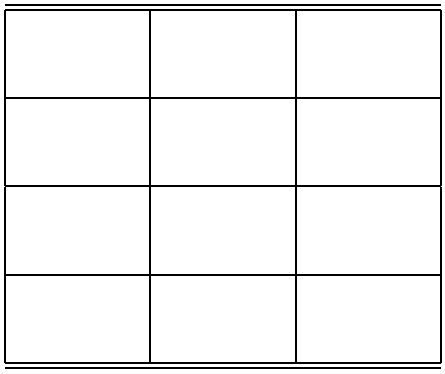

Figura 10.6: Iconos de clasificación de enlaces.

Figura 10.7: Otras operaciones sobre una sesión de filtrado.

Esta característica de la interfaz está basada en los trabajos de [Billsus97, Pazzani97a, Pazzani97b].

Existen otras operaciones que se pueden realizar sobre una sesión. Una sesión también puede interrumpirse y continuarse en otro momento. También se puede borrar con lo cual se perderán los datos de dicha sesión y no así el aprendizaje del agente en el perfil particular de dicha sesión. Estas operaciones se realizan en la pantalla de la figura 10.7.

10.2 Cuestiones de Implementación

En esta sección se explica cómo se logra el comportamiento descripto en la sección anterior.

10.2.1 Modificación de Páginas HTML

Las páginas para filtrar o aprender un nuevo concepto se pueden obtener de dos maneras: usando un reservorio de enlaces o haciendo consultas a un motor de búsqueda de la web. Los reservorios de enlaces son páginas mantenidas por una persona que contienen una lista de referencias a sitios, artículos y otros documentos HTML de un tema de interés. Por otro lado, los motores de búsqueda de la web ya fueron tratados en la sección 9.1.

```
<html>
<head><title>Dos Links</title></head>
<body>
   <br><a href="p1.htm">P1</a>
   <br><a href="p2.htm">P2</a>
   <br><a href="http://www.unsitio.com/p3.htm">P3</a>
   <br><a href="p4.htm">P4</a>
</body>
</html>
```

Figura 10.8: Documento reservorio de enlaces original.

Páginas Reservorios

Supongamos que el usuario elige como reservorio de enlaces al documento de URL:

`http://localhost/tesis/paginas/testx/index.htm`

y contenido como el de la figura 10.8. `Index.htm` es una página HTML reservorio de enlaces bien simplificada, pues solamente posee cuatro enlaces a cuatro páginas distintas.

El sistema modifica el documento HTML para quedar como el de la figura 10.9. Aquí, se puede observar que se ha agregado una cantidad considerable de código HTML. El código agregado corresponde a:

1. Un formulario HTML para que los datos de la evaluación de las páginas por parte del usuario se envíen a un programa CGI en el servidor remoto donde está el agente.

2. Los datos correspondientes a los identificadores del usuario, el perfil de filtrado y la sesión (la seguridad no es para nada fuerte en esta implementación). Estos datos se agregan como datos ocultos en el formulario en la forma de marcadores HTML `HIDDEN`.

3. La modificación de los enlaces. Considere el segundo de ellos. El enlace original era:

   ```
   <a href="p2.htm">P2</a>
   ```

   Ahora, pasó a ser:

   ```
   <SELECT name="CLASIFICACION_1" ID="CLASIF_1">
   <OPTION SELECTED>No Clasificado
   <OPTION>Buena
   <OPTION>Mala
   <OPTION>No Clasificar
   </SELECT>
   <INPUT TYPE="hidden" NAME="URL_ITEM_1"
          VALUE="http://localhost/tesis/paginas/testx/p2.htm">
   <a href="http://localhost/tesis/paginas/testx/p2.htm" target=_new>P2</a>
   ```

```
<html>
<head><title>Dos Links</title></head>
<body>
<FORM NAME=Formulario METHOD="POST" ACTION="aprendizFART.exe">
<INPUT TYPE=hidden NAME='UserID' VALUE='jperez'>
<INPUT TYPE=hidden NAME='PerfilID' VALUE='Perfil1'>
<INPUT TYPE=hidden NAME='SesionID' VALUE='29/11/2000 0:18.8'><br>

<SELECT name="CLASIFICACION_0" ID="CLASIF_0">
<OPTION SELECTED>No Clasificado
<OPTION>Buena
<OPTION>Mala
<OPTION>No Clasificar
</SELECT>
<INPUT TYPE="hidden" NAME="URL_ITEM_0"
       VALUE="http://localhost/tesis/paginas/testx/p1.htm">
<a href="http://localhost/tesis/paginas/testx/p1.htm" target=_new>P1</a>
<br>

... Idem enlaces posteriores ...

<INPUT TYPE=HIDDEN NAME="CANT_ITEMS" VALUE="4">

<SCRIPT language="JavaScript">
                 ......Script para manejo de las listas ....
</SCRIPT>

.......... Controles HTML para los botones del formulario ...

</FORM>

</body>
</html>
```

Figura 10.9: Página reservorio modificada.

En el enlace, se han sumado los siguientes elementos:

(a) Una lista de selección con las opciones *Buena* (relevante), *Mala* (irrelevante), *No Clasificado* (clasificar en esta sesión) y *No Clasificar* (no clasificar en esta sesión).

(b) Un "input hidden" con la URL del enlace y un identificador que crecerá incrementalmente.

(c) La URL del enlace se hace absoluta en el caso de que sea relativa al sitio de origen. Las URLs relativas lo serán al sitio del agente y, por lo tanto, dejan de ser válidas. Si las URLs son absolutas, el sistema no las modifica.

También, se agrega el parámetro *target=_new* para que, en caso de que el usuario explore las páginas asociadas al enlace, éstas se muestren en otra ventana del explorador.

Además, el sistema modifica cualquier marcador HTML que contuviera URLs relativas al sitio de origen del mismo. La lista de tags HTML con sus parámetros modificables se muestra en la tabla 10.2.1 y está basada en la información de tags HTML del libro David Gulbransen [Gulbransen98, Pp. 455-480] y el RFC de la especificación de HTML Version 4.0 [Raggett98].

Por lo anterior, el agente tendrá un archivo de configuración llamado `TAGS.TXT` donde se indicarán los datos enumerados en la tabla 10.2.1 (10.1) con el formato:

$$< TAG > Parametro_1, Parametro_2, \ldots, Parametro_n. \tag{10.1}$$

Modificación a las Páginas de Buscadores

Primeramente, se realizó una implementación donde se le hacía una consulta a Altavista y se obtenían las referencias a los resultados siguiendo las indicaciones de [Weber97]. Sin embargo, dicha implementación dejó de ser válida al día de la fecha ya que Altavista cambió el formato de sus páginas.

Actualmente, cuando el usuario decide usar un motor de búsqueda para entrenar al agente, especifa cuál motor va a usar y las claves de búsqueda. Así, el sistema construye la URL correspondiente siguiendo las indicaciones de la sección 9.1 y la página resultante es tratada como un reservorio de enlaces. Actualmente, el sistema trabaja con los siguientes motores de búsqueda: Altavista, Excite, Deja, Google, Hotbot, Infoseek, Lycos y Yahoo.

10.2.2 Estructura de Directorios

El sistema está implementado como un conjunto de documentos HTML con código JavaScript (sección 8.10.6) embebido y programas CGI (sección 8.8). Por lo tanto, el sistema reside en un servidor para hacerlo accesible de toda la web. Actualmente, el sistema ha sido probado solamente en el Servidor Personal de Web de Windows 98 (veáse la sección 8.9).

Los datos del agente están compuestos por archivos de texto y por bases de datos Access.

Los archivos y programas del agente están distribuidos en la siguiente estructura de directorios:

Tag HTML	*Lista de parámetros con referencias*
<A>	Class, HRef, Url.
<ADDRESS>	Class.
<APPLET>	Class, Code, CodeBase, Src.
<AREA>	Class, HRef.
<BASE>	Class, HRef.
<BASEFONT>	Class.
<BGSOUND>	Class, Src.
<BIG>	Class.
<BLOCKQUOTE>	Class.
<BODY>	Background, Class.
<BUTTON>	Class.
<CAPTION>	Class.
<COL>	Class.
<COLGROUP>	Class.
<DIV>	Class.
<EMBED>	Class, Code, Src.
<FONT>	Class.
<FORM>	Action, Class.
<FRAME>	Class, Src.
<FRAMESET>	Class.
<HR>	Class, Src.
<IMG>	Class, Src.
<INPUT>	Class, Src.
<LABEL>	Class.
<LINK>	HRef.
<MAP>	HRef.
<MARQUEE>	Class.
<META>	Url.
<OBJECT>	Class, Code.
<OPTION>	Class.
<SCRIPT>	Src.
<SELECT>	Class.
<SPAN>	Class.
<TABLE>	Class.
<TD>	Background, Class.
<TEXTAREA>	Class.
<TR>	Class.

Tabla 10.1: Tags HTML con referencias relativas al site de origen.

querando El directorio raíz del agente *Querando!*. El único archivo de este directorio es *hello.html*, que corresponde a la página de log-in al sistema.

Los subdirectorios son los siguientes:

cgi En este directorio residen los programas CGI de la aplicación:

aprendizfart.exe El programa que realiza el entrenamiento de la red neuronal de la teoría de la resonancia adaptativa difusa.

hello.exe Programa de manejo de perfiles y sesiones de usuario.

main.exe Programa encargado de hacer las modificaciones a las páginas reservorio e interactuar con los buscadores de la web.

nuevoUsuario.exe Programa encargado de dar de registrar un nuevo usuario.

datos Los datos manejados por el sistema son:

bd1.mdb La base de datos con información sobre usuarios, perfiles y sesiones.

spool.mdb La tabla que sirve de cache de los documentos HTML procesados.

stop2.txt La lista de las *stop words* (veáse la sección 3.8).

TAGS.TXT La lista de los tags a los que hay que modificarles parámetros que pudieran contener referencias locales a sitios de origen.

word_endings.txt La lista de sufijos del idioma inglés para realizar el *stemming* (veáse la sección 3.9).

html En este subdirectorio se alojan todas las páginas HTML del sistema. Éstas no son accedidas directamente por el usuario sino que el acceso se hace a través de guiones CGI. Este directorio contiene otro subdirectorio llamado **images** donde se almacenan las fotos correspondientes a los íconos de clasificación de enlaces (sección 10.1.3).

10.2.3 Programas CGI

Para realizar toda la comunicación en CGI, se implementaron librerías en C++ para recuperación de parámetros siguiendo recomendaciones de [Dwight97].

10.2.4 Base de Datos y Archivos

Para representar los datos persistentes del archivo se usaron archivos de texto y dos bases de datos.

Los archivos de texto ya fueron descriptos en la sección 10.2.2. Son archivos de texto plano, la única salvedad es que el archivo de stems contiene las palabras invertidas, ya que de esta manera la implementación es más eficiente. El archivo de marcadores a modificar tiene el formato explicado en la sección 10.2.1. El archivo de stop words es un archivo de texto donde hay una palabra por renglón.

Las bases de datos están en formato Access. Su estructura fue creada con dicho producto y se acceden usando *drivers* ODBC [Jennings99] usando programas C++. En general, las tablas se manejan usando una combinación de código procedural y consultas SQL [Korth86].

Hay dos bases de datos:

Figura 10.10: Base de datos de *Spool*.

1. *Spool.mdb*: Esta base de datos está compuesta por una única tabla y contiene el texto de los documentos HTML accedidos de la web. Esto permitió hacer el desarrollo de la interfaz con los motores de búsqueda sin necesidad de estar conectado en todo momento a Internet. Sólo se guardó texto HTML y/o de texto y no se almacenaron objetos referenciados como ser fotos, películas o archivos de sonido. La estructura de dicha base se observa en la figura 10.10.

2. *bd1.mdb*: Esta base de datos contiene toda la información de los usuarios del agente así como los perfiles que tuvieran definidos y las sesiones de filtrado definidas sobre esos perfiles. La estructura de esta base de datos se puede ver en la figura 10.11.

   En general, todas las relaciones son con entidades débiles [Korth86]. La única entidad fuerte es el usuario. Un usuario posee perfiles; a su vez, sobre éstos se definen sesiones. El resto de las tablas son para manejar los datos asociados a los perfiles (como los pesos de la red neuronal) y a las sesiones (como texto de las páginas modificadas y resultados obtenidos y pendientes).

   Las tablas de esta base son:

   (a) *Usuarios*: Almacena información sobre los usuarios del sistema. Las altas sobre esta tabla las hace el programa *NuevoUsuario.exe*. La clave es el campo "UserID".

   (b) *Perfiles*: Almacena información sobre los perfiles de un usuario dado y es una entidad débil relacionada con la tabla *Usuarios* de forma "one-to-many". Los campos clave son "UserID" y "PerfilID".

Figura 10.11: Base de datos *BD1*.

(c) *Pesos*: Cada registro almacena los pesos de una categoría de la red neuronal FART. Es una entidad débil de la tabla *Perfiles*.

(d) *Valores Categorías*: En esta tabla se almacena el valor de clasificación del usuario para cada categoría de filtrado. Su clave está dada por los campos: "UserID", "PerfilID" y "CategoriaID". Es una entidad débil relacionada a la tabla *Perfiles*.

(e) *Valores Categorías Bigus*: Idem a Valores Categorías salvo que esta vez se usa para el método de claves.

(f) *Sesiones*: Almacena la información concerniente a las sesiones de usuario como ser: la URL del reservorio de datos, la profundidad del recorrido BFS, las claves de consulta, el tiempo entre hits a servidores, etc. Su clave es "UserID", "PerfilID" y "SesionID" y es una entidad débil relacionada a *Perfiles*.

(g) *Texto Páginas Sesiones*: En esta tabla se almacenan los textos de las sesiones; éstos están formados por páginas HTML con marcadores especiales para poder reconstruir una sesión incompleta cuando el usuario lo desee. Se usan varios registros por sesión debido a que Access no soporta campos de tamaño superior a 65Kb. Esta tabla es una entidad débil asociada a *Sesiones* y su clave está formada por los campos "UserID", "PerfilID", "SesionID" y "NroPagina".

(h) *ResultadosSesiones*: En esta tabla se almacenan los resultados de la calificación de cada enlace. También es una entidad débil y está asociada a la tabla *Sesiones*. Su clave está conformada por los campos "UserID", "PerfilID", "SesionID" y "NroItemClasificacion".

(i) *Texto Páginas Reservorio*: En esta tabla se almacena el texto de las páginas de enlaces originales. También aquí se usa más de un registro por página debido a la restricción de los 65Kb como tamaño máximo de campo.

10.2.5 Clases y Archivos Fuente

En esta sección se describe el conjunto de programas ejecutables escritos en C++. La división en archivos del código fuente es la siguiente:

1. *AlocadorDeMemoria:* Encapsulamiento de rutinas de alocación y desalocación de memoria. Los archivos fuentes son *AlocadorDeMemoria.h* y *AlocadorDeMemoria.cpp* que implementan la clase *AlocadorDeMemoria*.

2. *AnalizadorDeFeatures:* Programa para el análisis inicial de la representación de trigramas explorada en el capítulo 4. El archivo se llama *AnalizadorDeFeatures.cpp* e implementa la clase *AnalizadorDeFeatures*.

3. *AplicacionCGI:* Esta clase implementa la interfaz común de gateway (sección 8.8), lo cual se hizo siguiendo direcciones de [Duncan96, Dwight97]. Los archivos implicados son *AplicacionCGI.h* y *AplicacionCGI.cpp*. Esta es una clase abstracta.

4. *AplicacionHello:* Esta clase implementa las operaciones de manejo de perfiles y de sesiones de usuario. Los archivos implicados son *AplicacionCGI.h* y *AplicacionCGI.cpp*.

5. *AplicacionMain:* Esta clase implementa la creación de sesiones de usuario. Los archivos implicados son *AplicacionMain.h* y *AplicacionMain.cpp*.

6. *AprendizFART:* Esta clase maneja todo lo concerniente al entrenamiento de los pesos de la red neuronal durante las sesiones de filtrado así como la calificación automática de las páginas HTML. Los archivos son *AprendizFART.h* y *AprendizFART.cpp*. Esta clase es subclase *AplicacionCGI*.

7. *ArchivosShare:* Esta clase se implementa el acceso compartido a archivos de texto de lectura/escritura. Los archivos implicados son *ArchivosShare.h* y *ArchivosShare.cpp*.

8. *ArchivoTexto:* Esta clase implementa el *pattern* de objetos "Decorador" para acceder a archivos de texto con un buffer interno en forma eficiente. Los archivos que la implementan son *ArchivoTexto.h* y *ArchivoTexto.cpp*.

9. *ArchivoTextoHandle:* Igual que la clase *ArchivoTexto* pero preparada para acceso concurrente. Los archivos que la implementan son *ArchivoTextoHandle.h* y *ArchivoTextoHandle.cpp*.

10. *Art1:* Implementación temprana de la red de la teoría de la resonancia adaptativa (sección 5.6.9).

11. *AVL:* Implementaciíon del árbol binario de búsqueda balanceado genérico. La genericidad de las colecciones se implementó en C++ usando las clases contenidas abstractas *Storable* y *StorableComparable* [Brokken95]. Los archivos que la implementan son *AVL.h* y *AVL.cpp*. Además, AVL es subclase de *Storable*.

12. *BaseAlfabeto:* Esta clase implementa una capa de abstracción para la base del alfabeto de símbolos usados en la representación de los trigramas de los documentos HTML. Se implementaron iteradores de los símbolos de los alfabetos para hacer que el código de los programas que la usan sean invariantes ante el cambio del conjunto de caracteres usado para representar a los documentos. Los archivos donde se implementan son *Base.h* y *Base.cpp*.

13. *BFS:* Subclase de *Storable* que implementa el recorrido en anchura [Aho83b, Weiss92] de un subgrafo acotado de la WWW acotado. Los archivos que lo implementan son *BFS.h* y *BFS.cpp*.

14. *ColaDinamica:* Clase que implementa el tipo abstracto cola de *Storable*. Los archivos correspondientes son *ColaDinamica.h* y *ColaDinamica.cpp*.

15. *Comunes:* Directorio que contiene los siguientes archivos:

    (a) *readingSTDIN.cpp* y *readingSTDIN.h*: Versión estática de rutinas CGI.

    (b) *comunes.cpp* y *comunes.h*: Funciones de uso común (como manipulación de strings y definición de constantes).

    (c) *paths.h*: Ubicaciones de archivos de uso común.

    (d) *readingSTDINDinamico.cpp* y *readingSTDINDinamico.h*: Versión dinámica de rutinas CGI.

    (e) *StdAfx.h*: Librería standard de las Microsoft Foundation Classes [Kruglinski97].

16. *ComunesNuevoUsuarioHello:* Rutinas compartidas entre las aplicaciones Hello y NuevoUsuario. Esta clase es superclase de las clases *AplicacionHello* y *NuevoUsuario*. Los archivos comprendidos son *ComunesNuevoUsuarioHello.h* y *ComunesNuevoUsuarioHello.cpp*.

17. *DecoradorDeHTML:* Esta clase implementa la funcionalidad de tomar una ColaDinamica conteniendo el stream de un archivo HTML y devuelve otra cola dinámica donde entrada será o bien un marcador HTML o el texto entre dos marcadores HTML. Los archivos que la implementan son *DecoradorDeHTML.h* y *DecoradorDeHTML.cpp*.

18. *FartCGI:* Implementación de la red neuronal FART (sección 5.6.10) pero orientada a una sesión CGI ya que almacena sus pesos en la base de datos *BD1*. Los archivos son *FartCGI.h* y *FartCGI.cpp*.

19. *FloatWrapper:* Clase que implementa el *wrapper* de los flotantes de C para poder usarlos como Storable en las colecciones genéricas. Archivos *FloatWrapper.h* y *FloatWrapper.cpp*.

20. *FuzzyART:* Implementación de la red neuronal FART (sección 5.6.10) como parte del programa *TestFART.exe*. Archivos *FuzzyART.h* y *FuzzyART.cpp*.

21. *GeneradorDeFeatures:* Clase que se encarga en generar el vector de características de un documento de texto o HTML. Los archivos son *GeneradorDeFeatures.h* y *GeneradorDeFeatures.cpp*.

22. *GenericTest:* Clase abstracta que factoriza comportamiento de *TestKMediasDocs* y *TestKohonenNet*. Archivos *GenericTest.h* y *GenericTest.cpp*.

23. *Hello:* Programa para la clase *AplicacionHello*. Archivo *Hello.cpp*.

24. *HTMLFileStream:* Esta clase implementa un iterador de un archivo de HTML almacenado en una cola dinámica. Los archivos de los que se compone son *HTMLFileStream.h* y *HTMLFileStream.cpp*.

25. *HTMLFileTokenizer:* Esta clase también es un decorador de un documento HTML, devuelve sólo las palabras del mismo sin devolver los marcadores. Se compone de los archivos *HTMLFileTokenizer.h* y *HTMLFileTokenizer.cpp*.

26. *InterfazBuscadores:* Esta clase implementa la recuperación y extracción de enlaces de los motores de búsqueda Altavista y Excite. Está compuesta por los archivos *InterfazBuscadores.h* y *InterfazBuscadores.cpp*.

27. *KMedias:* Esta clase es la implementación del algoritmo de las K-medias (sección 5.3.3). Está compuesta por los archivos *KMedias.h* y *KMedias.cpp*.

28. *KohonenNet:* Esta clase implementa las redes neuronales de contrapropagación y mapa autoorganizativo de Kohonen (secciones 5.6.7 y 5.6.8, respectivamente). Está compuesta por los archivos *KohonenNet.h* y *KohonenNet.cpp*.

29. *Main:* Programa cáscara para la clase *AplicacionMain*. Se compone del fuente *Main.cpp*.

30. *MaxHeap:* Clase para implementar una cola de prioridades; archivos *MaxHeap.h* y *MaxHeap.cpp*.

31. *ModificadorDePaginasHTML:* Clase encargada de modificar las páginas reservorios de enlaces y resultados de consultas a motores de búsqueda; archivos *ModificadorDePaginasHTML.h* y *ModificadorDePaginasHTML.cpp*.

32. *NuevoUsuario:* Programa que hace el alta de un nuevo usuario del agente *Querando!*. La clase nuevo usuario es subclase de *AplicacionCGI* y *ComunesNuevoUsuarioHello*. Archivos *NuevoUsuario.h* y *NuevoUsuario.cpp*.

33. *Páginas:* Directorio de páginas HTML de prueba.

34. *ParStorableComparableStorable:* Clase genérica que implementa una asociación; es decir, un par genérico. Esta clase es subclase de *StorableComparable* y los pares se comparan por la primera componente. Los elementos del par son: la primer componente de tipo *StorableComparable* y la segunda de tipo *Storable*. Los archivos fuentes son *ParStorableComparableStorable.h* y *ParStorableComparableStorable.cpp*.

35. *ParticionadorDeTags:* Esta clase se encarga de "parsear" los marcadores HTML devolviendo por un lado el nombre del mismo y por otro la secuencia de parámetros que pudieran tener. Archivos *ParticionadorDeTags.h* y *ParticionadorDeTags.cpp*.

36. *Perfil:* Subclase de *Storable* que almacena los datos de un perfil de filtrado. Archivos *Perfil.h* y *Perfil.cpp*.

37. *Prueba 2D FART:* Programa stand-alone que implementa la FART (sección 5.6.10) en dos dimensiones con una interfaz gráfica basada en las Microsoft Foundation Classes [Kruglinski97].

38. *Prueba_2D_KMedias:* Programa stand-alone que implementa el algoritmo de las k-medias (sección 5.3.3) en dos dimensiones con una interfaz gráfica basada en las Microsoft Foundation Classes [Kruglinski97].

39. *QuerandoDB:* Directorio donde están las clases wrappers generadas con el entorno visual de Microsoft Visual C++ para acceder a tablas de MS Access vía manejadores ODBC [Jennings99, Kruglinski97].

40. *Resultado:* Clase abstracta para implementar una forma común de acceder al resultado de la ejecución de un método. Archivos *Resultado.h* y *Resultado.cpp*.

41. *Sigmoideas:* Clase que implementa las funciones sigmoideas para aplicarle el escalado a los vectores de características (capítulo 4). Archivos *Sigmoides.h* y *Sigmoides.cpp*.

42. *SizeTWrapper:* Clase wrapper para los enteros sin signo para que se puedan usar con las colecciones genéricas. Archivos *SizeTWrapper.h* y *SizeTWrapper.cpp*.

43. *Storable:* Clase abstracta raiz de la jerarquía de los objetos contenidos en colecciones genéricas. Archivos *Storable.h* y *Storable.cpp*.

44. *StorableComparable:* Clase abstracta subclase de *Storable* de donde cuelgan las clases que componen colecciones genéricas pero que definen las comparaciones por igual y menor. Archivos *StorableComparable.h* y *StorableComparable.cpp*.

45. *StringWrapper:* Clase que implementa un wrapper de cadenas de caracteres para que se puedan usar con las colecciones genéricas. Archivos *StringWrapper.h* y *StringWrapper.cpp*.

46. *Tabla:* Clase que implementa una colección genérica accedida por hash. Archivos *Tabla.h* y *Tabla.cpp*.

47. *Tear:* Clase que implementa la recuperación de documentos HTML de la WWW. Está basada en un ejemplo del Visual Studio 97 [Kruglinski97].

48. *TestFART:* Subclase de *AplicacionCGI* que sirve de base para la aplicación de prueba de la red neuronal de la Teoría de la Resonancia Adaptativa Difusa (sección 5.6.10). Archivos *TestFART.h* y *TestFART.cpp*.

49. *TestKMediasDocs:* Subclase de *GenericTest* y programa para hacer pruebas de clustering de documentos con el algoritmo de las K-medias (capítulo 6). Archivos *TestKMediasDocs.h* y *TestKMediasDocs.cpp*.

50. *TestKohonenNet:* Subclase de *GenericTest* y programa para hacer pruebas de clustering de documentos con la red neuronal de Kohonen y Contrapropagación (capítulo 6). Archivos *TestKohonenNet.h* y *TestKohonenNet.cpp*.

51. *TestPool:* Directorio que contiene documentos HTML de prueba y resultados de mediciones del capítulo 6.

52. *URLSplitter:* Clase para realizar el parsing y resolución de URLs relativas. Archivos *URLSplitter.h* y *URLSplitter.cpp*.

53. *VectorDeBits:* Subclase de *Resultado* que implementa un vector de booleanos empaquetado con lo cual ocupa $\frac{1}{8}$ de lo que ocuparía en condiciones normales. Archivos *VectorDeBits.h* y *VectorDeBits.cpp*.

54. *VectorDeFloat:* Subclase de *Storable* que implementa un vector ralo de números flotantes de tamaño arbitrario y dinámico. Archivos *VectorDeFloat.h* y *VectorDeFloat.cpp*.

55. *VectorDeFloatDenso:* Subclase de *Storable* que implementa un vector de números flotantes de tamaño arbitrario y dinámico. Archivos *VectorDeFloatDenso.h* y *VectorDeFloatDenso.cpp*.

56. *VectorDeFloatFuzzy:* Subclase de *Storable* que implementa un vector ralo de números flotantes entre 0 y 1 de tamaño arbitrario y dinámico. Archivos *VectorDeFloatFuzzy.h* y *VectorDeFloatFuzzy.cpp*.

57. *VectorRalo:* Clase abstracta subclase de *Storable* que implementa un vector ralo genérico. Archivos *VectorRalo.h* y *VectorRalo.cpp*.

Otras Opciones de Implementación

Existía otra opción para realizar la implementación del agente. Por ejemplo, hacer un ejecutable *stand-alone* basado en la API de Windows.

Otra solución podría haber sido planteada en Java usando una aplicación cliente/servidor basada en sockets [Weber97].

Dentro del CGI, otras opciones ya fueron mencionadas en la sección 8.10.

En lugar de JavaScript, otra opción para la validación de formularios podría haber sido VBScript.

10.3 Evaluación Experimental del Agente

En esta sección, se muestra un ejemplo de aplicación del agente construido a un conjunto de páginas reales de la WWW.

10.3.1 Páginas Usadas

La primer prueba para probar la capacidad del agente implementado en la Web real se hizo creando un perfil llamado *Dummt1*, definido por los siguientes parámetros:

- Parámetro $\alpha = 0.0$
- Velocidad de aprendizaje $\beta = 1.0$ (aprendizaje rápido)
- Parámetro de vigilancia $\rho = 0.9$

- Claves de filtrado: omega, watches, seiko, casio, festina, speedmaster, x33.

La primer sesión de filtrado se disparó desde la URL: *http://www.omega.ch/*, con profundidad 3, cantidad de páginas a recuperar igual a 20.

Los documentos recuperados con sus vectores de características fueron los siguientes[2]:

1. *http://www.omega.ch/*:

   (a) omega 1.000000
   (b) watch 1.000000
   (c) seiko 0.000000
   (d) casio 0.000000
   (e) festina 0.000000
   (f) speedma 1.000000
   (g) x33 1.000000
   (h) ——- 0.000000

   Cluster Ganador: 0

2. *http://www.omega.ch/home.html*

   (a) omega 1.000000
   (b) watch 1.000000
   (c) seiko 0.000000
   (d) casio 0.000000
   (e) festina 0.000000
   (f) speedma 1.000000
   (g) x33 1.000000
   (h) ——- 0.000000

   Cluster Ganador: 0.

3. *http://www.omega.ch/nav_up.html*

   (a) omega 0.017986
   (b) watch 0.017986
   (c) seiko 0.000000
   (d) casio 0.000000
   (e) festina 0.000000
   (f) speedma 0.000000

[2]En el caso del agente se agrega automáticamente una característica más *ficticia* (identificada con guiones). Cuando los documentos poseen alguna de las claves de búsqueda, la característica fictica va en 0.0; por otro lado, cuando los documentos no poseen ninguna de la claves de filtrado, la característica ficticia va en 1.0. Esto permite separar documentos sin las claves de filtrado de documentos con pocas apariciones de las mismas.

(g) x33 0.000000

(h) ——- 0.000000

Cluster Ganador: 1.

4. *http://www.omega.ch/homepages/0900_anna/index.html*

   (a) omega 0.268941

   (b) watch 0.047426

   (c) seiko 0.000000

   (d) casio 0.000000

   (e) festina 0.000000

   (f) speedma 0.000000

   (g) x33 0.000000

   (h) ——- 0.000000

   Cluster Ganador: 1.

5. *http://www.omega.ch/nav_base.html*

   (a) omega 0.017986

   (b) watch 0.017986

   (c) seiko 0.000000

   (d) casio 0.000000

   (e) festina 0.000000

   (f) speedma 0.000000

   (g) x33 0.000000

   (h) ——- 0.000000

   Cluster Ganador: 1.

6. *http://www.omega.ch/index.html*

   (a) omega 1.000000

   (b) watch 1.000000

   (c) seiko 0.000000

   (d) casio 0.000000

   (e) festina 0.000000

   (f) speedma 1.000000

   (g) x33 1.000000

   (h) ——- 0.000000

   Cluster Ganador: 0.

7. *http://www.omega.ch/Space/index_space.html*

(a) omega 0.000000
(b) watch 0.000000
(c) seiko 0.000000
(d) casio 0.000000
(e) festina 0.000000
(f) speedma 0.000000
(g) x33 0.000000
(h) ——- 1.000000

Cluster Ganador: 2.

8. *http://www.omega.ch/coll_2000/home_coll99.html*

(a) omega 0.017986
(b) watch 0.000000
(c) seiko 0.000000
(d) casio 0.000000
(e) festina 0.000000
(f) speedma 0.000000
(g) x33 0.000000
(h) ——- 0.000000

Cluster Ganador: 1.

9. *http://www.omega.ch/events/events.html*

(a) omega 0.017986
(b) watch 0.017986
(c) seiko 0.000000
(d) casio 0.000000
(e) festina 0.000000
(f) speedma 0.000000
(g) x33 0.000000
(h) ——- 0.000000

Cluster Ganador: 1.

10. *http://www.omega.ch/sitemap/index_sitemap.html*

(a) omega 0.017986
(b) watch 0.017986
(c) seiko 0.000000
(d) casio 0.000000
(e) festina 0.000000

(f) speedma 0.017986

(g) x33 0.000000

(h) ——- 0.000000

Cluster Ganador: 1.

11. *http://www.omega.ch/sport/index.html*

(a) omega 0.017986

(b) watch 0.000000

(c) seiko 0.000000

(d) casio 0.000000

(e) festina 0.000000

(f) speedma 0.000000

(g) x33 0.000000

(h) ——- 0.000000

Cluster Ganador: 1.

12. *http://www.omega.ch/contact_form/contact_form.html*

(a) omega 0.268941

(b) watch 0.017986

(c) seiko 0.000000

(d) casio 0.000000

(e) festina 0.000000

(f) speedma 0.000000

(g) x33 0.000000

(h) ——- 0.000000

Cluster Ganador: 1.

13. *http://www.omega.ch/blakexpeditions/index.html*

(a) omega 0.047426

(b) watch 0.000000

(c) seiko 0.000000

(d) casio 0.000000

(e) festina 0.000000

(f) speedma 0.000000

(g) x33 0.000000

(h) ——- 0.000000

Cluster Ganador: 1.

14. *http://www.omega.ch/help/help.html*

(a) omega 0.047426
(b) watch 0.017986
(c) seiko 0.000000
(d) casio 0.000000
(e) festina 0.000000
(f) speedma 0.000000
(g) x33 0.000000
(h) ——- 0.000000

Cluster Ganador: 1.

15. *http://www.omega.ch/distrib.html*

(a) omega 1.000000
(b) watch 1.000000
(c) seiko 0.000000
(d) casio 0.000000
(e) festina 0.000000
(f) speedma 0.000000
(g) x33 0.000000
(h) ——- 0.000000

Cluster Ganador: 3.

16. *http://www.omega.ch/company2000/index.html*

(a) omega 0.000000
(b) watch 0.000000
(c) seiko 0.000000
(d) casio 0.000000
(e) festina 0.000000
(f) speedma 0.000000
(g) x33 0.000000
(h) ——- 1.000000

Cluster Ganador: 2.

17. *http://www.omega.ch/company2000/index.html*

(a) omega 0.000000
(b) watch 0.000000
(c) seiko 0.000000
(d) casio 0.000000
(e) festina 0.000000

(f) speedma 0.000000

(g) x33 0.000000

(h) ——- 1.000000

Cluster Ganador: 2.

10.3.2 Desarrollo de la Sesión de Filtrado

La calificación de la primer página era, obviamente por tener un perfil recién creado, desconocida. Por lo tanto, se la calificó como *Buena* a mano. El agente aprendió de esta manera que los documentos que formaran parte del cluster número 0 serían relevantes.

La siguiente también fue calificada a mano como *Buena*, también indicando que es relevante.

Luego, se disparó al agente en forma automática y clasificó (como era de esperarse) correctamente todos los documentos en estos dos clusters.

Se calificó como *Malos* a los documentos de los clusters 2 y 3. Una cualidad interesante de estos documentos es que a pesar de no poseer ninguna de las claves de filtrado en la forma de texto, *sí* las poseen como parte de fotografías con lo cual un usuario las podría calificar como relevantes mientras que el agente haría lo opuesto. Este caso se podría inscribir como una instancia de caso dificultoso de filtrado ya que la característica que hace que el usuario decida la relevancia o irrelevancia del documento no está representada en el vector de características (ver 3.12).

10.3.3 Muestra de Entrenamiento con Documentos Irrelevantes

Cuando se realizaron pruebas sobre documentos irrelevantes (siguiendo directivas de [Skapura96] quien dice que *el conjunto de entrenamiento debe estar formado por una mitad de patrones relevantes y otra mitad de irrelevantes*), se obtuvieron los siguientes resultados:

El primer documento de la serie clasificado fue *vcg01.htm*, el agente contestó que no lo podía calificar y se le enseñó que era irrelevante. Este documento tuvo el vector de características:

1. omega 0.000000
2. watch 0.000000
3. seiko 0.000000
4. casio 0.000000
5. festina 0.000000
6. speedma 0.000000
7. x33 0.000000
8. ——- 1.000000

y el cluster asignado a su firma fue el 2.

Luego, el resto de los documentos[3] fueron clasificados en el mismo cluster (y, por lo tanto, también irrelevantes).

El único error cometido por el agente fue con el documento *vcg16.htm* que fue incorrectamente clasificado como relevante. El error surgió porque el documento poseía una de las claves de filtrado de los documentos de relojes. El cluster de este documento fue el 1 y tuvo el siguiente vector de características.

1. omega 0.000000
2. watch 0.047426
3. seiko 0.000000
4. casio 0.000000
5. festina 0.000000
6. speedma 0.000000
7. x33 0.000000
8. ——- 0.000000

10.4 Escalabilidad

Una solución se dice escalable si tiene un comportamiento eficiente frente a un número creciente de requerimientos.

Existe una razón a favor de la escalabilidad de la implementación de *Querando!* y dos en contra.

La razón a favor de la escalabilidad está dada por el aprovechamiento de las rutinas en JavaScript para hacer validación de la entrada del usuario en los formularios HTML. De esta manera, los ciclos de CPU para determinar si la entrada de usuario es correcta se ejecutan en la máquina del cliente y no sobrecargan al servidor remoto donde reside la aplicación.

Las razones en contra de la escalabilidad son:

1. La naturaleza del CGI con el procesamiento remoto en el servidor apunta en contra de la escalabilidad ya que el filtrado de los documentos HTML implica un elevado tiempo de procesamiento (debido a las dimensiones de los vectores de la red neuronal) y éste es realizado en un servidor central.

2. En particular, esta implementación está basada en aplicaciones de consola Win-32. Esto quiere decir que cada vez que hay un requerimiento se dispara un programa nuevo que consumirá recursos no sólamente de cómputo sino también de memoria. En teoría las aplicaciones basadas en 32 bits sobre Windows se ejecutarán en un ambiente multitarea "preemptivo" con lo cual cada una obtendrá su porción de

[3]El resto de los documentos fueron los siguientes: *vcg02.htm, vcg03.htm, vcg04.htm, vcg05.htm, vcg06.htm, vcg07.htm, vcg08.htm, vcg09.htm, vcg10.htm, vcg11.htm, vcg12.htm, vcg13.htm, vcg14.htm, vcg15.htm, vcg17.htm, vcg18.htm* y *vcg19.htm*.

tiempo de CPU para cumplir con su requerimiento. Implementaciones más eficientes pueden lograrse usando ISAPI (ver sección 8.10) donde los scripts CGI se codifican en DLL en vez de en ejecutables.

La razón a favor de la solución puede hallarse en el hecho de que si la máquina del cliente es mucho más lenta que la del servidor central y éste no se halla recargado de requerimientos, la solución basada en web será mucho más eficiente que la solución basada en un ejecutable *stand-alone*.

Con respecto a la escalabilidad desde el punto de vista del acceso a la base de datos se puede decir que si bien la implementación no está ligada a ningún producto en particular (ya que está basada en ODBC), es demasiado lenta ya que el tiempo de acceso de Access deja mucho que desear. Los programas fueron codificados con manejo de transacciones (locking de tablas y rollbacks en caso de fallas) para permitir a libre concurrencia de procesos.

En particular, no se hicieron pruebas para medir la escalabilidad de la solución ya que solamente se ejecutó en el servidor de web personal de Windows 98.

10.5 Características del Agente

Aquí, se dan las características del agente *Querando!* con los parámetros definidos por [Huhns97a].

Al enumerar las características de los agentes, Huhns [Huhns97a, p. 2–3], las divide en cuatro categorías (sección 2.4). Esta taxonomía se aplicará para comprender a QUERANDO! y ver qué cualidades posee.

Se enumeraran las características de los agentes y se determinará cómo es QUERANDO!.

- Características Intrínsecas
  - *Lifespan*: El agente es long-lived, a pesar de que su ejecución dura lo que dura la clasificación, los datos producidos por la corrida son persistentes.
  - Nivel de cognición: Como está implementado con una red neuronal, diremos que es reactivo.
  - Construcción: La construcción del agente es procedural.
  - Movilidad: Estacionario.
  - Adaptabilidad: Es enseñable (para comprender qué le gusta al usuario) y es autodidacta (porque sabe hacer clustering el sin intervención del usuario).
  - Modelado: Tiene un modelo de él mismo y de algunos recursos de la WWW, como los buscadores.
- Características Extrínsecas
  - Localidad: La localidad es remota (si se lo ve desde el punto de vista del usuario).
  - Autonomía social: La autonomía es controlada.
  - Sociabilidad: Es un jugador de equipo.

  - Amigabilidad: Es cooperativo.
  - Interacciones: Trabaja vía mediadores (como los buscadores).

- Características del Sistema
  - Unicidad: Es homogéneo.
  - Estructura de control: Reactivo.

- Características del *Framework*
  - Autonomía de diseño: Es independiente de la plataforma y del lenguaje de programación. No del protocolo de interacción (CGI/HTTP).
  - Infraestructura de comunicación: Basado en mensajes, asincrónico.
  - Protocolo de mensajes: HTTP y HTML.
  - Servicios de mediación: Transaccionales.
  - Servicios de seguridad: No tiene (leve autenticación).
  - Soporte de operaciones: Archivos y bases de datos ODBC.

- Características del Ambiente del Agente
  - Conocible: Sólo conoce las páginas Web que le da el usuario como índice, sólo los buscadores para los que está programado (se puede extender para que el usuario le explique como interactuar con otros buscadores a través de alguna interfaz).
  - Predecible: Con ninguna.
  - Controlable: Ninguna.
  - Histórico: Sí, la clasificación depende de lo que haya dicho el usuario hasta ahora.
  - Teleológico: Sí, los buscadores.
  - En tiempo real: Sí, las páginas de la Web pueden darse de baja, los servidores pueden caerse.

10.6 Comparaciones con Trabajos Relacionados

En la sección 9.3 se describieron agentes que realizan funciones similares a las de *Querando!*. Con respecto a ello caben las siguientes consideraciones:

1. El filtrador de noticias de Bigus (sección 9.3.2), al utilizar los modelos de redes neuronales de propagación hacia atrás y red de Kohonen, requiere reentrenamiento cuando cambia el escenario de uso. La red FART, al resolver el dilema de estabilidad-plasticidad no tiene este inconveniente. El filtrados de Bigus está implementado como una aplicación stand-alone; esto tiene dos consecuencias:
   - puede incrementar la eficiencia del agente ya que el usuario no tiene que compartir CPU con otros usuarios

- no tiene la flexibilidad de la implementación basada en web en cuanto al acceso desde cualquier lugar del mundo.

2. En cuanto a las capacidades de ser un *spider*, *Querando!* es capaz de recorrer la estructura de grafo de la WWW. Sin embargo, en contraposición a Harvest (sección 9.3.3), éste no fue el objetivo principal del trabajo. Si bien el sistema almacena las páginas visitadas, ésto sólo se implementó como una necesidad a la hora de trabajar sin conexión a la red; además, el sistema sólo almacena documentos HTML y de texto, dejando de lado fotos, películas, etc.

3. En contraposición a los agentes basados en el clasificador bayesiano (veáse secciones 9.3.8 y 9.3.13), el uso de la red neuronal FART elimina la necesidad de construir diccionarios de frecuencias de apariciones de términos en la WWW.

4. La función del agente puede pensarse como un caso particular del agente de mantenimiento de directorios de recursos de Cohen (sección 9.3.9), ya que *Querando!* es capaz de aprender un concepto a través de la ejemplificación del mismo mediante la presentación de documentos. Sin embargo, por la forma en que está planteada la interfaz, *Querando!* no maneja hipótesis de mundo cerrado; esto quiere decir, que si el agente no sabe clasificar un documento no va a contestar que éste es irrelevante sino que va a preguntar una clasificación para el mismo y lo agregará a su base de conocimiento (pesos de la red neuronal).

5. La interfaz de *Querando!* está claramente basada en los trabajos de Michael Pazzani (sección 9.3.13) y de Marko Balabanovic (sección 9.3.8).

6. El tratamiento de los documentos es comparable a las técnicas tradicionales de la IR y no está a la altura de proyectos mucho más ambiciosos como CYC, WordNet y EDR (ver secciones 9.3.12, 9.3.12 y 9.3.12).

10.7 Resumen

En este capítulo se presentó a *Querando!*, un agente de clasificación y filtrado de páginas web que implementa la interfaz del modelo de filtrado de documentos HTML basado en la Teoría de la Resonancia Adaptativa. Además, se discutieron la interfaz, implementación y eficacia del mismo. También, se comparó este trabajo con los agentes de filtrado de información hallados en la literatura.

Capítulo 11

Conclusiones

En este capítulo final se exponen las que se piensa son las contribuciones de este trabajo, se discute la relación del mismo con los temas estudiados durante la carrera y se propone una línea de investigación para el futuro.

11.1 Contribuciones

Este trabajo se realizó con dos objetivos principales:

1. Estudiar la representación general de documentos de texto y en formato HTML orientada a la clasificación de los mismos en clusters.

2. Estudiar la tecnología de agentes e implementar un prototipo operacional que sirviera para ayudar al usuario en la tarea de filtrar un flujo de documentos.

Con respecto a los dos objetivos citados, las contribuciones de este trabajo de grado son las siguientes.

Con respecto al primer objetivo:

- Se planteó una nueva representación de documentos basada en la extracción de características de los mismos usando una contabilización de los trigramas que conforman los mismos. Esta representación presenta diversas variantes basadas en la cantidad de símbolos usados, en la eliminación de palabras stop, aplicación de stemming y en el uso de los marcadores HTML dentro del documento como ser los meta tags con información de claves de búsqueda.

- Se realizaron mediciones experimentales con la representación mencionada para estudiar medidas de similitud aplicadas a la separación de documentos en clases.

- Se realizaron mediciones experimentales usando la mencionada representación para estudiar el clustering de documetos. Dichas mediciones implicaron el uso de los algoritmos de clustering k-medias, de redes neuronales de contrapropagación, mapa autoorganizativo de Kohonen y teoría de la resonancia adaptativa difusa.

Con respecto al segundo objetivo:

- Se enumeraron las características fundamentales de las aplicaciones basadas en el paradigma de agentes y se hizo una clasificación de las mismas.
- Se hizo un estudio de los trabajos relacionados en el tema.
- Se implementó un agente filtrador de documentos basado en una arquitectura cliente/servidor. Este programa está basado en el protocolo CGI y la interacción con el usuario se hace a través de un conjunto de formularios HTML, los documentos a clasificar y un conjunto de programas instalados en un servidor remoto.

11.2 Relación de este Trabajo con la Carrera

Este trabajo de grado en el tema *Agentes* tiene la virtud de haber integrado, ampliado y actualizado muchos de los conocimientos obtenidos en las materias de la carrera de grado, a saber:

- *Programación Orientada a Objetos*: El haber trabajado con un lenguaje de programación orientado a objetos, como C++, y haber estudiado otras opciones, como ser Java y Delphi, me ha permitido profundizar en las técnicas de programación orientada a objetos. Dichas técnicas incluyen la creación de colecciones heterogéneas y la estructuración de un programa en clases reusables.
- *Estructuras de Datos*: El haber estudiado el área de recuperación de información junto con sus estructuras de datos y sus algoritmos relacionados, me ha hecho porfundizar en esta área.
- *Verificación de Programas*: El estudio de arquitecturas de agentes con información temporal y semántica formal me ha permitido aplicar los conocimientos de esta materia.
- *Bases de Datos* y *Bases de Datos Orientadas a Objetos*: La aplicación y profundización de estos conceptos se usaron en el estudio de diversos agentes. Se profundizó su aplicación y utilización concreta en la implementación del agente aplicación de este trabajo de grado.
- *Ingeniería de Software*: Si bien este trabajo sólo implicó la construcción de un prototipo, sirvió para considerar lo aprendido en dicha materia.
- *Lógica* e *Inteligencia Artificial*: En general, la teoría de agentes es una aplicación de las tecnologías del área de la Inteligencia Artificial de los últimos cuarenta años.
- *Redes Neuronales*: La aplicación es directa. Se profundizó en el estudio de arquitecturas alternativas a las estudiadas en clase.
- *Diseño de Compiladores*: El procesamiento de los archivos HTML se realizó con los conocimientos adquiridos en esta materia.
- *Redes*: El estudio de los protocolos TCP/IP y HTTP, es una profundización y extensión de los estudiado en la materia Seminario de Redes.
- *Hipermedia*: El estudio del lenguaje HTML es una profundización de los cursos de grado sobre Hipermedia.

11.3 Trabajo Futuro

Hay una variedad de direcciones en las cuales esta investigación puede orientarse en el futuro. Estas se pueden separar en dos clases: representación y clasificación de documentos y estudio posterior e implementación de otros agentes y sistemas multiagentes.

Con respecto a la representación y clasificación de documentos, se puede decir:

1. En este trabajo de grado, se dejaron deliberadamente de lado otros formatos de documentos que son igualmente interesantes para el filtrado. Entre los formatos dejados de lado podemos encontrar al formato de texto enriquecido (RTF), Postcript (PS), Abobe Portable Document Format (PDF), DVI, etc. Tampoco se consideraron los formatos comprimidos, por ejemplo: ARJ, ZIP, etc.

2. Tampoco fueron tenidas en cuenta características tales de los documentos como el contenido multimedia, compuesto por fotos, películas y sonidos. Estas características bien podrían formar parte del perfil del usuario.

3. Las técnicas de datamining también se podrían aplicar a la obtención de información en grandes bases de documentos [Glymour96].

Con respecto a la implementación de agentes y sistemas multiagentes, se puede decir:

1. Otra implementación posible del agente *Querando!* se podría haber basado en un programa stand-alone, esta es una posilidad a explorar en el futuro.

2. La implementación actual del agente está orientada a un usuario simple. Una forma de extender este trabajo puede orientarse al filtrado cooperativo haciendo que varios usuarios den forma a un perfil de filtrado.

3. Otra forma en la que se puede extender este trabajo es en la forma de la implementación de un sistema de filtrado multiagente en que, en vez de haber un único programa que realice el filtrado de los documentos, podría haber una multitud de los mismos realizando la tarea en mucho menor tiempo.

Finalmente, las aplicaciones de agentes en Internet parecen casi infinitas a medida que cada vez más gente tenga acceso a dicho mar de información con la retroalimentación que esto produce al generarse más contenido.

Apéndice A

Tablas de Stop Words y Stems

En este apéndice aparecen tablas de capítulos precedentes. En dichos capítulos, estas tablas no se incluyeron por una cuestión de espacio y que, por sus dimensiones, dificultaban la lectura fluida del texto.

Las tablas de este apéndice contienen información sobre (el detalle de cada una está en el índice de tablas de la página 11):

- Lista de palabras stop (*stop words*) usadas en la implementación del agente *Querando!* y en las mediciones experimentales de clustering de documentos. Ver tablas A.1, A.2 y A.3.
- Lista de *stems* usados en el algoritmo de stemming del agente *Querando!* y en las mediciones experimentales de clustering de documentos. Ver tabla A.4.

a	about	above	across
after	again	against	albeit
all	almost	alone	along
already	also	although	always
among	amongst	an	and
another	any	anybody	anyhow
anyone	anything	anywhere	are
area	areas	around	as
ask	asked	asking	asks
at	away	b	back
backed	backing	backs	be
became	because	become	becomes
becoming	been	before	beforehand
began	behind	being	beings
below	beside	besides	best
better	between	beyond	big
both	but	by	c
came	can	can	cannot
case	cases	certain	certainly
clear	clearly	come	could
d	did	differ	different
differently	do	do	does
done	down	downed	downing
downs	during	e	each
early	either	else	elsewhere
end	ended	ending	ends
enough	etc	even	evenly
ever	every	everybody	everyone
everything	everywhere	except	f
face	faces	fact	facts
far	felt	few	find
finds	first	for	former
formerly	four	from	full
fully	further	furthered	furthering
furthers	g	gave	general
generally	get	gets	give
given	gives	go	going
good	goods	got	great
greater	greatest	group	grouped
grouping	groups	h	had
has	have	having	he

Tabla A.1: Lista de *stop words* (primera parte).

hence	her	here	hereafter
hereby	herein	hereupon	hers
herself	high	higher	highest
him	himself	his	how
however	i	ie	if
important	in	inc	indeed
interest	interested	interesting	interests
into	is	it	its
itself	j	just	k
keep	keeps	kind	knew
know	known	knows	l
large	largely	last	later
latest	latterly	latter	least
less	let	lets	like
likely	long	longer	longest
ltd	m	made	make
making	man	many	may
me	meanwhile	member	members
men	might	more	most
mostly	mr	mrs	much
must	my	myself	n
namely	necessary	need	needed
needing	needs	neither	never
nevertheless	new	newer	newest
next	no	nobody	nobody
non	none	noone	nor
not	nothing	now	nowhere
number	numbered	numbering	numbers
o	of	off	often
old	older	oldest	on
once	one	only	open
opened	opening	opens	or
order	ordered	ordering	orders
other	others	our	out
over	p	part	parted
parting	parts	per	perhaps
place	places	point	pointed
pointing	points	possible	present
presented	presenting	presents	problem
problems	put	puts	q
quite	r	rather	really

Tabla A.2: Lista de *stop words* (segunda parte).

right	room	rooms	s
said	same	saw	say
says	second	seconds	see
seem	seemed	seeming	sees
semms	several	shall	she
should	show	showed	showing
shows	side	sides	since
small	smaller	smallest	so
some	somebody	somehow	someone
something	sometime	sometimes	somewhere
state	states	still	such
sure	t	take	taken
than	that	the	their
them	then	there	therefore
these	they	thing	things
think	those	though	thought
thoughts	three	throughthus	thru
thus	to	today	together
too	took	toward	towards
turn	turned	turning	turns
two	u	under	until
up	upon	us	use
used	uses	v	very
via	w	want	wanted
wanting	wants	was	way
ways	we	well	wells
went	were	what	whatever
whatsoever	when	whence	whenever
whensoever	where	whereabouts	whereafter
whereas	whereat	whereby	wherefrom
wherein	whereinto	whereof	whereon
whereto	whereunto	whereupon	wherever
wherewith	whether	which	whichever
whichsoever	while	whilst	who
whoever	whole	whom	whomever
whomsoever	whose	whosoever	why
will	with	within	without
work	worked	working	works
would	x	y	yet
young	younger	youngest	your
yours	yourself	yourselves	z

Tabla A.3: Lista de *stop words* (tercera parte).

ability	able	ade	age
aholic	al	an	ana
ean	ian	ance	ence
ancy	ency	ant	ent
ar	archy	ard	arian
ard	arian	ary	ate
athon	ation	ative	ator
bound	cide	cracy	craft
crat	cy	d	dom
drome	ean	ectomy	ed
ee	eer	en	ence
ency	ent	er	eyr
es	ese	esque	ess
est	eth	ette	ey
fashion	fold	free	friendly
ful	fy	gamy	genarian
gon	gram	head	high
hood	i	is	ial
ian	iana	ibility	ible
ic	ical	ician	icide
ics	ide	ie	iform
ify	in	ine	ing
ion	ise	ish	ism
ist	ite	itis	itude
ity	ive	ize	ise
kin	latry	led	less
let	like	ling	lived
logist	logue	logy	ly
manship	ment	meter	metre
monger	most	nd	ness
nik	ocracy	ocrat	oid
ologist	ology	or	ory
osis	ous	phile	philia
philiac	phobe	phobia	phobic
proof	rd	ridden	ry
s	’s	s’	scape
ship	smith	some	speak
sphere	spoken	st	ster
th	tion	tude	ty
ular	ule	ure	ville
ward	wards	ware	ways
wright	y		

Tabla A.4: Lista de *stems*

Apéndice B

Códigos de Respuesta HTTP y Variables CGI

Este apéndice contiene tablas concernientes a los siguientes puntos:

- Códigos de respuesta HTTP. Ver tablas B (B.1), B (B.2), B (B.3), B (B.4) y B (B.5).
- Lista de variables de ambiente CGI. Ver tabla B (B.6).

Dígito	*Tipo*	*Descripción*
1xx	Informacional	No usado, pero reservado para uso futuro.
2xx	Exitoso	La acción fue exitosamente recibida, entendida y aceptada.
3xx	Redirección	Una acción subsiguiente debe tomarse para completar el requerimiento.
4xx	Error del cliente	El requerimiento contenía una sintaxis incorrecta o no puede llevarse a cabo.
5xx	Error del servidor	El servidor falló en llevar a cabo un requermiento aparentemente válido.

Tabla B.1: Clases de Códigos de Respuesta HTTP.

Código de Estado	*Explicación*
`200 Ok`	El requerimiento ha sido llevado a cabo y una entidad correspondiente al recurso requerido se envió en respuesta.
`201 Creado`	El requerimiento ha sido llevado a cabo y resultó en la creación de un nuevo recurso.
`202 Aceptado`	El requerimiento se aceptó para su procesamiento, pero el procesamiento no ha sido completado.
`203 Información Provisional`	La meta información retornada en el encabezado de la entidad no es el conjunto definitivo como estaba disponible en el servidor de origen, pero se conseguirá de un local o de un tercero.
`204 Sin Contenido`	El servidor satisfizo el requerimiento, pero no se envió ninguna información de retorno.

Tabla B.2: Códigos 2xx Exitosos

Código de Estado	*Explicación*
`300 Elecciones múltiples`	El requerimiento requerido está disponible en una o más locaciones y la locación no puede determinarse a través de negociaciones de contenido.
`301 Movido Permanentemente`	El recurso requerido ha sido asignado a una nueva y permanente URI, y todas las futuras referencias a este recurso deben hacerse usando la URI retornada.
`302 Movido Temporariamente`	El recurso requrido reside temporariamente en una URI diferente.
`303 Método`	El método es obsoleto.
`304 No Modificado`	Si el cliente ha realizado un requerimiento GET condicional y el acceso está permitido, pero el documento no ha sido modificado desde los fecha y día especificados el campo *If-modified-Since*; el servidor responderá con este código y no enviará el cuerpo de ninguna entidad al cliente.

Tabla B.3: Códigos 3xx Redirección

Código de Estado	*Explicación*
`400 Requerimiento Mal`	El requerimiento tenía sintaxis incorrecta o era inherentemente imposible de satisfacer.
`401 No Autorizado`	El requerimiento requiere autenticación del usuario.
`402 Se Requiere un Pago`	Este código no está soportado actualmente.
`403 Prohibido`	El requerimiento está prohibido por alguna razón desconocida para el cliente.
`404 No Hallado`	El servidor no ha hallado nada que concuerde con la URI requerida.
`405 Método no Permitido`	El método especificado en la línea de requerimiento no está permitido para el recurso identificado por la URI del requerimiento.
`406 Ninguno Aceptable`	El servidor ha hallado un recurso concordando con la URI requerida, pero no uno que satisfaga las condiciones identificadas por los encabezados de requerimientos *Accept* y *Accept-Encoding*.
`407 Se Requiere Autenticación del Proxy`	Reservado para uso futuro.
`408 Timeout del Requerimiento`	El cliente no produjo un requerimiento en el tiempo que el servidor estaba preparado para esperar.
`409 Conflicto`	El requerimiento no puede completarse debido a un conflicto con el estado corriente del recurso.
`410 Se Fue`	El recurso requerido no está más disponible en el servidor ni se conoce una dirección de *forward*.

Tabla B.4: Códigos 4xx Error del Cliente

Código de Estado	*Explicación*
`500 Error Interno del Srervidor`	El servidor encontró una condición inesperada que previno que llevara a cabo el requerimiento.
`501 No Implementado`	El servidor no soporta la funcionalidad requerida para llevar a cabo el requerimiento.
`502 Gateway Incorrecto`	El servidor recibió una respuesta inválida del gateway que accedió en su intento de completar el requerimiento.
`503 Servicio No Disponible`	El servidor e incapaz de manejar el requerimiento debido a una sobrecarga temporaria o mantenimiento del servidor.
`504 Timeout del Gateway`	El servidor no recibió una respuesta en tiempo del gateway o servidor corriente arriba[1] en su intento de completar el requerimiento.

Tabla B.5: Códigos 5xx Error del Servidor

Nombre de la variable	*Descripción*
AUTH_TYPE	Método de autenticación usado para validar al usuario.
CONTENT_LENGTH	Longitud de los datos (en bytes o número de caracteres) pasados al programa CGI via entrada standard.
CONTENT_TYPE	Tipo MIME de los datos enviados al programa CGI. Por ej.: TEXT/HTML.
DOCUMENT_ROOT	Directorio raíz del servidor de Web.
GATEWAY_INTERFACE	Versión de CGI soportada por el servidor de Web.
HTTP_ACCEPT	Tipos de datos MIME que la máquina cliente puede aceptar.
HTTP_FROM	Dirección de e-mail del usuario haciendo el requerimiento (si el Web browser lo provee).
HTTP_REFERER	URL del documento HTML conteniendo la etiqueta FORM que activó el programa CGI.
HTTP_USER_AGENT	Nombre y versión del Web browser usado como cliente.
PATH_INFO	Información extra de camino pasada al programa CGI.
PATH_TRANSLATED	Versión traducida de la información en la variable PATH_INFO.
QUERY_STRING	Información de consulta pasada al programa CGI.
REMOTE_ADDR	Dirección IP de la máquina corriendo el Web browser que originó el requerimiento de CGI.
REMOTE_HOST	Nombre del anfitrión (*host*) de la máquina corriendo el Web browser que originó el requerimiento de CGI.
REMOTE_IDENT	Usuario haciendo el requerimiento (raramente usado).
REMOTE_USER	Nombre de usuario autenticado.
REQUEST_METHOD	Método usado para el requerimiento CGI (GET o POST).
SCRIPT_NAME	Ubicación del script a ser ejecutado.
SERVER_NAME	Nombre del servidor anfitrión o dirección IP.
SERVER_PORT	Número de puerto usado por el servidor de Web.
SERVER_PROTOCOL	Nombre y versión del protocolo de información usado en el requerimiento CGI.
SERVER_SOFTWARE	Nombre y versión del software del servidor de Web.

Tabla B.6: Variables de ambiente CGI

Bibliografía

[JS99] *Client-Side JavaScript Reference. Version 1.3* Netscape Communications Corporation, 1999.

[Aarsten96] Aarsten, A.; Brugali, D.; Vlad, C. *Cooperation among Autonomous Agents.* Dept. of Automatica e Informatica. Politecnico di Torino, Italy. (1996).

[Aho83a] Aho, Alfred V.; Sethi, Ravi; Ullman, Jeffrey D. *Compilers. Principles, Design and Tools.* Addison-Wesley Publising Co. Reading, Massachusetts, 1983.

[Aho83b] Aho, A.; Hopcroft, J,; Ullman, J. *Data Structures and Algorithms.* Addison-Wesley. 1983.

[Alta98] *AltaVista Search Engine Help.* http://www.altavista.com.

[Baeza92] Baeza-Yates, R. *Introduction to Data Structures and Algorithms Related to Information Retrieval.* In Information Retrieval Data Structures and Algoritms, W. Frakes & R. Baeza-Yates eds., págs. 13–27. Prentice Hall, 1992. Upper Saddle River, New Jersey.

[Bakken99] Bakken, Stig Sæther; Aulbach, Alexander; Schmid, Egon ; Winstead, Jim; Wilson, Lars Torben; Lerdorf, Rasmus; Suraski, Zeev. *PHP3 Manual.* The PHP Documentation Group, 1999.

[Balabanovic98] Balabanovic, M; Shoham, Y.; Yun, Y. *An Adaptive Agent for Automated Web Browsing.* 1998 http://elib.stanford.edu/Dienst/UI/2.0/Describe/stanford.cs

[Bates97] Bates, J.; Loyall, A.; Reilly, S. *An Architecture for Action, Emotion, and Social Behavior.* In Readings in Agents. Págs. 225–231. 1997. M. Hunhs & M. Singh, eds. Morgan Kauffman.

[Bayardo98] Bayardo, R. J.; Bohrer, W.; Brice, R.; Cichocki, A.; Fowler, J.; Helal, A.; Kashyap, V.; Ksiezyk, T.; Martin, G.; Nodine, M.; Rashid, M.; Rusinkiewicz, M.; Shea, R.; Unnikrishnan, C.; Unruh, A.; Woelk, D. *InfoSleuth: Agent-Based Semantic Integration of Information in Open and Dynamic Environments.* Microelectronics and Computer Technology Corporation (MCC). http://www.mcc.com/projects/infosleuth. Pp. 205–216. Readings in Agents. Hunhs, M. & Singh, M. Editores. Morgan Kaufmann. 1998.

[Bert98] Bert, B.; Lie, H.; Lilley, C.; Jacobs, I. *Cascading Style Sheets, level 2 CSS2 Specification.* W3C Recommendation, 12-May-1998. `http://www.w3.org/TR/1998/REC-CSS2-19980512`.

[Bigus98] Bigus, J. P.; Bigus, J. *Constructing Intelligent Agents with Java.* 1998. Wiley Computer Publishing. Ed. John Wiley & Sons. ISBN: 0-471-14135-3.

[Billsus97] Billsus, D; Pazzani, M. *Learning Probabilistic User Models.* In workshop notes of Machine Learning for User Modeling, Sixth International Conference on User Modeling, Chia Laguna, Sardinia, 2-5 June 1997.
`http://www.ics.uci.edu/~pazzani/Publications/ProbUserModels.ps`

[Brin2000] Brin, Sergey; Page, Lawrence. *The Anatomy of a Large-Scale hypertextual Web Search Engine.* Stanford University, Stanford, 2000.
`http://www7.scu.edu.au/programme/fullpapers/1921/com1921.htm`

[Brokken95] Brokken, F.; Kubat, K. *C++ Annotations.* 1995. ICCE, State university of Gronningen, Netherlands. ISBN 90 367 0470 7.

[Carter97] Carter, E. F. Jr. *Robots and Finite-State Machines.* Dr. Dobb's Journal. February 1997. Ed. Miller Freeman.

[Clancey93] Clancey, W. *The Level Reinterpreted: Modeling Socio-Technical Systems.* International Journal of Intelligent Systems, Vol. 8, 33-49. 1993. John Wiley & Sons, Inc. CCC 08884-8173/93/010033-17.

[Cohen96a] Cohen, P.; Cheyer, A.; Wang, M.; Baeg, S. *An Open Agent Architecture.* Pp. 197–204. Readings in Agents. Hunhs, M. & Singh, M. Editores. Morgan Kaufmann. 1998.

[Cohen96b] Cohen, W.; Singer, N. *Learning to Query the Web.* In The 1996 AAAI Workshop on Internet-Based Information Systems. 1996.
`http://www.research.att.com/~wcohen/postscript/aaai-ws-96.ps`

[Cheong96] Cheong, F. C. *Internet Agents: Spiders, Wanderers, Brokers, and Bots.* 1996. Ed. New Riders Publishing.

[Church95] Church, K.; Rau, L. *Commercial Applications of Natural Language Processing.* Communications of the ACM. November 1995. Vol. 38, No. 11.

[Cutkosky97] Cutkosky, M.; Engelmore, R.; Fikes, R.; Genesereth, M.; Mark, W.; Tenenbaum, J.; Weber, J. *PACT: An Experiment in Integrating Concurrent Engeneering Systems.* 1997. Readings in Agents, págs. 46–55. Morgan Kaufmann.

[Devore95] Devore, Jay. *Probability and Statistics for Engineering and the Sciences. Fourth Edition.* Puxbury Press, Brooks/Cole Publishing Co. Pacific Grove, CA., USA, 1995.

[Duncan96] Duncan, Ray. *How the Common Gateway Interface Works.* PC Magazine Vol. 15 No.16. September 1996. Págs. 207-214.

[Durfee97] Durfee, E.; Kiskis, D.; Birmingham, W. *The agent architecture of the University of Michigan Digital Library.* 1997. Readings in Agents, págs. 98–108. Morgan Kaufmann.

[Dwight97] Dwight, J; Erwin, M.; Niles, R. *Using CGI. Second Edition.* QUE Corp. 1997.

[Edmonds94] Edmonds, E.; Candy, L.; Jones, R.; Soufi, B. *Support for Collaborative Design: Agents and Emergence.* Communications of the ACM. July, 1994. Vol. 17, No. 7.

[Etzioni97] Etzioni, O.; Weld, D. *A Softbot-Based Interface to the Internet.* 1997. Readings in Agents, págs. 77–81. Morgan Kaufmann.

[Fischer97] Fischer, K.; Muller, J.; Pischel, M. *A Pragmatic BDI Architecture.* 1997. In Readings in Agents, págs. 217–224. M. Hunhs & M. Singh, eds. Morgan Kaufmann.

[Ford93] Ford, K.; Bradshaw, J.; Addams-Webber, J.; Agnew, N. *Knowledge Acquisition as a Constructive Modeling Activity.* International Journal of Intelligent Systems, Vol. 8, 9-32. 1993. John Wiley & Sons, Inc. CCC 0884-8173/93/010009-24.

[FoxC92] Fox, C. *Lexical Analysis and Stop Lists.* In Information Retrieval. Págs. 28–43. Frakes, W & Baeza-Yates eds. Prentice-Hall, 1992. Upper Saddle River, New Jersey.

[FoxE92] Fox, E.; Betrabet, S.; Koushik, M.; Lee, W. *Extended Boolean Models* In Information Retrieval. Págs. 393–418. Frakes, W & Baeza-Yates eds. Prentice-Hall, 1992. Upper Saddle River, New Jersey.

[Frakes92a] Frakes, W.; Baeza-Yates, R. *Information Retrieval. Data Structures & Algorithms*. Ed. Prentice Hall. 1992.

[Frakes92b] Frakes, W. *Stemming Algorithms.* En Information Retrieval. Data Structures & Algorithms. W. Frakes & R. Baeza-Yates editores. Págs. 131–160. Ed. Prentice Hall. 1992.

[Freeman93] Freeman, J; Skapura, D. Redes Neuronales. *Algoritmos, aplicaciones y técnicas de programación*. 1993. Ed. Addison-Wesley/Díaz de Santos.

[Gach96] Gach, Gary. *The Pocket Guide to the Internet.* 1996. Pocket Books. Simon & Schuster Inc. New York, USA.

[Genesereth97] Genesereth, M.; Ketchpel, S. *Software Agents*. Logic Group, Computer Science Departement. Stanford University, 1997.

[Genesereth92] Genesereth, M.; Fikes, R. *Knowledge Interchange Format Version 3.0 Reference Manual*. Technical Report Logic-92-1, Computer Science Department, Stanford University, June 1992.
`http://www-ksl.stanford.edu/knowledge-sharing/papers/kif.ps`

[Genesereth94a] Genesereth, M.; Singh, N. *A Knowledge Sharing Approach to Software Interoperation*. 1994. Computer Science Department, Stanford University.

[Genesereth94b] Genesereth, M.; Singh, N.; Syed, M. *A Distributed and Anonymous Knowledge Sharing Approach to Software Interoperation*. 1994. Computer Science Dept. Stanford University. Stanford.

[Girardi98a] Girardi, R. *Software Classification and Retrieval.* Filminas de Curso sobre Tecnologías para Desarrollo de Software en la World Wide Web. Universidad Nacional de La Plata. La Plata. 1998.

[Girardi98b] Girardi, R. *Agentes de la Información.* Filminas de Curso sobre Tecnologías para Desarrollo de Software en la World Wide Web. Universidad Nacional de La Plata. La Plata. 1998.

[Gómez99] Gómez, Sergio A. *Una Taxonomía de Cambios para el Grafo de Espacios de Nombres Contextuales para Sistemas Multi-Agente en Contextos Múltiples.* V Congreso Argentino de Ciencias de la Computación. Tandil, 1999.

[Glymour96] Glymour, C.; Madigan, D.; Pregibon, D.; Smyth, P. *Statistical Themes and Lessons for Data Mining.* Journal of Data Mining and Knowledge Discovery, 1, 25-42 (1996). Kluwer Academic Publishers, Boston.
`http://bayes.stat.washington.edu/PAPERS/dami.ps`

[Gonnet92] Gonnet, G. H.; Baeza-Yates, R.; Snider, T. *New Indices for Text: PAT trees and PAT arrays.* En Information Retrieval. Data Structures & Algorithms. W. Frakes & R. Baeza-Yates editores. Págs. 131–160. Ed. Prentice Hall. 1992.

[Gourney99] Gourney, Kevin. *Computers and Symbols versus Nets and Neurons.* Dept. Human Sciences, Brunel University. Udbridge, Middx., 1999.

[Gulbransen98] Gulbransen, D.; Rawlings, K. *HTML Dinámico. Edición Especial.* QUE, Prentice Hall. 1998.

[Hardy96] Darren R. Hardy; Michael F. Schwartz; Duane Wessels. *Harvest User's Manual. Version 1.4 patchlevel 2.* January 31, 1996. Technical Report CU-CS-743-94. Department of Computer Science. University of Colorado. Boulder, Colorado 80309-0430.
`http://harvest.cs.colorado.edu/`

[Harman92a] Harman, D.; Fox, E.; Baeza-Yates, R.; Lee, W. *Inverted Files.* In Information Retrieval. Págs. 28–43. Frakes, W & Baeza-Yates, R. eds. Prentice-Hall, 1992. Upper Saddle River, New Jersey.

[Harman92b] Harman, D. *Relevance Feedback and Other Query Modification Techniques.* In Information Retrieval. Págs. 241–262. Frakes, W & Baeza-Yates, R. eds. Prentice-Hall, 1992. Upper Saddle River, New Jersey.

[Harper96] Harper, N. *Intelligent Agents and the Internet.* 1996.
`http://osiris/sunderland.ac.uk/cbowww/AI/TEXTS/AGENTS3/agents.htm`

[Heckerman95] Heckerman, D. *A Tutorial on Learning with Bayesian Networks.* 1995. Technical Report. MSR-TR-95-06. Microsoft Research, Advanced Technology Division, Microsoft Corp.
`http://www.research.microsoft.com/DTAS/heckerma/TR-95-06.htm`

[Helman86] Helman, P.; Veroff, R. *Intermediate Problem Solving and Data Structures. Walls and Mirrors.* The Benjaming/Cummnings Publishing Company, Inc. 1986. Redwood City, California, USA.

[Huhns97a] Huhns, M.; Singh, M. *Readings in Agents*. 1997. Morgan Kaufmann Publishers, Inc.

[Huhns97b] Huhns, M.; Singh, M.; Ksiezyk. *Global Information Management via Local Autonomous Agents*. 1997. Readings in Agents, págs. 36–45. Morgan Kaufmann.

[Hyötyniemi96] Hyötyniemi, H. *Text Document Classification with Self-Organizing Maps*. 1996.
`http://www.hut.fi/~hhyotyni/HH3/HH3.ps`

[Jennings99] Jennings, Roger; Hipson, Peter *Database Developer's Guide with Visual C++ 4, Second Edition.* Ed. Macmillan Computer Publishing. 1999. Online version

[Johansen98] Johansen, D.; van Renesse, R.; Schneider, F. *Operating System Support for Mobile Agents.* In Readings in Agents, Michael Huhns & Munindar Singh, eds. 1998, Morgan Kauffman.

[Jones95] Jones, M. *Gofer 2.30 Release Notes.* Technical Report, Yale University, 1995.
`http://www.cs.nott.ac.uk:80/Department/Staff/mpj`

[Kaski97] Kaski, S. *Data Exploration Using Self-Organizing Maps.* PhD. Thesis. Acta Polytechnia Scandinavica. Mathematics, Computing and Management in Engineering Series No. 82. Espoo, Finland, 1997.

[Kaski96] Kaski, S.; Kohonen, T. *Exploratory Data Analysis by the Self-Organizing Map: Structures of Welfare and Poverty in the World*. In Refenes, A.; Abu-Mostafa, Y.; Moody, J.; Weigend, A. (Eds.) Neural Networks in the Capital Markets, London, England, 11-13 October, 1995. World Scientific, Singapore, pp. 498-507, 1996.

[Kernigham85] Kernigham, B.; Ritchie, D. *El lenguaje de programación C*. Ed. Prentice-Hall Hispanoamericana, 1985.

[Korth86] Horth, Henry F.; Silberschatz, Abraham. *Database System Concepts*. McGraw-Hill, 1986.

[Koster93] Martijn Koster. *Guidelines for Robot Writers*. 1993.
`http://info.webcrawler.com/mak/projects/robots/guidelines.html`

[Koster96a] Koster, M. *The Web Robots FAQ*. 1996
`http://info.webcrawler.com/mak/projects/robots/robots.html`

[Koster96b] Martijn Koster. *Evaluation of the Standard for Robots Exclusion*. 1996.
`http://info.webcrawler.com/mak/projects/robots/eval.html`

[Kruglinski97] Kruglinski, D. *Inside Visual C++*. Fourth Edition. 1997. Ed. Microsoft Press.

[Krulwich97] Krulwich, B. *Automating the Internet: Agents as User Surrogates.* IEEE Internet Computing, Vol. 1, No. 4, July-August, 1997. Ed. Institute of Electrical and Electronic Engineers, Inc.

[Kushmerick98] Nicholas Kushmerick. *Wrapper induction: Efficiency and expressiveness.* School of Computer Applications, Dublin City University Draft of May 1, 1998. `http://www.compapp.dcu.ie/~nick/research/download/kushmerick-wi-journal.ps`

[Langley95] Langley, P.; Simon, H. *Applications of Machine Learning and Rule Induction.* Communications of the ACM. November 1995. Vol. 8, No. 11. ACM 0002-0782/95/1100.

[Lashkari97] Lashkari, Y.; Metral, M.; Maes, P. *Collaborative Interface Agents.* 1997. Readings in Agents, págs. 111–116. Morgan Kaufmann.

[Lavoie99] Lavoie, P.; Crespo, J.; Savaria, Y. *Generalization, Discrimination, and Multiple Categorization Using Adaptive Resonance Theory.* IEEE Transactions on Neural Networks, Vol. 10, No. 4, July, 1999. Publisher Item Identifier S 1045-9227(99)05271-8.

[Labrow98] Labrow, Y.; Finin, T. *Semantics and Conversations for an Agent Communication Language.* In Readings in Agents, Michael Huhns & Munindar Singh, eds. 1998, Morgan Kauffman.

[Lamport94] Lamport, Leslie. *A Document Preparation System LaTeX. User's Guide and Reference Manual.* Addison-Wesley Publishing Co., 1994.

[Lenat95a] Lenat, Doug. *CYC: A Large-Scale Investment in Knowledge Infrastructure.* Communications of the ACM. Vol. 38, No. 11. Pp. 33–39. November, 1995.

[Lenat95b] Lenat, D.; Miller, G.; Yokoi, T. *CYC, WordNet, and EDR: Critiques and Responses.* Communications of the ACM. November, 1995. Vol. 38, No. 11.

[Lewis92] Lewis, D. *Representation and Learning in Information Retrieval.* PhD Dissertation Thesis. 1992. `http://portal.research.bell-labs.com/orgs/ssr/people/lewis/lewis91d.ps`

[Lewis95] Lewis, D. *A Brief Overview of Information Retrieval* Appeared in Proceedings IEEE Automatic Speech Recognition Workshop. December 10-13, 1995. Snowbird, Utah, p. 66. (Unpublished workshop notes.). `http://portal.research.bell-labs.com/orgs/ssr/people/lewis/lewis95h.ps`

[Linke98] Lincke, D.; Schmid, B. *Mediating Electronic Product Catalogs.* Communications of the ACM. July, 1998. Vol. 41, No. 7.

[Lloyd87] Lloyd, John. *Foundations of Logic Programming.* Springer-Verlag, 1987.

[Lohse98] Lohse, G.; Spiller, P. *Electronic Shopping.* Communications of the ACM. July, 1998. Vol. 41, No. 7.

[Luger93] Luger, George F.; Stubblefield, William A. *Artificial Intelligence. Structures and Strategies foe Complex Problem Solving. Second Edition.* Benjamin/Cummings Publishing Co. Redwood City, California, 1993.

[Maes94] Maes, P. *Agents that Reduce Work and Information Overload.* Communications of the ACM. July, 1994. Vol. 17, No. 7.

[Maes95] Maes, P. *Artificial Life Meets Entertainment: Lifelike Autonomous Agents.* Communicatios of the ACM. November, 1995. Vol. 38, No. 11.

[Maravall94] Maravall Gómez Allende, Darío. *Reconocimiento de Formas y Visión Artificial*. 1994. Ed. Addison-Wesley Iberoamericana.

[Meyer] Meyer, P. *Probabilidades y Aplicaciones Estadísticas.* Ed. Fondo Educativo Interamericano.

[Miller95] Miller, G. *WordNet: A Lexical Database for English.* Communications of the ACM. November, 1995. Vol. 38, No. 11. ACM 0002-0782/95/1100.

[Mostafa97] Mostafa, J.; Mukhopadhyay, S.; Lam, W.; Palakal, M. *A Multilevel Approach to Intelligent Information Filtering: Model, System, and Evaluation.* ACM Transactions on Information Systems, Vol. 15, No. 4, October 1997, Pages 368–399. ACM 1046-8188/97/1000-0368.
`/pubs/articles/journals/tois/1997-15-4/p368-mostafa/p368-mostafa.pdf`

[Pazzani97a] Pazzani, M.; Billsus, D. *Learning and Revising User Profiles: The Identification of Interesting Web Sites*. Machine Learning 27, 313-331. 1997. Ed. Kluwer Academic Publisers.
`http://www.ics.uci.edu/~pazzani/Publications/SW-MLJ.pdf`

[Pazzani97b] Pazzani, M; Nguyen, L; Mantik, S. *Learning from hotlists and coldlists: Towards a WWW information filtering and seeking agent*. Department of Information and Computer Science. University of California, Irvine. 1997.
`http://www.ics.uci.edu/~pazzani/TAI/ColdListfinal.html`

[Perkowitz96] Perkowitz, Mike; Etzioni, Oren. *Category Translation: Learning to understand information on the Internet.* Seattle, 1996.
`ftp://ftp.cs.washington.edu/pub/ai/`

[Quirk90] Quirk, Randolph. *Longman Dictionary of Contemporary English. New Edition.* 1990. Longman Dictionaries.

[Raggett98] Raggett, D.; Le Hors, A.; Jacobs, I. *HTML 4.0 Specification. W3C Recommendation.* 1998.
`http://www.w3.org/TR/1998/REC-html40-19980424`

[Rao95] Rao, V.; Rao, H. *C++ Neural Networks and Fuzzy Logic, Second Edition*. MIS Press, 1995.

[Rasmussen92] Rasmussen, E. *Clustering Algorithms*. Information Retrieval. Data Structures & Algorithms. Pp. 419 a 442. 1992. In Frakes, W.; Baeza-Yates, R. (Eds.) Prentice Hall.

[Rich94] Rich, E.; Knight, K. *Inteligencia Artificial. Segunda Edición.* 1994. Ed. McGraw-Hill. ISBN: 84-481-1858-8.

[Riecken94] Riecken, D. *A Conversation with Marvin Minsky About Agents.* Communications of the ACM. July, 1994. Vol. 17, No. 7.

[Scharf96] Scharf, Dean. *HTML Referencia Visual. Segunda Edición.* Prentice Hall Hispanoamericana. Mexico, 1996.

[Singh95] Singh, N.; Tawakol, O.; Genesereth, M. *A Name-Space Context Graph for Multi-Context, Multi-Agent Systems.* Stanford University, Stanford, CA, USA. (1995). `http://cuiwww.unige.ch/OSG/people/jvitek/Resources/Archive/ontology.ps.gz`

[Sopena76] *SAPIENS. Enciclopedia Ilustrada de la Lengua Castellana.* Ed. Sopena Argentina. Buenos Aires, 1976.

[Sebastiani98] Sebastiani, M. *Recuperación, filtrado y clasificación de documentos, y sus aplicaciones en Internet.* 1998. Escuela de Ciencias Informáticas. ECI'98. Departamento de Computación. Facultad de Ciencias Exactas y Naturales. Universidad de Buenos Aires.

[Skapura96] Skapura, David. *Building Neural Networks.* ACM Press, Addison-Wesley. New York, 1996.

[vanDam97] van Dam, A. *Post-WIMP User Interfaces.* Communications of the ACM. February, 1997. Vol. 40, No. 2.

[Wang97] Wang Baldonado, Michelle; Winograd, Terry. *SenseMaker: An Information-Exploration Interface Supporting the Contextual Evolution of a User's Interests.* In Proceedings of the Conference on Human Factors in Computing Systems, pp. 11-18. ACM Press, New York, Atlanta, Ga. March, 1997. `http://www-diglib.stanford.edu/cgi-bin/WP/get/SIDL-WP-1996-0048`

[Wasserman89] Wasserman, P. *Neural Computing. Theory and Practice.* 1989. Ed. Anza Research.

[Wayar99] Wayar, Luis Tomás. *LINUX, Manual de Referencia.* Ed. PC Forum S.A. Buenos Aires, 1999.

[Weber97] Weber, J. *Special Edition Using Java 1.1 Third Edition.* 1997. Ed. Que Corporation. ISBN: 0-7897-1094-3.

[Weiss92] Weiss, M. *Data Structures and Algorithm Analysis.* The Benjamin/Cummings Publishing Company, Inc. 1992. Redwood City, California, USA.

[Wiederhold85] Wiederhold, Gio. *Diseño de Bases de Datos. Segunda Edición.* Editorial Mc Graw-Hill, 1985.

[Wiederhold97] Wiederhold, Gio. *Mediators in the Architecture of Future Information Systems.* In Readings in Agents. Págs. 185–196. 1997. M. Hunhs & M. Singh, eds. Morgan Kauffman.

[Winston94] Winston, P. *Inteligencia Artificial. Tercera Edición.* 1994. Ed. Addison-Wesley Iberoamericana. ISBN: 0-201-51876-7.

[WinHelp98] Microsotf Windows 98 Help.

[Yokoi95] Yokoi, T. *The EDR Electronic Dictionary.* Communications of the ACM. November, 1995. Vol. 38, No. 11. ACM 0002-0782/95/1100.

[Zeller97] Zeller, M.; Sharma, R.; Schulten, K. *Motion Planning of a Pneumatic Robot Using a Neural Network.* June 1997. IEEE Control Systems.