UNIVERSIDAD NACIONAL DE INGENIERIA

FACULTAD DE INGENIERIA ELÉCTRICA Y ELECTRÓNICA

PROTOTIPO DE BÚSQUEDA EN INTERNET BASADO EN ALGORITMOS GENÉTICOS

TESIS

Para optar el Grado de Maestro en Ciencias

Mención: Telemática

Presentada por:

Héctor Manuel Bedón Monzón

LIMA - PERU

2003

UNIVERSIDAD NACIONAL DE INGENIERIA

FACULTAD DE INGENIERIA ELÉCTRICA Y ELECTRÓNICA

Prototype of search in Internet based on Genetic Algorithms

THESIS

Requirement for the degree of Master in Science

Field: Electronics Engineering
Major: Telematics

Author:

Héctor Manuel Bedón Monzón

Lima-Perú

Abstract

The present amount of information available in Internet is huge and diverse, that is the reason why, nowadays, it is necessary to make a research about software tools that will allow to find relevant information in a customized form.
This thesis presents the design and implementation of a prototype of an intelligent agent for extraction of information over a Linux platform, based on the genetic algorithm's methods which optimizes the searching process about specific topics in a customized way based on a profile provided by the user. With these goals in mind it has been designed and implemented a metasearch engine that offers a querying interface to multiple websearching machines such as: Google, Altavista and Yahoo, and a text extractor that allows, by the use of a model of vectorial space for word processing, to extract relevant text of documents in HTML to further evaluate its similarity with the profile provided by the user.
An increase of the similarity with the increase of the number of generations has been observed, not depending on the context search to being analyzed. This suggests an improvement in the information recovered with the increase of the number of generations. Under a scheme of refeeding by the user it has been possible to observe that the system adjusts better to the context of the user.

UNIVERSIDAD NACIONAL DE INGENIERIA

FACULTAD DE INGENIERIA ELÉCTRICA Y ELECTRÓNICA

Prototipo de Búsqueda en Internet basado en Algoritmos Genéticos

TESIS

Requerimiento para Optar el Grado de Maestro en Ciencias

Mención: Ingeniería Electrónica
Especialidad: Telemática

Autor:

Héctor Manuel Bedón Monzón

Lima-Perú

Resumen

La actual cantidad de información disponible en Internet es muy grande y diversa, esta es la razón por lo que se hace necesario hoy en día investigar sobre herramientas de software que permitan encontrar información relevante en forma personalizada.
Esta tesis presenta el diseño e implementación de un prototipo de agente inteligente para extracción de información sobre una plataforma Linux, basado en el método de algoritmos genéticos que optimiza el proceso de búsqueda sobre tópicos específicos de una manera personalizada basándose en un perfil que provee el usuario. Con el fin de lograr estos objetivos se ha diseñado e implementado un motor de metabusqueda que ofrece una interfase de consulta a múltiples motores de búsqueda tales como: Google, Altavista y Yahoo y un extractor de texto que permite, haciendo uso del modelo de espacio vectorial para procesamiento de texto, extraer texto relevante de documentos HTML para posteriormente evaluar su similitud con el perfil provisto por el usuario.
Un incremento de la similitud con el incremento del número de generaciones se ha observado, no dependiendo del contexto de búsqueda a ser analizado. Esto sugiere una mejora en la información recuperada con el incremento del número de generaciones. Bajo un esquema de realimentación por el usuario se ha podido observar que el sistema se ajusta mejor al contexto del usuario.

TABLA DE CONTENIDO

Pág.

Capítulo 1

Introducción

La actual cantidad de información presente en Internet y en su componente más visible, la World Wide Web, que posee cientos de millones de páginas de información[1] ha hecho que esta se convierta en una vasta fuente de información en una amplia variedad de tópicos. El problema es que hallar la información que una persona busca es usualmente bastante dificultosa y algunas veces una actividad que consume mucho tiempo, debido a la complejidad en la organización y la cantidad de información almacenada. La información es almacenada en archivos dentro de directorios en un sitio web identificado por una serie única de cuatro números o un nombre simbólico correspondiente. El numero de directorios y archivos son limitados únicamente por la capacidad de almacenaje y la disponibilidad del contenido. Hallar la pagina correcta que contenga la información que buscamos vía su URL (Uniform Resource Locator) seria imposible si tuviéramos que adivinar donde esta localizada la información. Afortunadamente motores de búsqueda como Google[2], Altavista[3], Yahoo[4] y muchos otros hacen que navegar en la web sea fácil. Estos servicios proveen acceso a los usuarios a grandes bases de datos que contienen palabras claves que referencian a URLs.

1.1 Internet

El Internet, algunas veces llamado simplemente "La Red", es un sistema mundial de redes de computadoras, un conjunto integrado por las diferentes redes de cada país del mundo, por medio del cual un usuario en cualquier computadora puede, en caso de contar con los permisos apropiados, acceder a información de otra computadora y poder tener inclusive comunicación directa con otros usuarios en otras computadoras.

1.1.1 Historia

Los años sesenta

La Agencia de Proyectos de Investigación Avanzada (**ARPA**) se inició en el Departamento de Defensa de los Estados Unidos en los últimos años de la década de los cincuenta para investigar los campos de ciencia y tecnología militar.

Paralelamente, entre 1962 y 1964 la **RAND** Corporation publicó artículos escritos por Paúl Baran sobre 'Redes de Comunicación Distribuidas". El objetivo de la propuesta era plantear una red que tuviera la máxima resistencia ante cualquier ataque enemigo. Se suponía que una red de comunicaciones, por si misma, no es fiable debido a que parte de ella podría ser destruida durante un ataque bélico.

Por lo tanto, cada nodo debería mantener la misma importancia que los demás para garantizar que no pudiera ser un punto critico que pudiera dejar la red inactiva o fuera de servicio. Baran promovió el uso de redes de conmutación de paquetes de datos (Packet Switching Networks) que

permitiesen que la información transmitida se dividiese en paquetes del mismo tamaño e importancia y se transmitieran a través de los nodos en los cuales se encontrara la ruta más eficiente para que al llegar a su destino se reagruparan en el orden que tenían previamente.

Los paquetes de información no necesitaban tener ninguna información sobre el ordenador de destino -salvo su dirección- ni sobre el medio de transmisión de la red. La utilidad fundamental de esta idea seria que cada paquete de información encontraría su propio camino independientemente de otros paquetes que constituirían parte del mismo mensaje. Al llegar al punto de destino todos los pequeños paquetes de información serían reagrupados en el orden correcto, el orden en que se encontraban antes de ser separados.

En 1968 el Laboratorio Físico Nacional en Inglaterra estableció la primera red de prueba basada en estos principios. En el mismo año, el primer diseño basado en estos principios de envío de paquetes de información, realizado por Lawrence G. Roberts, fue presentado en la ARPA. La red se llamó **ARPANET**.

Al año siguiente, el Departamento de Defensa dio el visto bueno para comenzar la investigación en **ARPANET**. El primer nodo de **ARPANET** fue la Universidad de California en Los Angeles. Pronto le siguieron otros tres nodos: la Universidad de California en Santa Bárbara, el Instituto de Investigación de Stanford y la Universidad de Utah. Estos sitios (como denominamos a los nodos) constituyeron la red original de cuatro nodos de **ARPANET**. Los cuatro sitios podían transferir datos en ellos en líneas de alta velocidad para compartir recursos informáticos.

En 1969 apareció el primer RFC (Request For Comment). Los RFC's, documentos emitidos periódicamente, se han convertido en su conjunto en las normas y estándares de Internet. Literalmente, "una solicitud para comentario", en su origen eran preguntas formuladas por estudiantes que no sabían qué acción tomar ante la falta de normativas. Es la respuesta a dicha pregunta o la iniciativa de tomar un camino particular ante la falta de orientación lo que convierte la RFC en norma.

Los años setenta

El comienzo de la década de los setenta vio el crecimiento de la popularidad del correo electrónico sobre redes de almacenamiento y envío. En 1971, **ARPANET** había crecido hasta 15 nodos con 23 ordenadores hosts (centrales). En este momento, los hosts de **ARPANET** comienzan a utilizar un protocolo de control de redes, pero todavía falta una estandarización. Además, había muy diferentes tipos de hosts, por lo que el progreso en desarrollar los diferentes tipos de interfaces era muy lento.

En 1972 Larry Roberts de DARPA decidió que el proyecto necesitaba un empujón. Organizó la presentación de **ARPANET** en la Conferencia Internacional sobre Comunicaciones por Ordenador. A partir de esta conferencia, se formó un grupo de trabajo internacional para investigar sobre los protocolos de comunicación que permitirían a ordenadores conectados a la red, comunicarse de una manera transparente a través de la transmisión de paquetes de información.

También en 1972 Bolt, Beranek y Newman (BBN) produjeron una aplicación de correo electrónico que funcionaba en redes distribuidas como **ARPANET**. El programa fue un gran éxito que permitió a los investigadores coordinarse y colaborar en sus proyectos de investigación y desarrollar las comunicaciones personales. Las primeras conexiones internacionales se establecieron en la Universidad College London, en Inglaterra y en el Royal Radar Establishment, en Noruega, junto con los ahora 37 nodos en EE.UU. La expansión en **ARPANET** era muy fácil debido a su estructura descentralizada.

En 1974 se estableció el Transmission Control Protocol (TCP), creado por Vinton Cerf y Bob Kahn que luego fue desarrollado hasta convertirse en el Transmission Control Protocol/Internet Protocol (TCP/IP). TCP convierte los mensajes en pequeños paquetes de información que viajan por la red de forma separada hasta llegar a su destino donde vuelven a reagruparse. IP maneja el direccionamiento de los envíos de datos, asegurando que los paquetes de información separados se encaminan por vías separadas a través de diversos nódulos, e incluso a través de múltiples redes con arquitecturas distintas.

En julio de 1975 **ARPANET** fue transferido por DARPA a la Agencia de Comunicaciones de Defensa.

El crecimiento de **ARPANET** hizo necesario algunos órganos de gestión: el Internet Configuration Control Board fue formado por ARPA en 1979. Más tarde se transformó en el Internet Activities Board y en la actualidad es el Internet Architecture Board of the Internet Society.

Los años ochenta

ARPANET en sí mismo permaneció estrechamente controlado por el DoD hasta 1983 cuando su parte estrictamente militar se segmentó convirtiéndose en MILNET.

La "European Unix Network" (EuNet), conectado a **ARPANET**, se creó en 1982 para proporcionar servicios de correo electrónico y servicios Usenet a diversas organizaciones usuarias en los Países Bajos, Dinamarca, Suecia e Inglaterra.

En 1984 el número de servidores conectados a la red había ya superado los 1,000.

Dado que el software de TCP/IP era de dominio público y la tecnología básica de Internet (como ya se denominaba esta red internacional extendida) era algo anárquica debido a su naturaleza, era difícil evitar que cualquier persona en disposición del necesario hardware (normalmente en universidades o grandes empresas tecnológicas) se conectase a la red desde múltiples sitios.

En 1986, la National Science Foundation (NSF) de EE.UU. inició el desarrollo de NSFNET que se diseñó originalmente para conectar cinco superordenadores. Su interconexión con Internet requería unas líneas de muy alta velocidad. Esto aceleró el desarrollo tecnológico de INTERNET y brindó a los usuarios mejores infraestructuras de telecomunicaciones. Otras agencias de la Administración norteamericana entraron en Internet, con sus inmensos recursos informáticos y de comunicaciones: NASA y el Departamento de Energía.

Un acontecimiento muy importante era que los proveedores comerciales de telecomunicaciones en EE.UU. y Europa empezaron a ofrecer servicios comerciales de transporte de señales y acceso.

En 1 987 el número de servidores conectados a Internet superaba ya los 10,000.

El día 1 de noviembre de 1988 Internet fue "infectada" con un virus de tipo "gusano". Hasta el 10% de todos los servidores conectados fueron afectados. El acontecimiento subrayó la falta de adecuados mecanismos de seguridad en Internet, por lo cual DARPA formó el Computer Emergency Reponse Team (CERT), un equipo de reacción rápida que mantiene datos sobre todas las incidencias en red y sobre las principales amenazas.

En 1989 el número de servidores conectados a Internet alcanza ya los 100.000. En este mismo año, se inauguró también la primera conexión de un sistema de correo electrónico comercial a Internet (MCI y Compuserve). Una nueva época estaba a punto de empezar, la de la explotación comercial de Internet.

Los años noventa

ARPANET como entidad se extinguió en 1989/90, habiendo sobrepasado con mucho los objetivos y metas que tenía en su origen. Los usuarios de la red apenas lo notaron, ya que las funciones de **ARPANET** no solamente continuaron, sino que mejoraron notablemente a través de nuevos órganos más representativos de la utilización actual de la red.

En 1990 redes de diversos países como España, Argentina, Austria, Brasil, Chile, Perú, Irlanda, Suiza y Corea del Sur se conectaron también a NSFNET.

En 1991 se retiraron las restricciones de NFS al uso comercial de INTERNET, desde el cual empezó un gran crecimiento en el área comercial hasta el día de hoy (Fig. 1.1). Ese mismo año también se Conectaron más países a la NSFNET incluyendo: Croacia, Hong Kong, República Checa, Sudáfrica, Singapur, Hungría, Polonia, Portugal, Taiwán y Túnez.

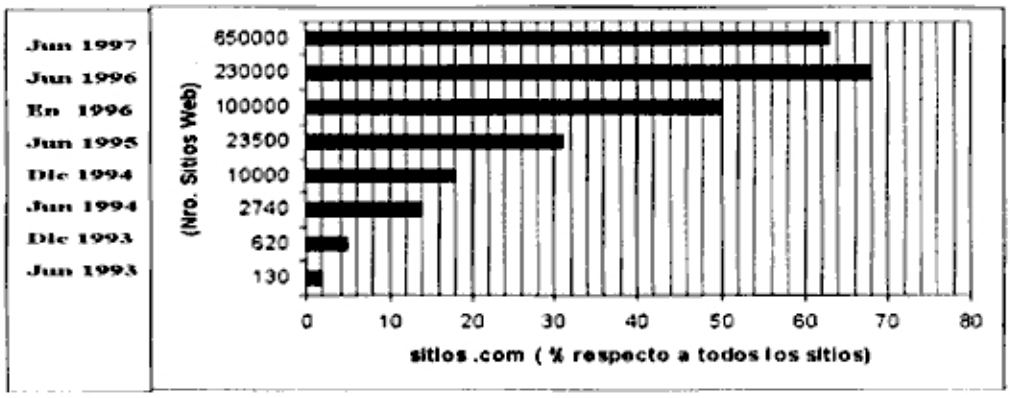

Fig. 1.1: Crecimiento de los sitios .com[5].

En 1992 el número de servidores conectados a INTERNET sobrepasaba la cifra de un millón de servidores. En ese año, la **Sociedad de INTERNET (ISOC)** se formó para promocionar el intercambio global de información. **La Internet Architecture Board (IAB)** fue reorganizada para llegar a formar parte del ISOC.

Como acontecimiento clave en la historia reciente de Internet, también en 1992 se desarrolló la World Wide Web en el Laboratorio Europeo de Física en Suiza. Esta tecnología provocó un drástico cambio en la apariencia, en el sentido y en el uso de INTERNET.

En 1993 el número de servidores INTERNET sobrepasa los 2,000,000. También NSF patrocina la formación de una nueva organización, InterNIC, creada para proporcionar servicios de registro en Internet y bases de datos de direcciones. El conocido navegador WWW 'Mosaic" se desarrolló en el National Center for Supercomputing.

El número de servidores de Internet alcanza los **3.800.000** en 1994. Las primeras tiendas Internet empiezan a aparecer junto con "emisores" de radio on-line. El conflicto potencial entre los internautas tradicionales y los nuevos usuarios se manifestó con el tumulto que causó un gabinete legal americano que introdujo publicidad en Internet.

En 1995 había más de 5 millones de servidores conectados a Internet. La espina dorsal de NSFNET empezaba a ser sustituido por proveedores comerciales interconectados.

Hoy en día Internet está formada, no solamente de restos de la **ARPANET** original, sino que también incluye redes como la Academia Australiana de Investigación de redes (AARNET), la NASA Science Internet (NSI), la Red Académica de Investigación Suiza (SWITCH), por no mencionar las miles de redes de mayor o menor tamaño de tipo educativo y de investigación.

La velocidad de crecimiento de Internet en la década de los noventa ha sido muy rápida (Fig. 1.2). Se extiende casi a la misma velocidad que los ordenadores personales en los años ochenta.

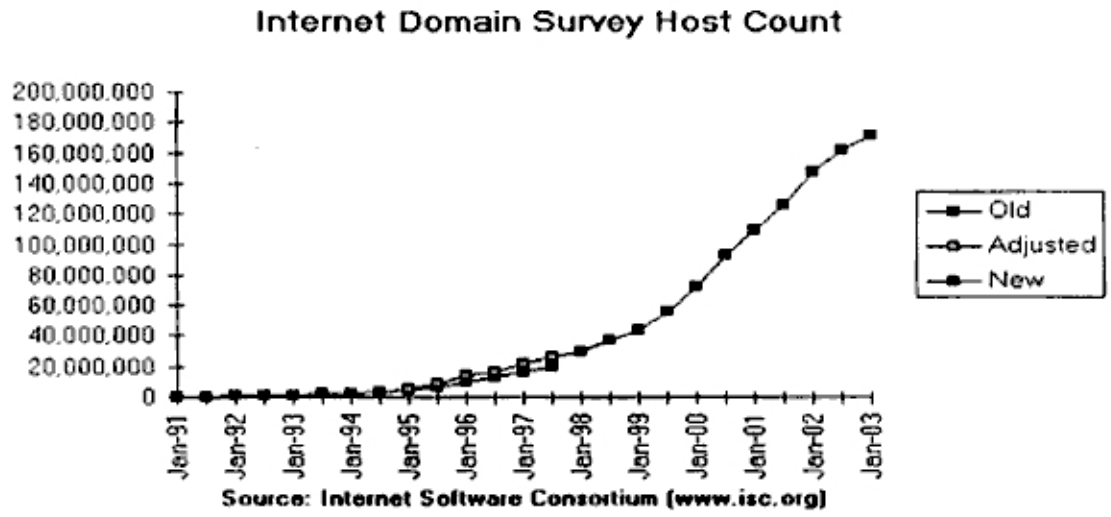

Fig. 1.2: Crecimiento de computadoras conectadas a Internet[5].

La Administración norteamericana sigue apoyando en gran medida a la comunidad de Internet, debido, sin duda, a que ésta era en su origen un programa de investigación respaldado federalmente, y ha llegado a ser una parte importante de la infraestructura de investigación académica e industrial estadounidense.

En nuestro país también ha habido un crecimiento de computadoras conectadas a Internet pero no a la velocidad que han crecido nuestros vecinos latinoamericanos, tal como se observan en las figuras 1.3 y 1.4.

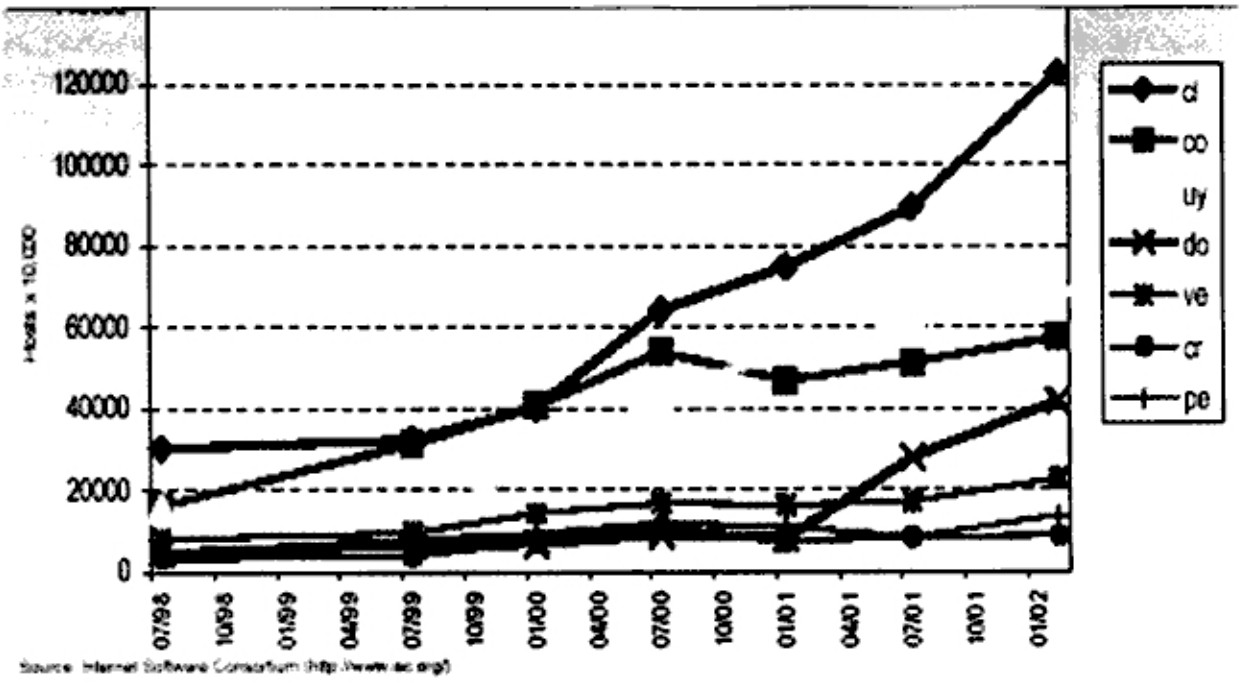

Fig. 1.3: Crecimiento de computadoras conectadas a Internet en Latinoamérica[6].

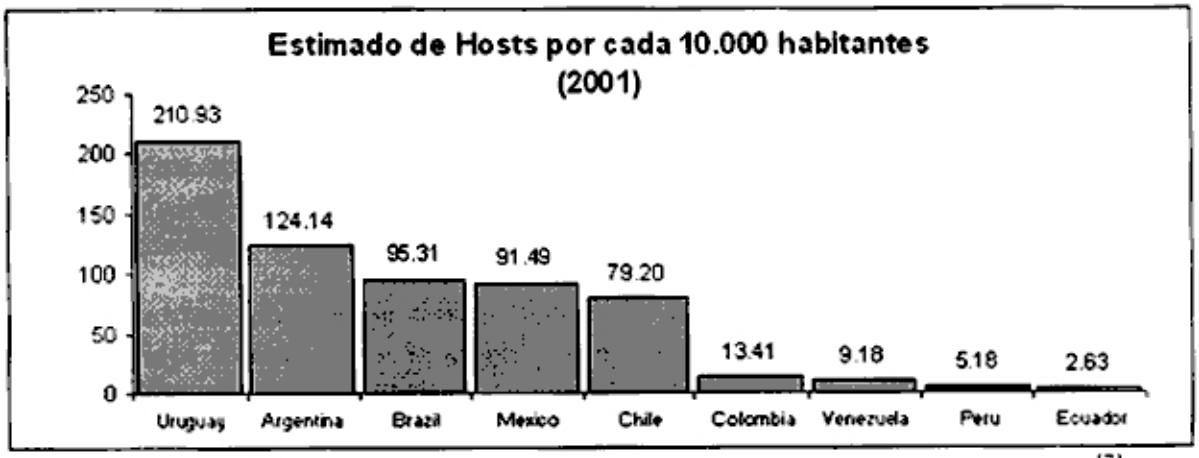

Fig. 1.4: Estimado de computadoras conectadas a Internet en Latinoamérica[7].

1.1.2 Problemas actuales

El planteamiento inicial de Internet en 1973 y 1974 contempló un total de 256 redes interconectadas. No se contemplaba la posibilidad de que participaran más de estas 256 redes. Cuando las redes locales aparecieron en grandes números, se inventó una manera de subdividir las direcciones TCP/IP con el fin de mantener el diseño original. Luego se crearon las direcciones de clases A, B y C para permitir la conexión de hasta millones de servidores.

Aún así, las direcciones se están agotando. La solución al problema debería permitir una implantación incremental. Es decir, se debería empezar a utilizar en paralelo con las direcciones actuales, pero sin impactar en la funcionalidad de las mismas. Existen propuestas de diferentes índoles para tratar el problema. Una de ellas y que ya se está implantando es la versión actualizada de TCP/IP. La transición de la versión 4 (actual) de IP a la siguiente generación (versión 6) ya se esta dando actualmente. Algunas de las ventajas de la nueva generación de IP incluyen:

- Aumento de direcciones de 32 bits a 128 bits.
- Cabeceras de mensajes simplificadas.
- Cabeceras extendidas opcionales que permiten mayor control de seguridad.

Las personas claves en esta transición son los vendedores de equipos de direccionamiento (routers) y software de sistemas host, y los proveedores de servicios Internet. Sobre todo estos últimos, tienen un interés en asegurar que el mayor número de usuarios puedan acceder a sus servicios, sin perder conectividad en el proceso de transición.

Además de estos problemas relativos al crecimiento de la Internet, grandes cambios han estado ocurriendo en el área de la oferta y la demanda de información. Algunas de estas características son:

a) Se eleva la cantidad de datos disponibles por medios electrónicos.
b) Puede ser obtenida fácilmente
c) La cantidad de proveedores ha crecido tanto que no se tiene un cuadro claro de todos ellos.

La cantidad tan grande de información disponible en la Web nos lleva a pensar erróneamente que todas nuestras necesidades de información han sido cubiertas, el problema es que esta principal fortaleza es su debilidad, ya que la información que verdaderamente nos interesa esta oculta.
Los métodos de extracción convencionales no parecen ser capaces de resolver estos problemas. Estos métodos se basan en principio en el conocimiento de **qué** información esta disponible y **dónde** puede ser exactamente hallada.

En la Internet esta estrategia falla completamente debido a:

a) **La naturaleza dinámica de la propia Internet:** No existe una supervisión central del crecimiento y desarrollo de Internet.
b) **La naturaleza dinámica de la información en la Internet:** Información que no puede ser hallada hoy puede estar disponible mañana.

c) **La información y los servicios de información en Internet son muy heterogéneos:** La información ofrecida en Internet esta siendo ofrecida en distintos tipos de formato y de muchos modos diferentes.

1.1.3 Administración Actual

No existe una autoridad definida a nivel global para Internet como ya habíamos mencionado anteriormente. Cada una de las redes sí tiene su administración, pero no el conjunto. Es una cierta acracia autoordenada.

- Existe un organismo de pertenencia voluntaria, el IAB (Internet Architecture Board o Consejo sobre Arquitectura de la Internet) que promociona el intercambio de información técnica y asigna determinados recursos, como las direcciones.
- El IAB decide qué estándares hacen falta y promueve su uso a través de la propia Internet.
- Los usuarios de Internet expresan sus opiniones sobre cuestiones técnicas a través de reuniones del IETF (Internet Engineering Task Force o Grupo de Trabajo sobre Ingeniería de Internet).
- El IETF crea comités de estudio para diversos problemas técnicos, como pueden ser:
    - Generación de documentación.
    - Forma de conectarse a algún servicio.
- Regla básica: Quien quiera beneficiarse del uso y los recursos de la Red debe observar sus normas, no escritas.

1.2 Propósito y Motivación

En este contexto de veloz incremento de Internet y por lo tanto de la información disponible en ella, hacen su aparición como hemos visto los motores de búsqueda para facilitar la labor de búsqueda de información. Los motores de búsqueda típicamente responden a la consulta del usuario recuperando URLs de sus propias bases de datos. Ellos proporcionan información desde los de tipo general como Google[2] hasta los más específicos como Webseek[8]. Tradicionalmente los motores de búsqueda retornan los resultados de la consulta visualizando una larga lista de documentos sin algún proceso de clasificación de datos o agrupación. También las características de búsqueda así como el tamaño de las bases de datos cambian y ningún motor de búsqueda provee el mejor índice para todas las materias, por lo que los usuarios tienen que aprender diferentes características e interfaces para cada motor de búsqueda. Si bien es cierto estas características de los motores de búsqueda inicialmente han sido muy útiles hoy en día dada la gran cantidad de información presente en la Internet se hace necesario la creación de nuevos sistemas de búsqueda que agrupen a los ya existentes y se ajusten a un perfil definido por el usuario.

Los motores de metabusqueda ofrecen una solución parcial a este problema. En vez de mantener una base de datos local ellos usan los índices de otros motores de búsqueda y dan a los usuarios una interfase simple hacia y desde los resultados de la búsqueda. El motor de metabusqueda es una herramienta de búsqueda de gran valor pero todavía deja algunos problemas sin resolver

como el hecho de que están sujetos a los datos que le provean los motores de búsqueda a los cuales ellos accedan que muchas veces no son del interés del usuario.

Así en años recientes agentes de software inteligente han emergido de la investigación en el área de la inteligencia artificial[34] como herramientas para solucionar estos problemas lo cual constituye en una motivación para el siguiente trabajo de tesis así como construir un sistema pionero en su genero en nuestro país.

1.3 Objetivos de la Tesis

Esta tesis trata de buscar una nueva herramienta que permita *optimizar la gran cantidad de información presente en la web sobre tópicos específicos de una manera personalizada*, es decir de acuerdo a un perfil de interés provisto por el usuario.

Con tal fin se ha construido en primera instancia un agente de metabúsqueda que ofrece una interfase común para colocar una consulta consecutivamente a tres de los buscadores accesibles más conocidos como Google[2], Altavista [3] y Yahoo[4].

También se ha construido un extractor de información de texto que consiste de un analizador léxico, un algoritmo de supresión de palabras insignificantes, el cual analiza las páginas HTML.

Por último un agente de aprendizaje basado en algoritmos genéticos colecciona, evalúa y presenta al usuario las páginas HTML de la web que más se ajustan al perfil de interés del usuario.

1.4 Organización de la Tesis

En el capitulo 2 se exponen las características, estructura y requerimientos de los motores de búsqueda y de metabúsqueda. En el capitulo 3 se presentan las características que presenta los agentes inteligentes que incorporan técnicas de inteligencia artificial en el proceso de búsqueda de información. Una técnica que viene del área de inteligencia artificial como son los Algoritmos Genéticos, muy usada en los procesos de optimización y que se implementa en esta tesis, es descrita en el capitulo 4. Conceptos relacionados al área de Extracción de Información (Information Retrieval) usados para ponderar las páginas HTML se consideran en el capitulo 5. El proceso de implementación y la arquitectura de todos los elementos que componen el prototipo se describen en el capitulo 6. las conclusiones a las que arribamos se presentan en el capitulo 7, además se sugieren algunos puntos que podrían profundizarse para optimizar el prototipo.

Capítulo 2

Arquitectura de Motores de Búsqueda y Metabúsqueda

2.1 Motores de Búsqueda

Hay muy poca información acerca de los motores de búsqueda debido a la naturaleza comercial que hoy en día tienen, si bien es cierto que empezaron como proyectos académicos, tal es el caso por ejemplo de los buscadores Google[2], WebCrawler[9], Altavista[3].

Un motor de búsqueda es esencialmente un sistema de recuperación de información para páginas Web[10].
Existen dos tipos de motores de búsqueda:

i) Motores de búsqueda, tal como Altavista[3], crean sus listados automáticamente
ii) Directorios, tal como Yahoo[4], depende de humanos para su listado.

Algunos motores de búsqueda conocidos como híbridos mantienen un directorio asociado.
Los motores de búsqueda consisten tradicionalmente de tres componentes el crawler, el software de indexación y el software de búsqueda y clasificación. Hay diferencias en la forma que los motores de búsqueda trabajan, pero todos ejecutan tres tareas básicas:

- Exploran la Web o seleccionan partes de ella basados en palabras importantes
- Mantienen un índice de las palabras que se hallan y donde se hallan.
- Permiten visualizar a los usuarios palabras o combinaciones de palabras que mantienen en su índice.

Los primeros motores de búsqueda poseían un índice de unos pocos cientos de miles de páginas y documentos y respondían a decenas de millones de consultas por día. Hoy en día un motor de búsqueda muy popular indexa cientos de millones de páginas y responde a decenas de millones de consultas por día. Al momento de preparación de esta tesis Google[2] mantiene un índice de 3,083,324,652 de páginas web.

Antes de que la Web (World Wide Web) se convierta en la parte más visible de la Internet, había motores de búsqueda que permitían hallar información sobre la red. Programas como "Gopher" y "Archie" mantenían índices de archivos almacenados sobre servidores conectados a la Internet y dramáticamente redujeron la cantidad de tiempo requerido para hallar programas y documentos.
Numerosas tecnologías de búsqueda han sido aplicadas a los motores de búsqueda y todavía no ha sido identificado un método dominante. Estas tecnologías de búsqueda en la web pueden ser clasificadas en seis categorías[11]: Exploración de Hiperenlaces, Recuperación de información, Metabuscadores, Aproximaciones SQL, Búsqueda multimedia basada en contenidos y Otros (Buscadores basados en Inteligencia Artificial, personalizados, específicos sobre un sitio Web, etc.).

2.1.1 Arquitectura de los Motores de Búsqueda

Los motores de búsqueda consisten tradicionalmente de tres componentes el crawler, el software de indexación y el software de búsqueda y clasificación (Fig. 2.1).

Crawler

Es un programa que automáticamente examina varios sitios Web y recoge documentos Web. Los crawlers siguen los enlaces de un sitio Web para hallar otras páginas relevantes. Dos algoritmos de búsqueda, búsqueda a lo ancho (bredth-first) y hacia abajo (depth-first) son ampliamente usados por los crawlers para surcar la Web. Los crawler ven a la Web como un diagrama, con los objetos localizados en los URLs (Uniform resource Locators) como nodos. Los objetos podrían ser HTTPs (Hypertext Transfer Protocols), FTPs (File Transfer Protocols), mailito(e-mail), noticias, etc. Estos crawlers retornan periódicamente a los sitios Web para poder mantener actualizada su base de datos.
Los puntos de partida usuales para el proceso de "web crawling" son los servidores más usados y las páginas más populares.

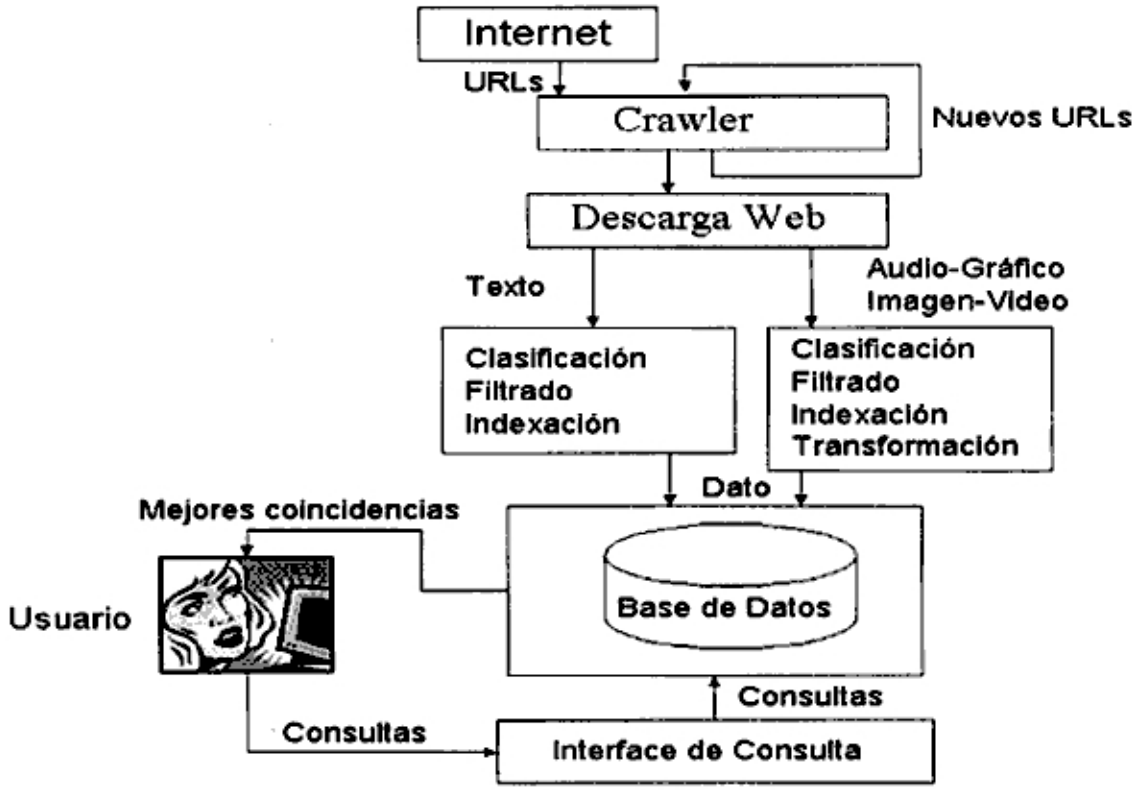

Fig. 2.1: Arquitectura de un Motor de Búsqueda

Software de Indexación

La indexación automática es el proceso algorítmico de examinar los elementos de información para construir una estructura de datos que pueda ser rápidamente rastreada.

Existen dos características muy importantes de las páginas web que los motores de búsqueda usan para el proceso de extracción de información de la web. La primera es el hecho de que las páginas HTML y en el futuro las páginas XML, están fuertemente etiquetadas (tagged). Un documento XML no únicamente provee la misma información que una página HTML como son las palabras claves, hiperenlaces, descripciones, etc., sino también nos da una información estructural de la página. Esta información estructural es la característica más crucial de un documento XML que no es proveída por un documento HTML.

Estas etiquetas llevan una rica información con respecto a las palabras claves usados en la página. Por ejemplo una palabra clave que aparezca en el título de un documento o que pueda estar resaltada nos puede indicar mucho acerca del contenido del documento. La segunda característica es que las páginas web están bastante enlazadas. Un enlace de una pagina X a una pagina Y provee un medio conveniente para que el usuario Web navegue desde la pagina X a la Y. Un enlace indica una buena probabilidad de que los contenidos de la pagina X y la Y están relacionados. La información del enlace ha sido usada para calcular la importancia global, es decir la jerarquía de la pagina web (*PageRank*) basada en si la pagina esta señalada por muchas páginas y/o páginas importantes [12].

El primer paso es indexar las palabras de sus páginas y luego seguir cada uno de los enlaces hallados dentro del sitio web. De esta manera el sistema rapidamente empezara a viajar propagándose a través de las mas ampliamente porciones usadas de la Web. Analizaremos el caso de Google[2] que es el motor de búsqueda que ha mostrado su estructura públicamente, además debemos saber también que esta estructura puede haber cambiado.
Google[2] empezó como un motor de búsqueda académico. Ellos construyen su sistema inicial para usar múltiples crawlers, usualmente tres al mismo tiempo. Cada crawler podría mantener alrededor de 300 conexiones a páginas web al mismo tiempo. En su pico de performance usando cuatro crawlers su sistema podría indexar arriba de las 100 páginas por segundo generando alrededor de 600 Kb de datos cada segundo.

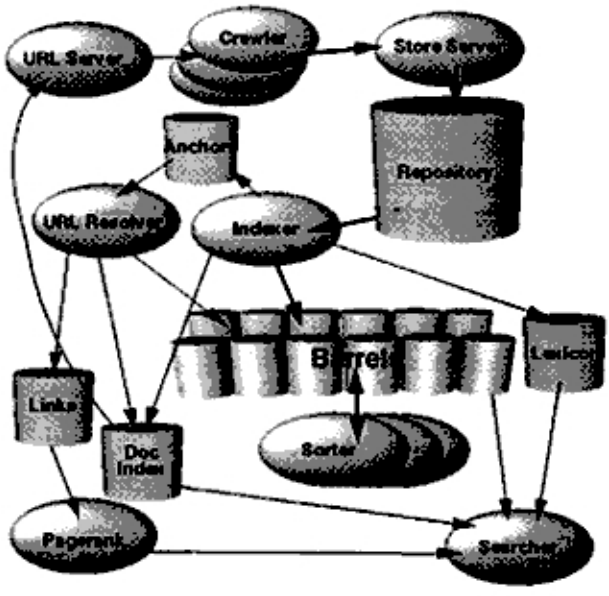

Fig. 2.2: Arquitectura de Google[12].

Mantener todo el sistema corriendo rápidamente significa alimentar con información necesaria a los crawlers por lo que Google tenía un servidor dedicado a proveer URLs a estos. Mas que depender de un proveedor de servicios de Internet para el servidor de dominios (DNS), el cual traslada el nombre de un servidor en una dirección, Google tuvo su propio DNS para poder mantener los retrasos al mínimo (Fig. 2.2).

Cuando el crawler de Google encuentra una página html toma nota de dos cosas:

- Las palabras dentro de la página
- Donde las palabras son halladas

Las palabras que ocurren en el titulo, subtitulo, metatags y otras posiciones de relativa importancia tienen una consideración especial durante una

Existen dos características muy importantes de las páginas web que los motores de búsqueda usan para el proceso de extracción de información de la web. La primera es el hecho de que las páginas HTML y en el futuro las páginas XML, están fuertemente etiquetadas (tagged). Un documento XML no únicamente provee la misma información que una página HTML como son las palabras claves, hiperenlaces, descripciones, etc., sino también nos da una información estructural de la página. Esta información estructural es la característica más crucial de un documento XML que no es proveída por un documento HTML.

Estas etiquetas llevan una rica información con respecto a las palabras claves usados en la página. Por ejemplo una palabra clave que aparezca en el título de un documento o que pueda estar resaltada nos puede indicar mucho acerca del contenido del documento. La segunda característica es que las páginas web están bastante enlazadas. Un enlace de una pagina X a una pagina Y provee un medio conveniente para que el usuario Web navegue desde la pagina X a la Y. Un enlace indica una buena probabilidad de que los contenidos de la pagina X y la Y están relacionados. La información del enlace ha sido usada para calcular la importancia global, es decir la jerarquía de la pagina web (*PageRank*) basada en si la pagina esta señalada por muchas páginas y/o páginas importantes [12].

El primer paso es indexar las palabras de sus páginas y luego seguir cada uno de los enlaces hallados dentro del sitio web. De esta manera el sistema rapidamente empezara a viajar propagándose a través de las mas ampliamente porciones usadas de la Web. Analizaremos el caso de Google[2] que es el motor de búsqueda que ha mostrado su estructura públicamente, además debemos saber también que esta estructura puede haber cambiado.
Google[2] empezó como un motor de búsqueda académico. Ellos construyen su sistema inicial para usar múltiples crawlers, usualmente tres al mismo tiempo. Cada crawler podría mantener alrededor de 300 conexiones a páginas web al mismo tiempo. En su pico de performance usando cuatro crawlers su sistema podría indexar arriba de las 100 páginas por segundo generando alrededor de 600 Kb de datos cada segundo.

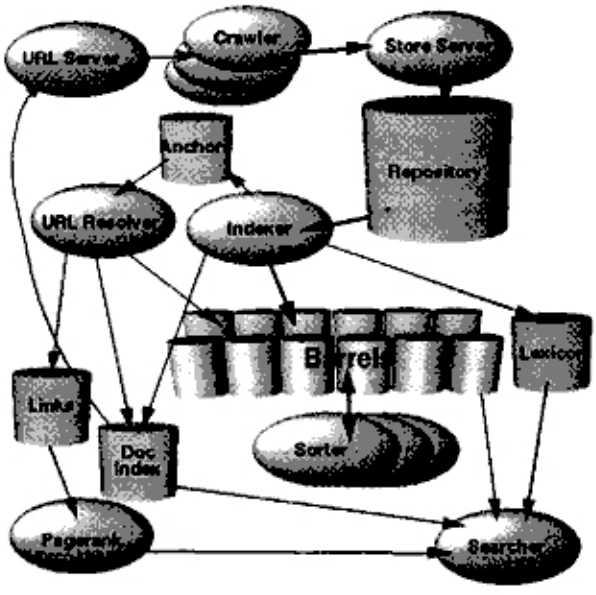

Fig. 2.2: Arquitectura de Google[12].

Mantener todo el sistema corriendo rápidamente significa alimentar con información necesaria a los crawlers por lo que Google tenía un servidor dedicado a proveer URLs a estos. Mas que depender de un proveedor de servicios de Internet para el servidor de dominios (DNS), el cual traslada el nombre de un servidor en una dirección, Google tuvo su propio DNS para poder mantener los retrasos al mínimo (Fig. 2.2).

Cuando el crawler de Google encuentra una página html toma nota de dos cosas:

- Las palabras dentro de la página
- Donde las palabras son halladas

Las palabras que ocurren en el titulo, subtitulo, metatags y otras posiciones de relativa importancia tienen una consideración especial durante una

búsqueda subsiguiente del usuario. El crawler de Google fue construido para indexar cada palabra significativa de una pagina, dejando de lado los artículos "un", "y","los", "para". Otros crawlers toman aproximaciones diferentes.

Cada aproximación en el proceso de extracción busca la eficiencia en la búsqueda para el usuario así como mejorar la velocidad de extracción por el crawler.

En el caso de Lycos[13] este mantiene una huella de las palabras en el título, subtítulo y los enlaces, junto con las 100 palabras mas usadas en la página y cada palabra en las primeras veinte líneas de texto.

Otros sistemas tal como Altavista[3] indexan cada palabra simple sobre la página, incluyendo "un","una","los" y otras palabras insignificantes.

A medida que los crawlers van hallando la información en la web, los motores de búsqueda deben almacenar la información en una forma que sea útil. Aquí juegan un rol muy importante dos cosas, la primera es la información almacenada con el dato y segundo, el método usado para indexar la información.

En el caso más simple un motor de búsqueda podría solo almacenar la palabra y la URL donde fue hallada, este motor seria muy limitado ya que no habría forma de decir si la palabra fue usada en una forma trivial o importante, el numero de veces que aparece en la pagina o si la pagina contiene enlaces a otras paginas que contienen la palabra. Por lo tanto no habría forma de clasificar en forma jerárquica las páginas halladas.

Debido a esto muchos motores de búsqueda almacenan mucho más que la palabra y la URL. Un motor de búsqueda podría almacenar el numero de veces que la palabra aparece en la página. El motor de búsqueda podría asignar un peso a cada entrada, con valores más altos a palabras que se encuentran mas cerca de la parte superior del documento, en subtítulos, en enlaces, en los metatags o en el título de la página. Cada motor de búsqueda tiene una fórmula diferente para asignar el peso a las palabras en su índice. Esta es una de las razones por la que muchas veces al buscar una palabra en varios buscadores estos arrojan resultados distintos.

Además de las piezas adicionales de información almacenadas por los motores de búsqueda esta información es codificada para ahorrar espacio de almacenamiento. En el caso de Google [2] usa 2 bytes para almacenar información sobre los pesos es decir si la palabra esta en mayúsculas, el tamaño del texto, posición y otra información que ayude en la clasificación de la ocurrencia. Cada factor podría tomar de 2 a 3 bits dentro del grupo de 8 bits, como consecuencia de esto una gran cantidad de información podría ser almacenada en una forma muy compacta. Después que la información es comprimida esta lista para ser indexada.

Un índice tiene el propósito de permitir que la información sea hallada tan rápida como sea posible. Hay varias formas en la que un índice pueda ser construido, pero una de las más efectivas es construir una tabla "Hash". En este proceso una formula es aplicada para vincular un valor numérico a cada palabra. La formula esta diseñada para distribuir las entradas uniformemente a través de un predeterminado numero de divisiones. Esta distribución numérica

es diferente de la distribución de palabras en el alfabeto y esto es lo que hace importante y efectivo a la tabla "Hash".

La tabla "Hash" contiene el valor numérico junto con un puntero hacia los datos reales que pueden ordenarse de cualquier manera lo que permite ser almacenado eficazmente. La combinación de una indexación y un almacenamiento eficaz hacen posible obtener resultados rapidamente aun cuando los usuarios creen búsquedas complicadas.

Software de Búsqueda y Clasificación

El procesamiento de la consulta es la actividad de analizar una consulta y compararla a los índices que mantiene el motor de búsquedas para hallar artículos relevantes.
Para generar un índice un usuario debe enviar una consulta a través de un motor de búsqueda, esta consulta al menos debe ser de una palabra clave. Al construir consultas más complejas requerimos usar operadores boleanos que permiten refinar y extender los términos de la búsqueda. Los más usuales son:

Y : Todos los términos unidos por "y" deben aparecer en las páginas o documentos. Algunos motores de búsqueda sustituyen el operador "+" por la palabra Y.

O : Al menos uno de los términos unidor por "O" deben aparecer en las páginas y documentos.

NO : El termino o términos que siguen a "NO" no deben aparecer en las páginas y documentos. Algunos motores de búsqueda sustituyen el operador "-" por la palabra NO.

Marcas con Comillas: Las palabras entre comillas son tratadas como una frase y aquella frase debe ser hallada dentro del documento o archivo.

Para poder determinar en que orden mostrar las paginas web al usuario, los motores de búsqueda usan un algoritmo para clasificar los sitios web que contienen las palabras claves. Por ejemplo el motor de búsqueda podría contar el numero de veces que la palabra clave aparece en la página.

2.2 Metabuscadores

Un metabuscador es un sistema que provee un acceso unificado a múltiples motores de búsqueda lo que los hace más efectivos pero un tanto más lentos y presentan los resultados en un formato integrado. Un metabuscador no mantiene su propio índice sobre los documentos lo que lo hace estar a merced de las políticas de clasificación que tiene cada buscador al cual se ha realizado la consulta. El hecho de que cada motor de búsqueda tenga un formato particular tanto para el formulario de consulta como para la presentación de sus resultados hace que la tarea del metabuscador sea un tanto compleja. El protocolo STARTS[13] ha sido propuesto para estandarizar los procesos de recuperación y búsqueda en Internet. Los objetivos son escoger las mejores fuentes (motores de búsqueda) para evaluar una consulta, enviar la consulta a las fuentes seleccionadas y finalmente combinar los resultados de la consulta obtenidos desde diferentes fuentes. Este protocolo ha recibido poco reconocimiento ya que ninguno de los motores de búsqueda mas usados lo ha aplicado.

Los metabuscadores tienen hoy en día una gran importancia debido a las siguientes razones:

i. Mayor cobertura de la Web

El estudio de S. Lawrence[1] indica que el porcentaje de páginas Web indexadas por los motores de búsqueda ha ido decreciendo sostenidamente debido principalmente a que la Web ha ido creciendo mas rápidamente que la capacidad de indexación que tienen los motores de búsqueda. Por lo que usando múltiples motores de búsqueda a través de un metabuscador incrementaríamos mucho el rango de cobertura en la Web.

ii. Resuelve el problema de escalabilidad de búsqueda en la Web

Es casi imposible que un motor de búsqueda pueda indexar todos los documentos que existen en la Web y si fuera así surgiría luego el problema de la actualización de los datos almacenados que también es otra gran tarea compleja, por lo que si tuviéramos un metabuscador que se enlace a un número de buscadores especializados se podría aliviar el problema, es mas, este tipo de búsqueda tal como lo informo S. Lawrence[1] seria mas eficiente que si usáramos buscadores de propósito general. Una idea favorable en este sentido y que hace pensar que las tecnologías de búsqueda se orientaran hacia aquel punto es que los motores de búsqueda especializados no necesitan grandes recursos tanto en el ámbito de software como de hardware ya que son relativamente pequeños lo que hace aun más fácil su proceso de actualización.

iii. Facilitan el acceso a múltiples motores de búsqueda

La información requerida por un usuario frecuentemente es almacenada en las bases de datos de múltiples motores de búsqueda. Obtener dicha información en primer lugar requeriría conocer los diferentes motores de búsqueda, en segundo lugar se tendría que conocer como elaborar la consulta en una forma adecuada a cada motor de búsqueda, en tercer lugar se tendría que realizar las consultas a cada motor de búsqueda y luego para clasificarlas tendríamos que leer cada documento que se obtiene ya que cada buscador tiene su propio sistema de clasificar la información (ranking) por lo que no habría forma de comparar los primeros resultados de un motor de búsqueda contra los primeros de un segundo motor de búsqueda.

Un metabuscador nos ofrece en cambio una interfase única de consulta y los resultados son sometidos a una única política de clasificación.

iv. Mejoran la efectividad de la recuperación de información

En el proceso de recuperación de la información tipo texto, documentos de una misma colección suelen estar agrupados en grupos (clusters) tal que los documentos en un mismo grupo están más relacionados que los documentos en diferentes grupos. Cuando se evalúa una consulta, son los grupos relacionados a la consulta los que se identifican primero y las búsquedas se llevan a cabo sobre estos grupos. Se ha demostrado que este método mejora la efectividad de la recuperación de información[15]. Para los documentos en la Web las bases de datos de diferentes motores de búsqueda de propósito especial son grupos naturales. Como resultado de esto si una consulta es enviada a un metabuscador, la búsqueda podría estar

restringida a los motores de búsqueda especializados relacionados a la consulta lo que mejoraría la efectividad de un metabuscador frente a un motor de búsqueda de propósito general[16]. En estos sistemas luego de que la consulta del usuario ha sido enviada, el sistema primero trata de hallar alguna información adicional sobre la consulta del usuario y propone al usuario diferentes áreas del conocimiento, donde sus términos de consulta son hallados y también los diferentes motores de búsqueda especializados en las áreas del conocimiento, los cuales facilitan al usuario la exploración del tópico particular de búsqueda ya que tiene el poder de elegir el área de conocimiento[17].

Los motores de metabúsqueda confían en los resultados retornados por los motores de búsqueda estándar lo que distorsiona los resultados del metabuscador ya que ninguno de los motores de búsqueda estándar da resultados no distorsionados. El motor de metabúsqueda del instituto de investigación NEC [18] resuelve este problema bajando y analizando cada documento para luego mostrar la consulta en un contexto. Esto ayuda a los usuarios a determinar rápidamente si el documento es relevante sin tener que bajar cada pagina, además de que bajando las páginas localmente nos aseguramos de tener las más recientes versiones de cualquier página y eliminamos enlaces muertos y contenidos antiguos, esta es una de las cualidades del sistema que estoy desarrollando.

De estos resultados se puede decir que en futuro los motores de búsqueda y de metabúsqueda deberán incrementar el uso del contexto de la información del usuario para poder incrementar la competencia y la diversidad en la Web [19].

2.2.1 Arquitectura de un Metabuscador

La arquitectura de un metabuscador normalmente consiste de tres componentes (Fig. 2.3):

Despacho: Nos determina a que motor de búsqueda especifico hay que enviar la consulta. La selección de un motor de búsqueda dependerá de la relación que este guarde con la consulta.

Interface: Adapta el formato de consulta del usuario para que coincida con el formato de un motor de búsqueda particular, el cual varía de un motor de búsqueda a otro.

Visualización: Los resultados en bruto son procesados para ser mostrados al usuario. Desde que cada motor de búsqueda usa una forma diferente de presentar sus resultados, el proceso de extracción de los registros importantes se individualiza para motor de búsqueda. Finalmente las referencias duplicadas pueden ser removidas y compaginadas en una lista simple [20].

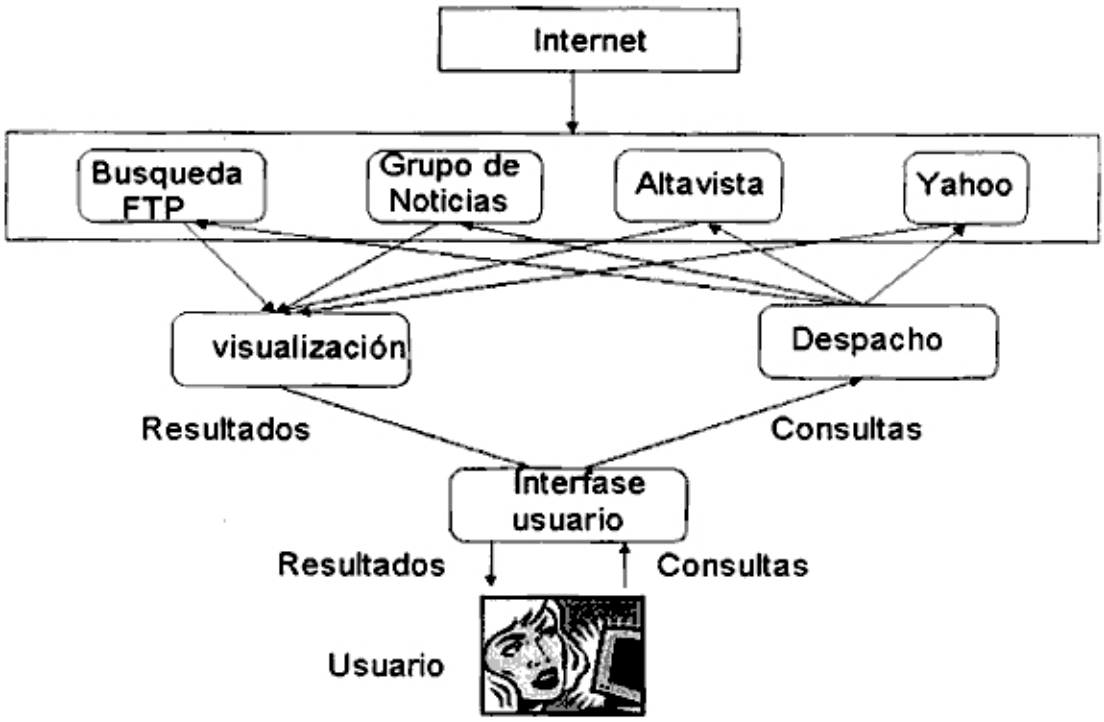

Fig. 2.3: Arquitectura de un Metabuscador.

2.2.2 Proceso de Transacción entre un Browser y un Motor de Búsqueda

Para poder entender como trabaja un metabuscador es necesario primero entender como es el proceso de transacción entre un browser y un motor de búsqueda, tal como explicamos a continuación.

La transacción entre el browser del usuario y el servidor Web de un motor de búsqueda esta basado en el protocolo de consulta / respuesta HTTP[33]. El cliente envía su consulta al servidor y espera la respuesta del servidor. Generalmente durante la comunicación cliente-servidor muchas consultas se refieren a archivos HTML estáticos que se encuentran en el servidor del disco duro. De cualquier modo en muchas aplicaciones Web así como en el caso de los motores de búsqueda, hay páginas Web dinámicas generadas en el proceso de navegación. El envío de una consulta al servidor se hace a través de un formulario HTML, usando uno de los dos métodos de envío, GET o POST.

La Fig. 2.4 muestra la interfase de un motor de búsqueda muy popular como Google[2]. El usuario escribe las palabras claves del tema de su interés, por ejemplo: *agentes de software inteligente*, en la caja de texto que aparece y luego presionando el botón *Búsqueda en Google* el browser envía la consulta al servidor y mas específicamente a la URL indicada por el parámetro ACTION dentro de la declaración <FORM>. El motor de búsqueda ofrece opciones al usuario como

búsqueda en: Web, Imágenes, Grupos y Directorios. Unas pocas líneas del código HTML usado por Google[2] para su formulario de entrada se muestra en la Fig. 2.5.

Fig. 2.4: Interfase de búsqueda del motor de búsqueda Google[2].

```
<FORM action=/search name=f><SPAN id=hf></SPAN>
<TABLE cellPadding=0 cellSpacing=0>
 <TBODY>
  <TD width=75> </TD>
  <TD align=middle><INPUT maxLength=256 name=q size=55>
   <SCRIPT>
document.f.q.focus();
</SCRIPT>
    <INPUT name=ie type=hidden value=ISO-8859-1><INPUT name=hl type=hidden
    value=es><BR><INPUT name=btnG type=submit value="Búsqueda en Google"><INPUT
name=btnI type=submit value="Voy a Tener Suerte"></TD>
  <TD noWrap vAlign=top><FONT size=-2> &#8226; <A
    href="http://www.google.com.pe/advanced_search?hl=es">Búsqueda
    Avanzada</A><BR> &#8226; 
<A  href="http://www.google.com.pe/language_tools?hl=es">Herramientas del
    idioma</A></FONT></TD></TR>
 <TR>
  <TD align=middle colSpan=3><FONT size=-1><INPUT CHECKED name=lr type=radio
    value="">Buscar en la Web<INPUT name=lr type=radio value=lang_es>Buscar
    sólo páginas en español</FONT></TD></TR></TBODY></TABLE></FORM>
```

Fig. 2.5. Parte del código fuente HTML que fue usado para generar el formulario de consulta de Google[2].

Para cada resultado usamos el término *registro web* para definir un conjunto de elementos que componen la información de un proceso de búsqueda como: una URL, una descripción del titulo y un sumario. La figura 2.6 muestra nuestro primer *registro web* definido como sigue:

Url	: http://black.rc.unesp.br/IA/cintiab/busca/cap3.html
Descripción	: Capítulo 3 - **Agentes** de **Software Inteligente** na Prática
Sumario	: CAPÍTULO 3. **Agentes** de **Software Inteligente** na Prática. ÍndiceGeral do Capítulo Três: 3.1, Aplicações de **Agentes** Inteligentes. ...

Fig. 2.6: Registro Web.

La figura 2.7 muestra la página de respuesta de Google a la consulta "agentes de software inteligente", tal como lo ve el usuario.

Fig. 2.7: Página de respuesta de Google[2] a la consulta "agentes de software inteligente".

La figura 2.8 ilustra únicamente el código referente a los primeros dos resultados visibles al usuario. Si una aplicación tuviera que extraer y usar únicamente la información que aparece en el *registro web*, ella debería ignorar todos los HTML TAGS y la información irrelevante (es decir anuncios publicitarios, iconos de ayuda, etc.) durante la etapa de análisis y mantener únicamente la información referente a los campos del *registro web*.

```
.....

<P class=g>
<A href="http://black.rc.unesp.br/IA/cintiab/busca/cap3.html">Capítulo 3 -
<B>Agentes</B> de <B>Software</B> <B>Inteligente</B> na Prática</A><BR><FONT
size=-1>CAPÍTULO 3. <B>Agentes</B> de <B>Software</B> <B>Inteligente</B> na
Prática. Índice Geral<BR>do Capítulo Três: 3.1, Aplicações de <B>Agentes</B>
Inteligentes. <B>...</B> <BR><FONT
color=#008000>black.rc.unesp.br/IA/cintiab/busca/cap3.html - 5k - </FONT>
<Aclass=fl
href="http://www.google.com.pe/search?q=cache:MkW0r5ECNSYC:black.rc.unesp.br/IA/cintiab/busca/cap
3.html+agentes+de+software+inteligente&amp;hl=es&amp;ie=UTF-8">En
caché</A> - <A class=fl
href="http://www.google.com.pe/search?hl=es&amp;lr=&amp;ie=UTF-
8&amp;q=related:black.rc.unesp.br/IA/cintiab/busca/cap3.html">Páginas
similares</A></FONT>
<BLOCKQUOTE class=g>

.....

 <P class=g>
<A  href="http://black.rc.unesp.br/IA/cintiab/busca/cap22.html">Teoria de
  <B>Agentes</B> de <B>Software</B> Inteligentes - Definição</A><BR><FONT
  size=-1><B>...</B> Na Lista de Correspondência sobre <B>Agentes</B> de
  <B>Software</B>, porém, uma possível<BR>definição informal de um agente de
  <B>software</B> <B>inteligente</B> foi determinada: <B>...</B> <BR><FONT
  color=#008000>black.rc.unesp.br/IA/cintiab/busca/cap22.html - 18k - </FONT>
<A  class=fl
href="http://www.google.com.pe/search?q=cache:3KQMsM2KxfoC:black.rc.unesp.br/IA/cintiab/busca/cap
22.html+agentes+de+software+inteligente&amp;hl=es&amp;ie=UTF-8">En
  caché</A> - <A class=fl
  href="http://www.google.com.pe/search?hl=es&amp;lr=&amp;ie=UTF-
8&amp;q=related:black.rc.unesp.br/IA/cintiab/busca/cap22.html">Páginas
  similares</A><BR>| <A class=fl
  href="http://www.google.com.pe/search?hl=es&amp;lr=&amp;ie=UTF-
8&amp;q=+site:black.rc.unesp.br+agentes+de+software+inteligente">Más
  resultados de black.rc.unesp.br</A> ]</FONT> </P></BLOCKQUOTE>
.....
```

Fig. 2.8: El código fuente HTML de respuesta de Google a la consulta "agentes de software inteligente".

Capítulo 3

Agentes de Software e Internet

Hay una considerable cantidad de definiciones del termino de Agentes, con diferentes puntos de vista, contexto y aplicaciones, voy a definir el termino agente en el contexto de esta tesis para lo cual en primer lugar voy a definir el termino *agente*, luego el termino *agente de software* y por último *agente de software inteligente (ASI)*.

3.1 Agente

Es aquel que está autorizado a actuar por otro.
Ejemplos de agentes humanos son los agentes de ventas, de viaje, políticos, etc.

3.2 Agente de Software

Un agente artificial que opera en un ambiente de software. El ambiente de software incluye sistemas operativos, aplicaciones computacionales, bases de datos, redes de computadoras y dominios virtuales.

3.2.1 Características de los Agentes de Software

Las características más generales que poseen los agentes de software se dan a continuación:

- **Autonomía**
  Los agentes operan sin intervención directa de los humanos. La experiencia mas el conocimiento integrado, les da una conducta definida por su propia experiencia, por lo que actúan satisfactoriamente en una amplia gama de entornos.
- **Reactividad**
  La reacción instantánea a ciertos cambios en el entorno, les da la posibilidad de reaccionar en entornos que requieren respuestas inmediatas.
- **Continuidad Temporal**
  Un agente esta corriendo procesos continuamente en el tiempo (activa en el primer plano o dormida en el background) lo que lo diferencia de otro tipo de programas.
- **Comunicación**
  Un agente puede interaccionar con otros agentes a través de un lenguaje de comunicación. Se pueden crear comunidades de agentes, los llamados sistemas multiagente.
- **Proactivo**
  Su actuación no solo va dirigida por cambios en el entorno sino que pueden actuar en función de sus propios objetivos, tomando la iniciativa en un momento dado.
- **Orientadas a Objetivos**: Un agente es capaz de manejar complejas tareas de alto nivel. La decisión de que una tarea sea dividida en subtareas y el orden en que estas subtareas se ejecutan debe ser hecho por el propio agente.

La *continuidad temporal* de los agentes de software se basa en la persistencia. Los agentes de software permanecen residentes o persistentes como procesos de fondo después de ser lanzados, tomando decisiones y actuando en su ambiente independientemente (*autonomia*), los agentes de software reducen la sobrecarga de trabajo humano interactuando con el usuario solo a la hora de entregar resultados. Este tipo de automatización puede inducir a una alta performance en términos de volumen y velocidad.

El agente dentro del ambiente de software requiere conocer de los protocolos de *comunicación* del dominio. Protocolos tales como SQL para bases de datos, HTTP para la WWW y las llamadas API para los sistemas operativos deben ser preprogramados dentro de los agentes de software limitando su rango de aplicación.

La *reactividad* para agentes de software no inteligente esta generalmente limitada a proveer acciones de control y la generación de reportes que requieren una revisión humana. Tales agentes a menudo tienden a ser frágiles en un ambiente dinámico por lo que necesitan una modificación del programa para restaurar su funcionamiento.

3.3 Agentes de Software Inteligente

Un agente de software inteligente usa la inteligencia artificial (IA) con el propósito de lograr los objetivos del cliente.

La inteligencia artificial se puede considerar la imitación de la inteligencia humana por medios mecánicos. Los clientes pueden entonces reducir su sobrecarga de trabajo delegando a los Agentes de Software Inteligente tareas que normalmente requieren inteligencia similar a la humana. La palabra "agente" hoy en día significa Agentes de Software Inteligente y así es como lo tomaremos a lo largo de la tesis.

Hay algunas definiciones adicionales de los agentes inteligentes:

"Los agentes inteligentes son entidades de software que realizan un conjunto de operaciones en beneficio de un usuario u otro programa, con algún grado de independencia o autonomía y al hacerlo emplean algún conocimiento o representación de los objetivos o deseos de usuario" [21].

"Los agentes son sistemas computacionales que habitan en ambientes dinámicos complejos, perciben y actúan de forma autónoma en este ambiente, realizando un conjunto de tareas y cumpliendo objetivos para los cuales fueron diseñados" [22].

"Un agente es una entidad que percibe su medio ambiente a través de sensores y actúa sobre aquel a través de actuadores" [23].

"Es una entidad persistente de software dedicado aun propósito especifico. "Persistencia" distingue los agentes de las subrutinas; los agentes tienen sus propias ideas acerca de cómo ejecutar las tareas, tienen su propia agenda. El propósito especial los distingue de las aplicaciones multifunción; los agentes son típicamente mucho mas pequeños " [24].

Adicionalmente a las características mencionadas anteriormente para los agentes de software los investigadores en Inteligencia Artificial atribuyen otras características adicionales a los Agentes de Software Inteligente.

- **Movilidad**
  Habilidad del agente para moverse a través de una red electrónica.
- **Veracidad**
  Se asume que el agente dice siempre la verdad, esto es comunica siempre datos verdaderos.
- **Benevolencia**
  El agente no incorpora objetivos que sean incompatibles.
- **Racionalidad**: Es puramente la asunción que un agente actuara para alcanzar sus objetivos y ella no actuara en contra de estas..
- **Adaptabilidad**
  Tiene capacidad de aprendizaje lo que le permite adaptarse al entorno. Esta propiedad esta pensada para entornos muy dinámicos con cambios drásticos.

La *autonomía* de los Agentes de Software Inteligente es de lejos mas absoluta. Estos agentes tienen la capacidad de generar e implementar nuevas reglas de comportamiento que los seres humanos nunca podrían tener la oportunidad de revisar. La relación de confianza entre el cliente y el agente debe ser grande especialmente cuando se le confía al agente el consumo de recursos del cliente.

La *comunicación* practicada por los agentes agrega una funcionalidad de orden mas alto a la mezcla de capacidades. Además de comunicarse con su ambiente para recoger datos y generar cambios los agentes pueden a menudo analizar la información para hallar patrones no obvios o escondidos extrayendo conocimiento de la información en bruto.

La *adaptabilidad* y *reactividad* en estos agentes pueden incluir el auto monitoreo hacia el logro de las metas del cliente con un aprendizaje en forma continua para mejorar su funcionamiento. Los mecanismos de adaptabilidad en los agentes son menos frágiles a los cambios en el ambiente y pueden mejorar actualmente. Además la respuesta al cliente puede ir mas allá de tal manera que el agente puede inferir lo que el cliente desea así el cliente no conozca o no pueda expresar sus objetivos adecuadamente.

Hasta ahora no se ha logrado tener un consenso acerca de la importancia relativa de cada una de estas características del agente como un todo. Lo que se observa es que estas características distinguen a los agentes de los programas ordinarios.

3.3.1 Niveles de Inteligencia

Se pueden establecer niveles de inteligencia tal como lo sugiere Lee [25] para los agentes Web:

- **Nivel 0:** Recuperan documentos directamente.

- **Nivel 1:** Proveen facilidades de búsqueda iniciadas por el usuario para hallar páginas relevantes.
- **Nivel 2:** Mantienen un perfil del usuario, supervisan la información de Internet; notifican a los usuarios cuando información relevante es hallada o cambiada.
- **Nivel 3:** Tienen un mecanismo de aprendizaje y deducción del perfil del usuario para ayudar al usuario.

3.3.2 Estructura de un Agente Inteligente

Un agente esta formado por:

- **Programa de agente**

Función que implementa la correspondencia entre percepciones y acciones.

- **Arquitectura** (computadora, hardware específico)

Proporciona al programa las percepciones, ejecuta el programa y alimenta al actuador con las acciones determinadas por el programa.

3.3.3 Agentes y Motores de Búsqueda

Los motores de búsqueda y los metabuscadores como se ha visto anteriormente en el capitulo 2 han jugado un rol muy importante en el proceso de extracción de información pero hoy en día esta emergiendo un nuevo enfoque para el proceso de extracción de información y son los llamados Agentes de Software Inteligente. Estos agentes tienen algunas ventajas sobre los motores de búsqueda actuales como veremos a continuación en la tabla 3.1:

Motores de Búsqueda	Agentes
La búsqueda de información esta basada en una o mas palabras claves dadas por un usuario. Se supone que el usuario es capaz de formular un conjunto de palabras claves para recuperar la información.	Los agentes son capaces de buscar la información mas inteligentemente al igual que las enciclopedias permiten que se busquen términos relacionados.
La información ofrecida por los meta buscadores esta hecha sobre la base de información que esta disponible en Internet. Genera mucho trafico de datos y pierde eficiencia.	Agentes de usuario individuales pueden crear sus propias bases de datos de conocimiento sobre información disponible en Internet que esta actualizada y es expandida después de cada extracción.
La búsqueda de información esta limitada frecuentemente a algunos servicios de Internet como WWW.	Los agentes alivian al usuario de la necesidad de preocuparse sobre como y donde hacer la consulta sino de que es lo que busca.
Muchos motores de búsqueda son independientes del dominio de la búsqueda	Agentes de software pueden extraer información basada en contexto
Los motores de búsqueda no aprenden a través de las búsquedas del usuario.	Los agentes se pueden ajustar a las preferencias del usuario individual.

Tabla 3.1: Ventajas de los Agentes con respecto a los Motores de Búsqueda.

3.4 Aplicaciones

Los agentes tienen una gran cantidad de aplicaciones, entre los que podemos mencionar:
Aplicaciones industriales
Aplicaciones médicas
Áreas de entretenimiento (juegos, teatro interactivo)
Aplicaciones comerciales
Gestión de la Información
Comercio Electrónico

3.4.1 Aplicaciones en Gestión de la Información

- **Agentes Desktop**: Agente de software que se ejecuta localmente en una máquina realizando diferentes tareas con el objetivo de beneficiar a su usuario.
  Podemos considerar:
  Agentes de sistema operativo. Ejm: setup de máquinas, personalización de entornos, automatización de tareas repetitivas.
  Agentes de aplicación: Ejm. Recuperación de la información de la Web, filtrado de email, automatización de consultas repetitivas a bases de datos.

- **Agentes de Internet**: Agente de software que accede a la información distribuida en Internet para desarrollar tareas en beneficio de su usuario. Podemos considerar:
  Agentes de búsqueda en la Web
  Agentes de filtrado de información
  Agentes de notificación
  Agentes móviles
  Softbots

- **Agentes de Intranet**: Agente software que ayuda al desarrollo de tareas empresariales en beneficio de los empleados, clientes y suministradores de una empresa.
  Podemos considerar:
  Mecanismo de comunicación interno para publicación de información corporativa
  Soporte automatizado en ayudas técnicas.
  Servicios de compartición de conocimiento entre usuarios.
  Agentes de automatización de procesos.
  Agentes de bases de datos.
  Agentes de búsqueda, filtrado, etc. en Intranet.

Capítulo 4

Algoritmos Genéticos

4.1 Introducción

Los Algoritmos Genéticos (GA) fueron introducidos por John Holland [26] en 1975 inspirándose en el proceso observado en la evolución natural de los seres vivos. En palabras del propio J. Holland:

"Se pueden encontrar soluciones aproximadas a problemas de gran complejidad computacional mediante un proceso de "evolución simulada",

Toda tarea de búsqueda y optimización posee varios componentes, entre ellos: el espacio de búsqueda donde están consideradas todas las posibilidades de solución de un determinado problema y la función de evaluación, una manera de evaluar a los miembros en el espacio de búsqueda.

Las técnicas de búsqueda y optimización tradicionales se inician con un único candidato que iterativamente es manipulado utilizando algunas heurísticas (estáticas) directamente asociadas al problema a ser resuelto. Generalmente estos procesos heurísticos no son algorítmicos y su simulación en computadoras puede ser muy compleja. A pesar de que estos métodos no son suficientemente robustos esto no implica que sean inútiles. En la practica son ampliamente utilizados con éxito en numerosas aplicaciones.

Por otro lado las técnicas de computación evolutiva operan sobre una población de candidatos en paralelo. Así ellos pueden hacer una búsqueda en las diferentes áreas del espacio de solución, asignando un numero de miembros apropiado para la búsqueda en varias regiones.

Los Algoritmos Genéticos (AGs) difieren de los métodos tradicionales de búsqueda y optimización principalmente en cuatro aspectos.

- Los AGs trabajan con un conjunto de soluciones o parámetros codificados (hipótesis o individuos).
- Los AGs trabajan con una población y no con un único punto.
- Los AGs utilizan la información de la función de evaluación y no derivadas de otro conocimiento auxiliar.
- Los AGs utilizan reglas de transición probabilísticas y no determinísticas.

Los AGs son muy eficientes para la búsqueda de soluciones óptimas o aproximadamente óptimas en una gran variedad de problemas, pues no imponen muchas limitaciones como las encontradas en los métodos de búsqueda tradicionales.

Los Algoritmos Genéticos son algoritmos de optimización global, basados en mecanismos de selección natural y de genética. Representan una estrategia de búsqueda paralela, estructurada y aleatoria, que se dirige hacia puntos de alta aptitud es decir a puntos en los cuales la función a ser minimizada (o maximizada) tiene valores relativamente bajos (o altos).

El punto de partida para la utilización de los algoritmos genéticos, como herramienta para la solución de problemas, es la representación de estos problemas de manera que los Algoritmos Genéticos puedan trabajar adecuadamente sobre ellos. La mayoría de representaciones son genotípicas, utilizando vectores de tamaño finito en un alfabeto finito.

Tradicionalmente, los individuos son representados genotípicamente por vectores binarios, donde cada elemento de un vector denota una presencia (1) o una ausencia (0) de una determinada característica (genotipo). Estos elementos pueden ser combinados formando las características reales del individuo (fenotipo). Teóricamente, esta representación es independiente del problema, pues una vez encontrada la representación en vectores binarios las operaciones estándar pueden ser utilizadas, facilitando su empleo en diferentes clases de problemas.

Inicialmente se genera una población formada por un conjunto aleatorio de individuos que pueden ser vistos como posibles soluciones del problema. Durante el proceso evolutivo esta población es evaluada: para cada individuo se da un valor o índice, reflejando su habilidad de adaptación a determinado ambiente. Un porcentaje de los mas adaptados es mantenido, en tanto los otros son descartados (darwinismo). Los miembros mantenidos por la selección pueden sufrir modificaciones en sus características fundamentales a través de cruzamientos (crossover) y mutaciones en la recombinación genética, generando descendientes para la próxima generación. Este proceso llamado de reproducción es repetido hasta que una solución satisfactoria sea encontrada o se cumpla alguna condición de parada. Aquí abajo mostramos el seudo código un Algoritmo Genético genérico.

Algoritmo Genético

```
{ t = 0;
iniciar_poblacion (P, t)
evaluación (P, t);
repetir hasta (t = d)
        { t = t +1;
        selección de los padres (P, t);
        cruzamiento (P, t);
        mutación (P, t);
        evaluación (P, t);
        sobrevivientes (P, t)
        }
}
```

donde:

t - Tiempo actual;
d – Tiempo determinado para finalizar el algoritmo;
P - Población

Estos algoritmos a pesar de ser computacionalmente muy simples, son bastante poderosos. Además, ellos no están limitados por suposiciones sobre el espacio de búsqueda, relativas a continuidad, existencia de derivadas, etc. La búsqueda en problemas reales está repleta de discontinuidades, ruidos y otros aspectos. Por tanto los métodos que dependan fuertemente de restricciones de continuidad y de existencia de derivadas son adecuados apenas para problemas en un dominio limitado.

4.2 Aspectos Teóricos

4.2.1 Concepto de Esquema

La teoría tradicional de los Algoritmos Genéticos asume que a un nivel muy general de descripción, los Algoritmos Genéticos trabajan descubriendo, acentuando y combinando buenos "bloques de construcción" de soluciones en una forma altamente paralela. La idea es que las mejores soluciones tiendan a ser construidas de los mejores bloques de construcción que son combinaciones de valores de bits que confieren el mas alto ajuste sobre los individuos en los cuales ellos están presentes.

Holland (1975) introdujo la noción de esquemas para formalizar la noción informal de "bloques de construcción". Un esquema es un conjunto de cadenas de bits (individuos) que pueden ser descritas por una plantilla hecha de unos, ceros y asteriscos, los asteriscos representan ceros o unos. Por ejemplo el esquema H = 1****1 representa el conjunto de todas las cadenas de 6 bits que empiezan y finalizan con 1. H es denotado para usar esquemas porque los esquemas definen hyperplanos (planos de varias dimensiones) en el espacio de l dimensiones de las cadenas de l bits. Las cadenas que se ajustan a esta plantilla (Por ejm. 100111 y 110011) son llamadas instancias de H. Dos propiedades importantes de los esquemas son definidas abajo:

a) El orden de un esquema H, denotado por $\sigma(H)$: Numero de bits definidos (no asteriscos) (Por ejemplo $\sigma(10\#01) = 4$).
b) La longitud definida de H, denotada por d(H): La distancia entre sus bits definidos más externos (Por ejemplo d(#10#10##) = 4).

Note que no cualquier subconjunto del conjunto de cadenas de bits de longitud l puede ser descrito como un esquema, de hecho la gran mayoría no puede. Cualquier cadena de bits de longitud l es una instancia de 2^l diferentes esquemas. Entonces dada una población de n cadenas esta contiene instancias entre 2^l y n x 2^l diferentes esquemas. Si todas las cadenas son idénticas entonces habrá instancias de exactamente 2^l diferentes esquemas; de otra manera el número es menor o igual a n x 2^l. Esto significa que dada una generación, mientras el Algoritmo Genético esta explícitamente evaluando la aptitud promedio de la población de n cadenas al tiempo t dada por:

$$\bar{f}(t) = \frac{\sum^{n} f(x_i)}{n} \qquad (4.1)$$

.x_i: cadena i-esima.

éste esta implícitamente estimando la aptitud promedio de un numero mucho más grande de esquemas. Ahora dejemos que H sea un esquema con al menos una instancia presente en la población al tiempo t, m(H,t) el número de instancias de H en el tiempo t entonces $\hat{u}(H,t)$ la aptitud promedio de las instancias de H en la población al tiempo t queda definida por:

$$\hat{u}(H,t) = \frac{\sum_{x \in H} f(x)}{m(H,t)} \qquad (4.2)$$

4.2.2 Operador de reproducción

Un principio básico de funcionamiento de los AGs es que un criterio de selección genere después de cada generación un conjunto inicial de individuos más aptos. Un operador de reproducción designa a cada individuo de la población una probabilidad de ser seleccionada para permanecer en la próxima población. La mayoría de los métodos de selección son diseñados para escoger preferencialmente los individuos con mayores valores de aptitud, $f(x_i)$, aunque no exclusivamente, a fin de mantener una diversidad de población. Un método de selección muy utilizado es el *método de la ruleta*. En este método a cada individuo en función directa a su valor de aptitud se le asigna un valor del área de la ruleta. Así a los individuos con alta aptitud se les da una porción mayor de la ruleta que los individuos de bajo valor de aptitud. Finalmente la ruleta es girada un determinado numero de veces, dependiendo del tamaño de la población y son escogidos, como individuos que participaran en la próxima generación, los individuos que han sido sorteados en la ruleta. (Fig. 4.1)

Individuo S_i	Aptitud $f(S_i)$	Aptitud Relativa
S1 10110	2.23	0.14
S2 11000	7.27	0.47
S3 11110	1.05	0.07
S4 01001	3.35	0.21
S5 00110	1.69	0.11

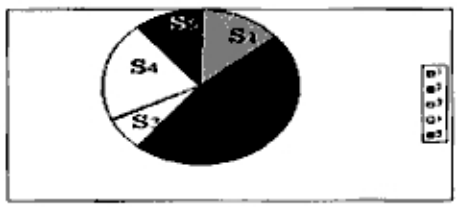

Fig. 4.1: Distribución del área de la ruleta en razón directa a la aptitud.

Un conjunto de operaciones se requieren para que dada una población, se consiga generar poblaciones sucesivas que mejoren su aptitud en el tiempo. Estas operaciones son: *cruzamiento* (crossover) y *mutación*. Estas son utilizadas para asegurar que la nueva generación sea totalmente nueva, o sea, una población se diversifica y mantiene características de adaptación adquiridas por las generaciones anteriores. Para prevenir que los mejores individuos no desaparezcan de la población por la manipulación de los operadores genéticos, ellos pueden ser automáticamente colocados en la próxima generación a través de una reproducción elitista.

Este ciclo se repite un determinado número de veces. Durante el proceso los mejores individuos así como algunos datos estadísticos o valores de aptitud, pueden ser almacenados para su posterior evaluación.

4.2.3 Operador de Cruzamiento

El operador de cruzamiento es el operador responsable primero, de introducir nuevos organismos recombinando los organismos ya existentes y segundo tienen un efecto de selección eliminando los esquemas de baja aptitud.

Es considerado el operador genético predominante, por eso es aplicado con una probabilidad dada por la taza de cruzamiento Pc, que debe ser mayor que la tasa de mutación.

Los Operadores de Cruzamiento más utilizados son:

- **De un punto:** Se elige aleatoriamente un punto de ruptura en los padres y se intercambian sus bits (Fig. 4.2).
- **De dos puntos:** Se eligen dos puntos de ruptura al azar para intercambiar.
- **Uniforme:** No utiliza puntos de cruce, en cada bit se elige al azar un padre para que contribuya con su bit al del hijo, mientras que el segundo hijo recibe el bit del otro padre.
- **PMX, SEX:** Son operadores más sofisticados fruto de mezclar y aleatorizar los anteriores.

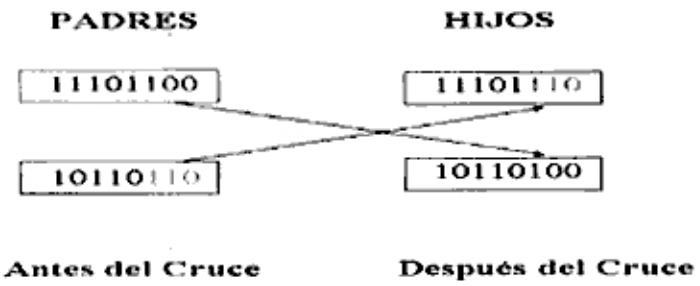

Fig. 4.2: Cruzamiento de un punto.

4.2.4 Operador de Mutación

El operador de mutación es necesario para la introducción y mantenimiento de la diversidad genética de la población, alterando arbitrariamente una o mas componentes de la estructura escogida (Fig. 10), proveyendo así medios para la introducción de nuevos elementos en la población. De esta forma una mutación asegura que la probabilidad de llegar a cualquier punto del espacio de búsqueda nunca sea cero, además, de bordear el problema de los mínimos locales, pues con este mecanismo, se altera ligeramente la dirección de búsqueda. El operador de mutación es aplicado a los individuos con una probabilidad dada por la tasa de mutación p_m: generalmente se utiliza una tasa de mutación pequeña, por eso es un operador genético secundario.

Antes de la Mutación: **111[0]000**

Después de la Mutación: **1110000**

Fig. 9. Mutación de un punto.

4.2.5 Teorema del Esquema

A través del análisis de la actuación de los operadores genéticos es posible derivar uno de los resultados teóricos más importantes en el que se basa la potencia de los Algoritmos Genéticos, el Teorema de los Esquemas de Holland[26].

Nuestro objetivo es calcular E(m(H,t+1)) que es el número esperado de instancias de H al tiempo t+1. Para esto empezamos definiendo la probabilidad de que se elija un individuo para su reproducción por:

$$\Pr(x) = \frac{f(x)}{\sum_{i=1}^{n} f(x_i)} \qquad (4.3)$$

de la ecuación 4.1

$$\Pr(x) = \frac{f(x)}{\hat{f}(t)n} \qquad (4.4)$$

Luego ahora calculemos la probabilidad que sea un individuo del esquema H,

$$P(x) = \sum_{x \in H} \frac{f(x)}{\bar{f}(t)n} = \frac{\sum_{x \in H} f(x)}{\bar{f}(t)n} = \frac{\hat{u}(H,t)m(H,t)}{\bar{f}(t)n} \qquad (4.5)$$

Por lo tanto el numero esperado de instancias E(m(H,t+1)) de H al tiempo t+1, ignorando los efectos del cruce y mutación será:

$$E(m(H,t+1)) = \frac{\hat{u}(H,t)m(H,t)}{\bar{f}(t)} \qquad (4.6)$$

Entonces aunque los Algoritmos Genéticos no calculan $\hat{u}(H,t)$ explícitamente, el incremento o decremento de la instancias de un esquema en la población dependen de esta cantidad.

Los operadores de cruce y mutación pueden ambos crear y destruir instancias de H. Analizaremos primero los efectos destructivos del cruce y mutación, es decir aquellos que disminuyen el numero de instancias de H. Al incluir estos efectos vamos a modificar el lado derecho de la ecuación 4.6 para dar un limite inferior a E(m(H, t+1)). Si consideramos que p_c es la probabilidad de que un cruce de punto simple sea aplicado a un individuo y suponiendo que una instancia del esquema H sea elegida a ser un padre. Se dice que el esquema H sobrevive sobre un cruce de punto simple si uno de los hijos es también una instancia del esquema H. Podemos entonces dar un limite inferior a la probabilidad $S_c(H)$ que H sobrevivirá a un cruce de punto simple:

$$S_c(H) \geq 1 - p_c\left(\frac{d(H)}{l-1}\right) \qquad (4.7)$$

Donde d(H) es la longitud definida de H y l es la longitud del individuo en el espacio de búsqueda. Esto es, un cruce que ocurre dentro de la longitud definida de H puede destruir H (es decir puede producir hijos que no sean instancias de H), así nosotros multiplicamos la fracción del individuo que H ocupa por la probabilidad de cruce para obtener un limite superior en la probabilidad de que ella sea destruida. (Este valor es un limite superior porque algunos cruces dentro de las posiciones definidas por el esquema no serán destruidos, por Ejm. Si dos individuos idénticos se cruzan entre ellos). Sustrayendo este valor de 1 nos da un limite inferior en la probabilidad de supervivencia $S_c(H)$. En resumen, la probabilidad de supervivencia debido al cruce es mas alto para esquemas más cortos.

Los efectos disruptivos de la mutación se cuantifican de la siguiente manera. Sea p_m la probabilidad de que un bit sea mutado. Luego $S_m(H)$, la probabilidad de que un esquema H sobreviva a una mutación de una instancia de H, es igual a $(1-p_m)^{\sigma(H)}$, donde $\sigma(H)$ es el orden de H (es decir el numero de bits definidos en H). Esto es para cada bit, la probabilidad de que el bit no será mutado es $(1-p_m)$, así la probabilidad de que los bits definidos del esquema H no serán mutados es esta cantidad multiplicada por sí misma $\sigma(H)$ veces. En resumen, la probabilidad de sobrevivencia debido a la mutación es mas alta para esquemas de orden mas bajo.

Estos efectos de cruce y mutación pueden ser usados para reformular la ecuación 4.6:

$$E(m(H,t+1)) \geq \frac{\hat{u}(H,t)m(H,t)}{\bar{f}(t)}(1-p_c\frac{d(H)}{l-1})\left[(1-p_m)^{o(H)}\right] \qquad (4.8)$$

Este resultado es conocido como el Teorema del Esquema. Este teorema describe el crecimiento de un esquema de una generación a la siguiente. El teorema del Esquema implica que esquemas cortos, de bajo orden y cuya aptitud promedio permanece encima de la media recibirán un incremento exponencial de muestras (es decir instancias evaluadas) en el tiempo, ya que el numero de muestras de aquellos esquemas que no son destruidos y permanecen encima del promedio en aptitud se incrementan por un factor $\hat{u}(H,t)/\bar{f}(t)$ en cada generación.

4.2.6 Parámetros Genéticos

Es importante también analizar de que manera algunos parámetros influyen en el comportamiento de los Algoritmos Genéticos, para que se puedan establecer conforme a las necesidades del problema y los recursos disponibles.

Tamaño de la Población: El tamaño de la población afecta el desempeño y la eficiencia de los algoritmos genéticos. Con una población pequeña el desempeño puede caer, pues de este modo una población ofrece una pequeña cobertura del espacio de búsqueda del problema. Una gran población generalmente ofrece una cobertura representativa del dominio del problema, además de prevenir convergencias prematuras a soluciones locales. En tanto que para trabajar con grandes poblaciones, son necesarios mayores recursos computacionales o que el algoritmo trabaje por un periodo de tiempo mayor.

Tasa de Cruzamiento: Cuanto mayor es esta tasa, mas rápidamente nuevas estructuras serán introducidas en la población. Mas si esta es muy alta podría ocurrir que, estructuras con buenas aptitudes sean removidas rápidamente por lo que se podrían perder estructuras con alta aptitud. Mientras que con un valor muy bajo el algoritmo se puede tornar muy lento.

Tasa de Mutación: Una baja tasa de mutación previene que una posición dada quede estancado en un valor, además de hacer posible que se llegue a cualquier punto del espacio de búsqueda. Con una tasa muy alta la búsqueda se torna esencialmente aleatoria.

Intervalo de Generación: Controla el porcentaje de la población que será sustituida durante la próxima generación. Con un valor alto, la mayor parte de la población será sustituida, mas con valores muy altos puede ocurrir que se pierdan estructuras de alta aptitud. Con un bajo valor el algoritmo se puede tornar muy lento.

Capítulo 5

Extracción de Información

La extracción de información trata acerca de la representación, almacenaje, organización y acceso a los artículos de información. La representación y organización de los artículos de información deben proveer al usuario un fácil acceso a la información en el cual este interesado. Desafortunadamente, caracterizar la necesidad de información del usuario no es un problema simple. Consideremos por ejemplo el siguiente caso hipotético de necesidad de información del usuario en el contexto de la Web (World Wide Web):

> *Hallar todas la paginas que muestren los grupos de investigación en ingeniería electrónica de las universidades peruanas que,*
> *a)investiguen en tecnologías de la información*
> *b)que tengan apoyo del CONCYTEC. Para ser relevante la pagina debe incluir información sobre las publicaciones del grupo en los últimos cuatro años y además el correo electrónico o teléfono del jefe del grupo de investigación.*

Se puede observar que esta descripción total de la necesidad de información del usuario no puede ser usada para requerir información usando las interfaces de web de los motores de búsqueda actuales. En vez de eso el usuario primero debe trasladar esta necesidad de información en una *consulta (query)* que pueda ser procesada por el motor de búsqueda (o Sistema de Extracción de Información).

En su forma más común, esta traslación produce un conjunto de palabras claves, (keywords) que resumen la descripción de la necesidad del usuario. Dada la consulta del usuario, el objetivo principal de un sistema de extracción de información es recuperar información que pueda ser útil o relevante al usuario. El énfasis se da en la recuperación de información mas que en la recuperación de datos.

La recuperación de datos, en el contexto de un sistema de extracción de información, consiste principalmente en determinar que documentos de una colección contienen las palabras claves de la consulta del usuario que, muy frecuentemente, no es suficiente para satisfacer la necesidad de información del usuario, ya que de hecho, el usuario de un sistema de extracción de información esta mas interesado en la recuperación de información acerca de un tema (que no siempre esta bien estructurada y puede ser semánticamente ambigua), que en recuperar datos que satisfagan una consulta dada (tal como una base de datos relacional.).

Para ser efectivo en su intención de satisfacer las necesidades de información del usuario, un sistema de extracción de información debe hacer algo así como interpretar los contenidos de los artículos de información (documentos) en una colección y clasificarlos de acuerdo a un grado de relevancia a la consulta del usuario. Esta "interpretación" del contenido de un documento requiere extraer información sintáctica y semántica del texto del documento y usar esta información para compararla a la necesidad de información del usuario. La dificultad no es

únicamente saber como extraer esta información sino también como usar esto para decidir la relevancia. Por lo que la noción de relevancia esta en el centro de la extracción de información.

Tal como habíamos visto en el capitulo uno, existe un rápido incremento de la Internet y por lo tanto de la información contenida en ella, lo que dificulta la búsqueda de información ya que no esta estructurada y es muchas veces de muy baja calidad, por lo que se hace necesario investigar métodos automáticos para la búsqueda y organización de los documentos de texto tal que la información pueda ser accesada en forma rápida y precisa. El problema radica en que la información puede ser mal interpretada debido a las ambigüedades en el lenguaje natural o que la información requerida pueda ser definida tanto imprecisa como vagamente por el usuario.
Existen varios modelos de extracción de la información para archivos de texto, nosotros nos vamos a enfocar en el estudio del Método de Extracción Booleano, para luego revisar el Modelo de Espacio Vectorial que ha sido implementado en esta tesis.

5.1 Método de Extracción Booleana

El interés por la extracción de información ha existido antes de la Internet. El método de Extracción Booleano es el más simple de estos métodos de extracción y se basa en el uso de operadores Booleanos. Los términos en una consulta son unidos a través de Y, O y NO. Este método es usado a menudo en los motores de búsqueda debido a su rapidez por lo que puede ser usado en línea. Este método también tiene sus problemas. El usuario tiene que tener algún conocimiento del tópico de búsqueda para que la búsqueda sea eficiente ya que se puede correr el riesgo de que una palabra equivocada podría clasificar un documento relevante como no relevante. Los documentos recuperados están todos *igualmente* clasificados con respecto a su relevancia y el numero de documentos recuperados puede cambiarse únicamente reformulando la consulta. Se asume que los términos son independientes uno de otro.

La Extracción Booleana ha sido extendida y refinada para resolver estos problemas. Operaciones de asignación de pesos a los términos en un documento de acuerdo a su frecuencia en el documento hacen posible la clasificación de los documentos. La Extracción de Información Booleana ha sido combinada con la navegación basada en contenido usando concepto de redes, donde los términos de documentos ya previamente obtenidos son usados para refinar y expandir la consulta [27]. Los operadores Booleanos han sido reemplazados por operadores difusos para interpretar la consulta del usuario [28].

5.2 Asociación de Palabras

Las personas no usan las mismas palabras para describir el mismo concepto por lo que una asociación automática de palabras puede hacer más efectivo el proceso de búsqueda.
Existen métodos dependientes del lenguaje tales como el Algoritmo de Stemming, Thesaurus y el Método de Realimentación Relevante que no es dependiente del lenguaje.

El algoritmo de Stemming esta basado en dos estados; el despojo de los sufijos y la fusión de las palabras. El despojo de los sufijos puede ser alcanzado a través de una serie de reglas los cuales inducen a algunos errores (alrededor del 5%), ya que siempre hay excepciones a las reglas. El problema con el Método Stemming es que es dependiente del lenguaje.

El Método de Thesaurus o de Diccionario es una forma común de expandir las consultas y puede ser usado para ampliar el significado del termino así como estrechar el termino o simplemente hallar términos relacionados. El principal problema de usar el Método Thesaurus es que los términos tienen diferentes significados dependiendo del tema. Es por eso que se usa dentro de campos específicos donde puedan ser construidas manualmente.

El método de realimentación relevante se usa para reformular una consulta. Este método revisa automáticamente la consulta usando términos relevantes del documento previo y pondera estos para obtener un documento similar a los documentos previos. El Método Thesaurus y el de Realimentación por Relevancia han sido comparados y se ha mostrado que el Método de Relevancia usualmente se ejecuta mejor pero el Método de Thesaurus es mejor si el resultado de la consulta original es pobre[29].

5.3 Representación del Documento

Para reducir la complejidad de los documentos y facilitar su manejo, los documentos tienen que ser transformados desde una versión llena de texto a un vector documento que describe los contenidos del documento. Una definición de documento podemos decir es, que esta hecha de una asociación de términos unidos que tienen varios patrones de ocurrencia.

Estos patrones de ocurrencia son usualmente despreciados debido a la complejidad y solo una estadística simple de la palabra se usa. Inicialmente en el campo de la Extracción de Información se uso la frecuencia de términos como una característica descriptiva del contenido de los documentos y hoy todavía es el método de descripción de documentos más usado.

Utilizar la frecuencia de palabras co-ocurrentes es una técnica alentadora para encontrar asociaciones de palabras en los documentos. Esto no dice cualquier cosa acerca de la naturaleza del significado de los términos asociados; si son sinónimos, definen un término juntamente o es solo una redundancia de uso común. Recientemente, vectores de contexto para explotar estos patrones de co-ocurrencia han sido implementados en el campo de la Extracción de Información [30].

5.4 Modelo de Espacio Vectorial

El modelo de espacio vectorial para procesamiento de texto es el modelo de extracción de información mas ampliamente usado. El procedimiento de este modelo puede ser dividido en tres etapas. La primera etapa es la indexación del documento donde los términos más relevantes son extraídos del texto del documento. La segunda etapa es la ponderación de los términos indexados para mejorar la recuperación de documentos relevantes al usuario. La ultima etapa compara el documento con respecto a la consulta de acuerdo a una medida de similitud.

5.4.1 Indexación del Documento

Es obvio que muchas de las palabras en un documento no describen el contenido del documento tales como el, la, los, etc. Al usar una indexación automática del documento estas palabras poco significativas ("palabras función") son removidas del vector documento, por lo que el documento solo estará representado por palabras relevantes en el contenido. Esta indexación puede estar

basada en la frecuencia del termino, donde términos que tienen altas y bajas frecuencias dentro del documento son consideradas "palabras función". En la practica el método basado en la frecuencia de términos ha sido difícil de implementar en la indexación automática. En vez de esto se usa una lista de parada el cual remueve palabras de alta frecuencia (stop words), lo cual hace que el método de indexación sea dependiente del lenguaje. En general el 40 a 50% del numero total de palabras en un documento son removidos con la ayuda de esta lista de parada [31].

5.4.2 Ponderación de Términos

La ponderación de términos para el Modelo de Espacio Vectorial esta enteramente basado en una estadística simple de términos. Hay tres factores muy importantes en la ponderación de términos: el factor de frecuencia del término(tf), el factor de frecuencia del documento en la colección (df: document frecuency) y el factor de normalización de longitud(longitud euclidiana de vector documento). Estos tres factores son multiplicados juntamente para obtener el peso resultante del termino.

Un esquema de asignación de pesos a los términos dentro de un documento es usar la frecuencia de ocurrencia del termino dentro de aquel. La frecuencia de términos (tf) es algo así como una descripción del contenido para los documentos y es generalmente considerado como la base de los vectores documentos ponderados.

La importancia de un término en un documento no es independiente de la colección de documentos que esta siendo investigada. Por ejemplo el término "inca" no es muy importante cuando investigamos una colección de artículos sobre el Imperio del Tahuantinsuyo mientras que puede ser muy importante si estamos investigando acerca de las culturas antiguas de América. Esto nos indica que en cuantos mas documentos aparezca el término tanto menos importante este puede ser. Así que debemos incorporar en el proceso de asignación de pesos a los términos el factor idf (inverse document frecuency). Este factor es proporcional a 1/df.

Podemos observar que, documentos muy largos usualmente usan los mismos términos repetidamente, como consecuencia el factor de frecuencias de términos (tf) puede ser muy grande para estos documentos, además de que estos documentos tienen numerosos términos diferentes por lo que se incrementa el numero de coincidencias entre la consulta y el documento largo. Esta situación nos lleva a incrementar la posibilidad de recuperar documentos largos sobre los documentos cortos. Para compensar este efecto se usa la normalización de los pesos de los términos. La normalización es una forma de imponer alguna penalidad a los pesos de los términos de documentos largos. Cada vector documento es dividido por su longitud euclidiana, $\sqrt{w_1^2 + w_2^2 + \ldots w_n^2}$ donde w_i = tf*idf del i-ésimo término en el documento. Esto nos da un peso final para un término de:

$$\frac{tf * idf}{\sqrt{w_1^2 + w_2^2 + \ldots + w_n^2}} \qquad (5.1)$$

Los términos ausentes en un documento tienen peso cero.

La ponderación de términos ha sido definida para controlar la exhaustividad y especificidad de la búsqueda, donde la exhaustividad esta relacionada al "recall" y la especificidad a la precisión. Suponiendo que el conjunto de documentos relevantes respecto a la consulta en la colección de documentos pueda ser determinado, las dos cantidades "recall" y precisión están definidos como sigue:

$$recall = \frac{el\_numero\_de\_documentos\_relevantes\_recuperados}{el\_numero\_de\_documentos\_relevantes} \tag{5.2}$$

$$precision = \frac{el\_número\_de\_documentos\_relevantes\_recuperados}{el\_número\_de\_documentos\_recuperados} \tag{5.3}$$

5.4.3 Coeficientes de Similitud

La similitud en los Modelos del Espacio Vectorial se determina usando los coeficientes asociativos basados en el producto escalar del vector documento y el vector consulta, donde la superposición de palabras indica la similitud. El producto interno esta usualmente normalizado. El coeficiente más popular de similitud es el coeficiente coseno, que mide el ángulo entre el vector documento y el vector consulta.
Si Q es el vector consulta y D_i es la representación del vector documento i-ésimo, entonces la similitud entre la consulta y el documento se calcula como:

$$Sim(Q, D_i) = \sum_{ter\,\min\,oscomunes_j} q_j * d_{ij} \tag{5.4}$$

Donde t_j es un término que está presente tanto en la consulta como en el documento, q_j es el peso del termino t_j en la consulta y d_{ij} es su peso en el documento i. La suma se ejecuta sobre todos los términos t_j presentes tanto en la consulta como en el documento. De esta manera el cálculo de similitud usando el producto escalar nos genera una lista de documentos clasificados por su potencial utilidad.

Capítulo 6

Implementación y Resultados

6.1 El prototipo METAPERU

El prototipo es un sistema de búsqueda que dada la gran cantidad de información presente en la Web usa el método de Algoritmos Genéticos, para optimizar el proceso de búsqueda sobre tópicos específicos de una manera personalizada basándose en un perfil que provee el usuario. El prototipo consiste de un conjunto de agentes:

- **Metabusqueda**
- **Extractor de Texto**
- **Avanzado.**

La figura 6.1, muestra la arquitectura del prototipo, las tres componentes trabajan en forma independiente, el Avanzado que es el que implementa el método de Algoritmos Genéticos tiene incorporado la funcionalidad del Extractor de Texto y el Metabuscador para su proceso de optimización. La figura 6.2 muestra la pantalla principal del prototipo, donde los usuarios pueden escoger entre las tres opciones como: Metabusqueda, Extractor y Avanzado. El código fuente del prototipo se puede encontrar en el Apéndice A.

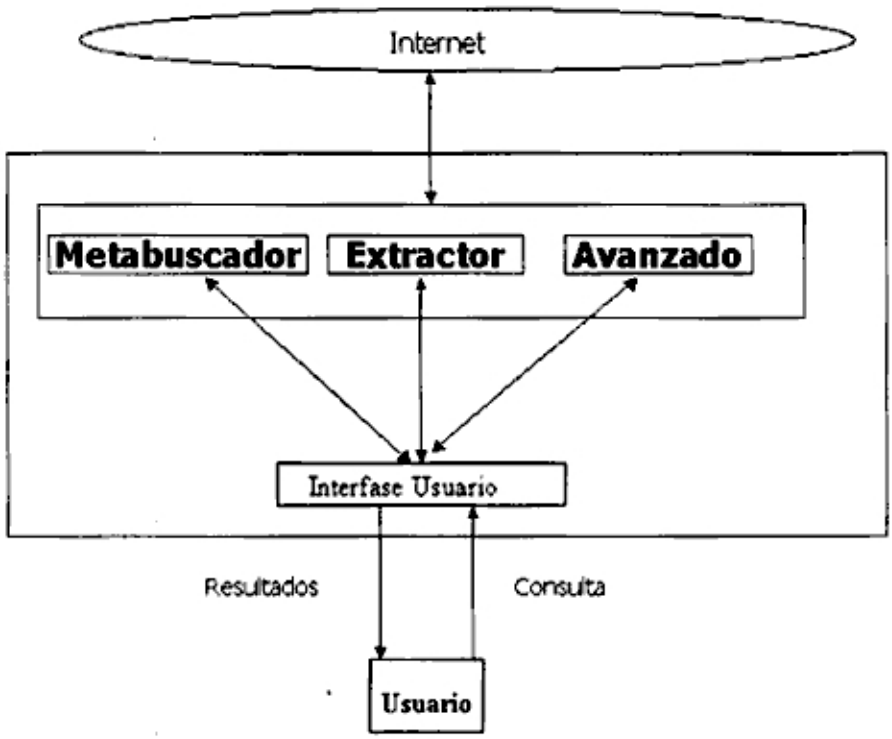

Fig. 6.1: Arquitectura de la interfase del prototipo MetaPeru.

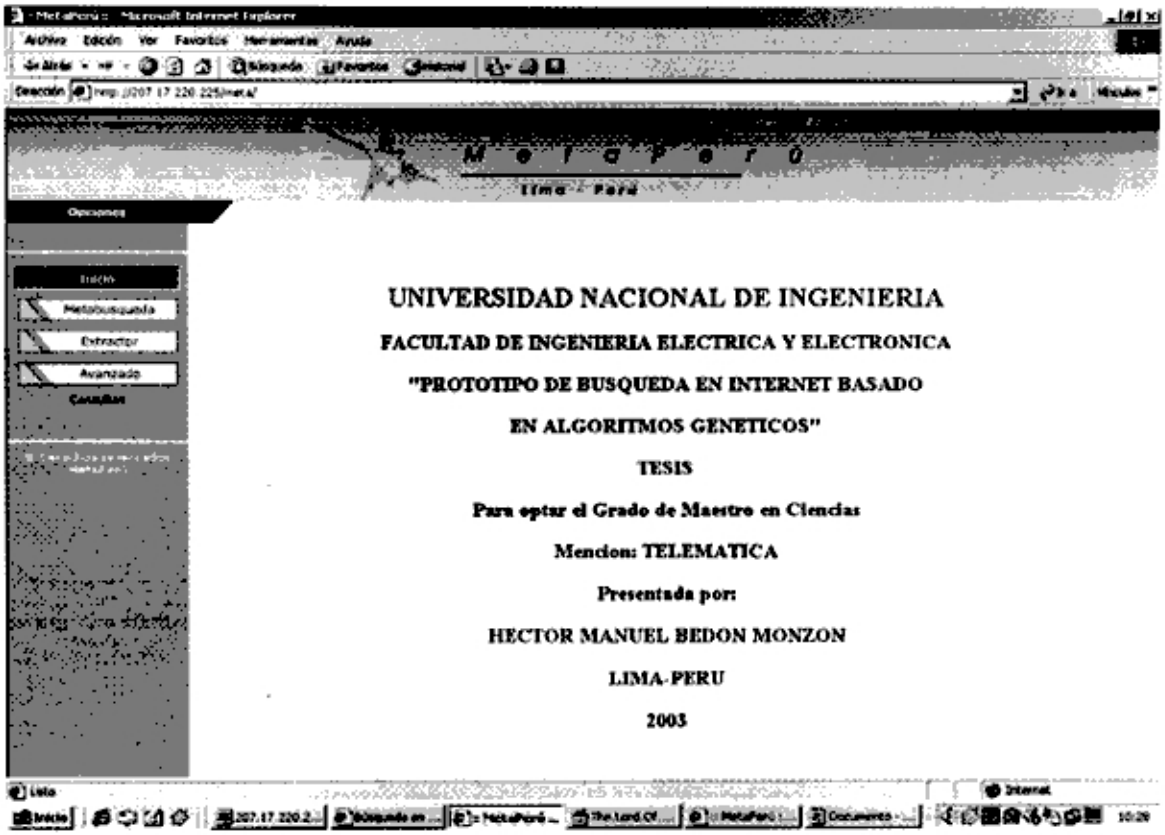

Fig. 6.2: Pantalla principal para el prototipo Metaperu©.

El prototipo esta implementado sobre una estación de trabajo que tiene las siguientes características:

Hardware

Sistema Operativo:	Linux Red Hat
Versión :	7.2,
Tipo de Sistema:	x86 (Pentium III) based PC,
Fabricante del Sistema:	Intel
Procesador:	x86 Family 6 model 8 Stepping 6GenuineIntel ~ 800 MHz
Cantidad total de memoria RAM:	192 MB

Software

Servidor de Páginas Web:	Apache;
Versión:	1.3.20
Servidor de Base de Datos:	MySQL
Versión:	3.23.41
Lenguaje Script:	PHP
Versión:	4.0.6

6.2 Metabusqueda

6.2.1 Arquitectura

Aquí se ha desarrollado un módulo aparte para el proceso de colección de información (el motor de metabusqueda) debido a que de vez en cuando (mas o menos tres en un año) los administradores de los motores de búsqueda cambian el formulario HTML de entrada así como el formato de salida, lo que nos lleva a una actualización de solo el proceso de colección. Este motor de metabusqueda nos permite cubrir un rango más amplio de búsqueda ya que un solo motor de búsqueda no seria capaz de cubrir todas las áreas.

De la sección anterior se hace claro que los motores de búsqueda se diferencian entre ellos por: el nivel de codificación del formulario HTML, es decir cada motor de búsqueda usa un método diferente para colectar las entradas de los usuarios, diferentes TAGS FORM, etc., y las opciones ofrecidas, búsqueda en la Web, grupos, directorios, traducción, etc. Dividiré la implementación del motor de metabusqueda en dos fases (Fig. 6.3):

- **Fase de Consulta**
- **Fase de Respuesta**

Fase de Consulta: Durante su operación la consulta del usuario se forma para ser llevada a cada motor de búsqueda. Esta fase se puede dividir en tres estados:

- **Análisis de Consulta**: En este primer estado se colecta la consulta del usuario.
- **Construcción de Consulta**: Durante el estado de construcción de la consulta, esta es codificada.
- **Envío de Consulta**: Por ultimo en este estado, el motor de metabusqueda agrega a la consulta específicos FORM TAGS y el método FORM indicado y lo envía al Script indicado por el parámetro ACTION dentro de la declaración <FORM>.

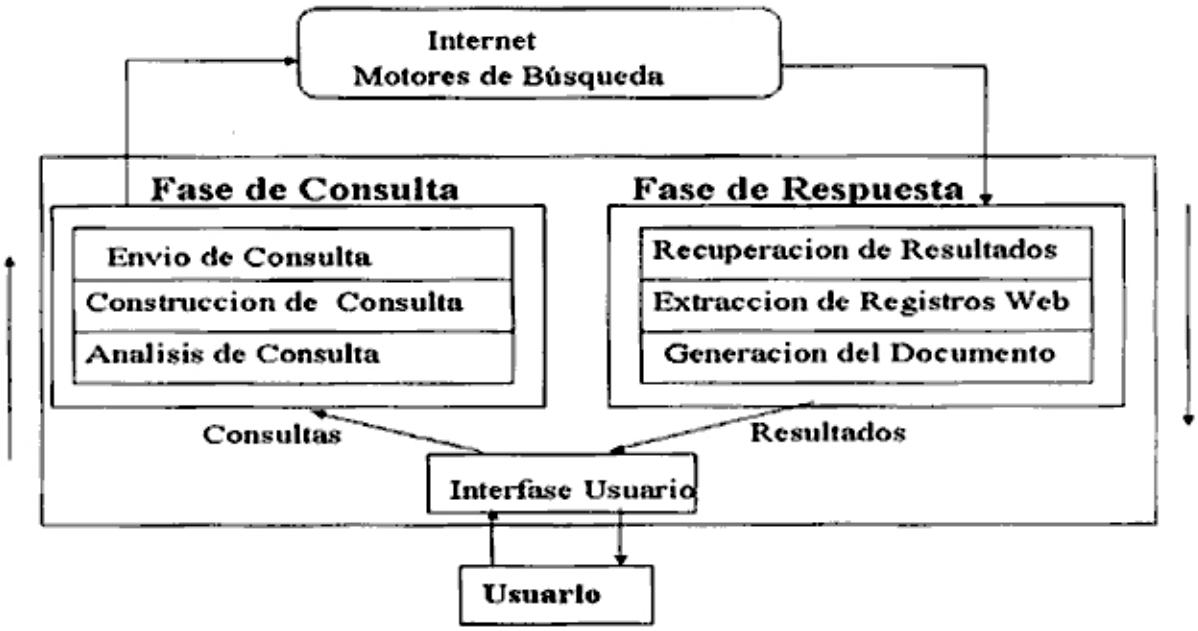

Fig. 6.3: La arquitectura del Metabuscador.

Fase de Respuesta: Durante esta fase se colectan los resultados de cada motor de búsqueda. Desde que cada motor de búsqueda usa una forma distinta de presentar sus resultados usaremos un conjunto de funciones especificas para cada motor de búsqueda. Esta fase la podemos dividir en tres estados:

- **Recuperación de Resultados:** Implica traer las páginas de respuesta de cada motor de búsqueda para el procesamiento.
- **Extracción de Registros Web:** En este estado los *registros web* son extraídos de las páginas de respuesta.
- **Generación de Documento:** Finalmente en este estado, se organizan y presentan los resultados que cada motor de búsqueda bajo un formato de presentación, que en nuestro caso es <titulo><url><descripción>. También durante este estado los resultados que tienen el mismo URL son removidos.

En lo que sigue discutiremos la interfase de nuestra implementación. Esta interfase es de mucha importancia ya que permite al usuario describir la consulta en un formato general que luego es adaptado a los requerimientos de cada motor de búsqueda.

6.2.2 Interfase de Consulta

Esta implementación ofrece una interfase común de consulta a tres motores de búsqueda (pueden agregarse más) tales como: Google[2], Altavista[3] y Yahoo[4]. La interfase de consulta soporta operadores lógicos básicos como "Y", "O" y "NO". Una vez elegida la opción de metabusqueda y el número de URLs no repetidos de respuesta por cada motor de búsqueda, tendremos una interfase como se muestra en la Figura 6.4. La interfase de metabusqueda esta basado en frames HTML y formularios. Los frames nos permiten separar el browser en múltiples ventanas y mostrar un documento HTML en cada ventana. De otro lado Action y Scripts en un frame pueden ser programados para controlar y actualizar los contenidos de frames adyacentes. La página de respuesta de Metabusqueda a la consulta *agentes de software inteligente* se muestra en la Figura 6.5. Las páginas de respuesta a la misma consulta de cada motor de búsqueda en forma individual, a modo de verificación se muestran en las Figuras 6.6, 6.7 y 6.8.

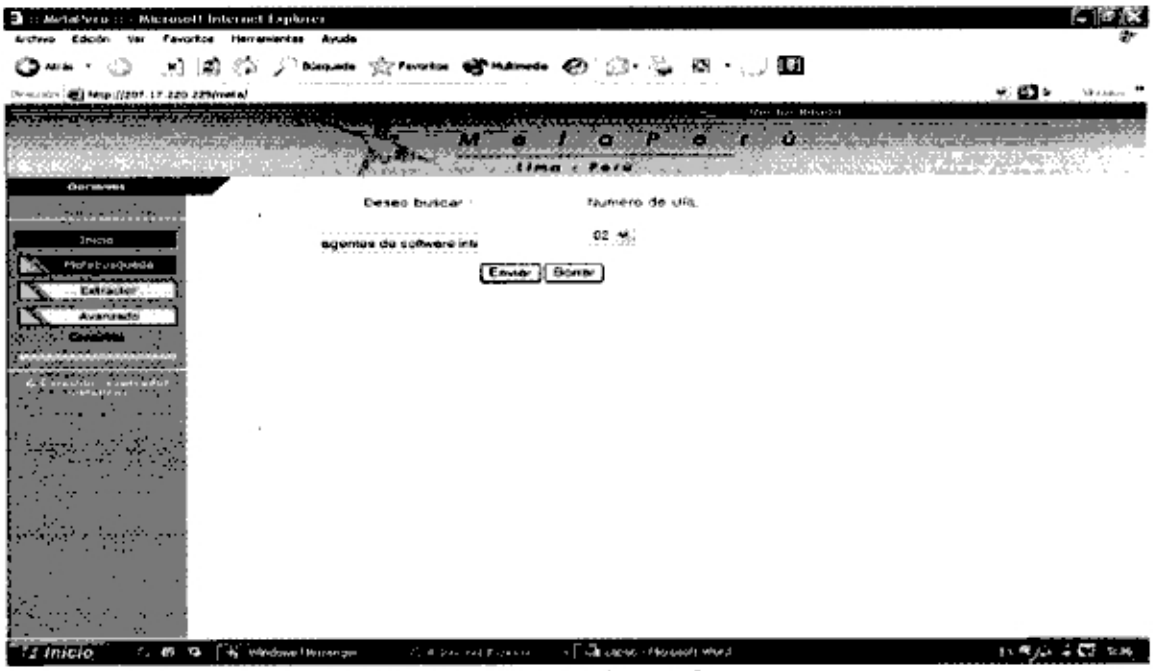

Fig. 6.4: Interfase de consulta para el Metabuscador.

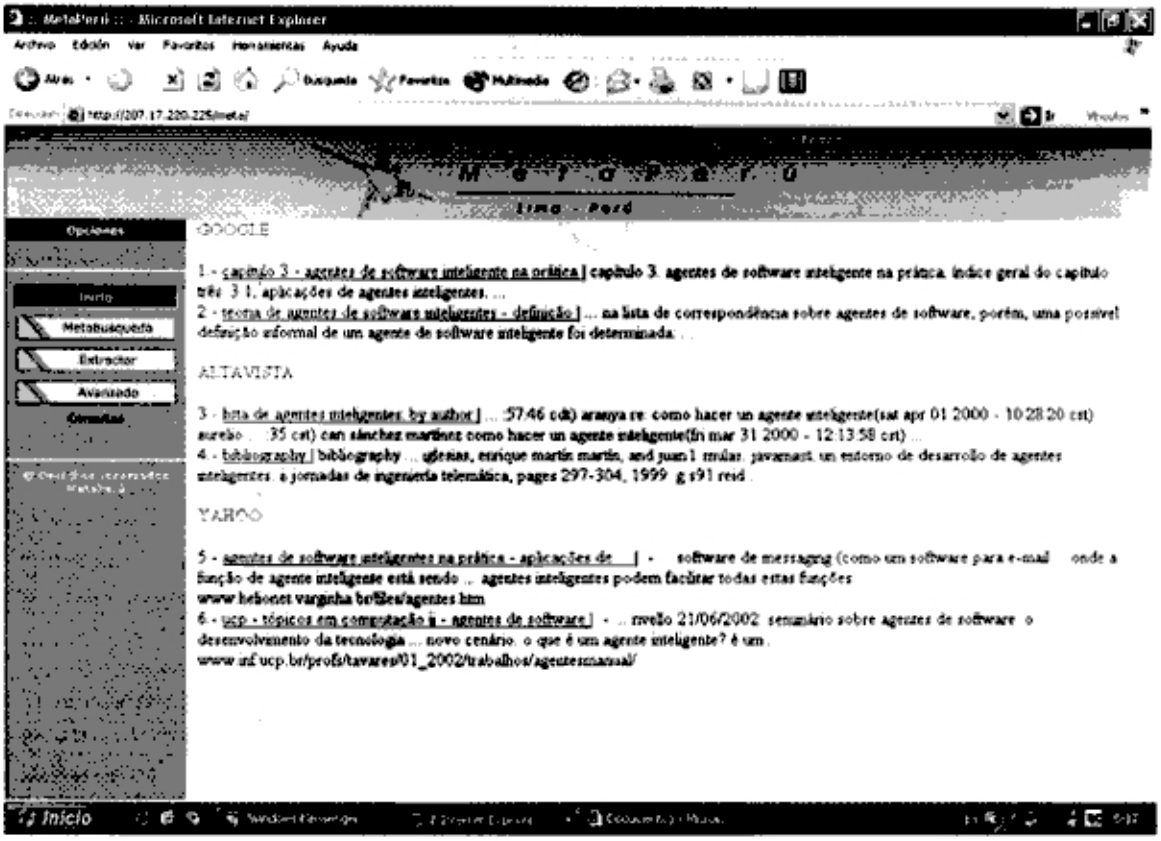

Fig. 6.5: Página de respuesta de Metabusqueda a la consulta "agentes de software inteligente".

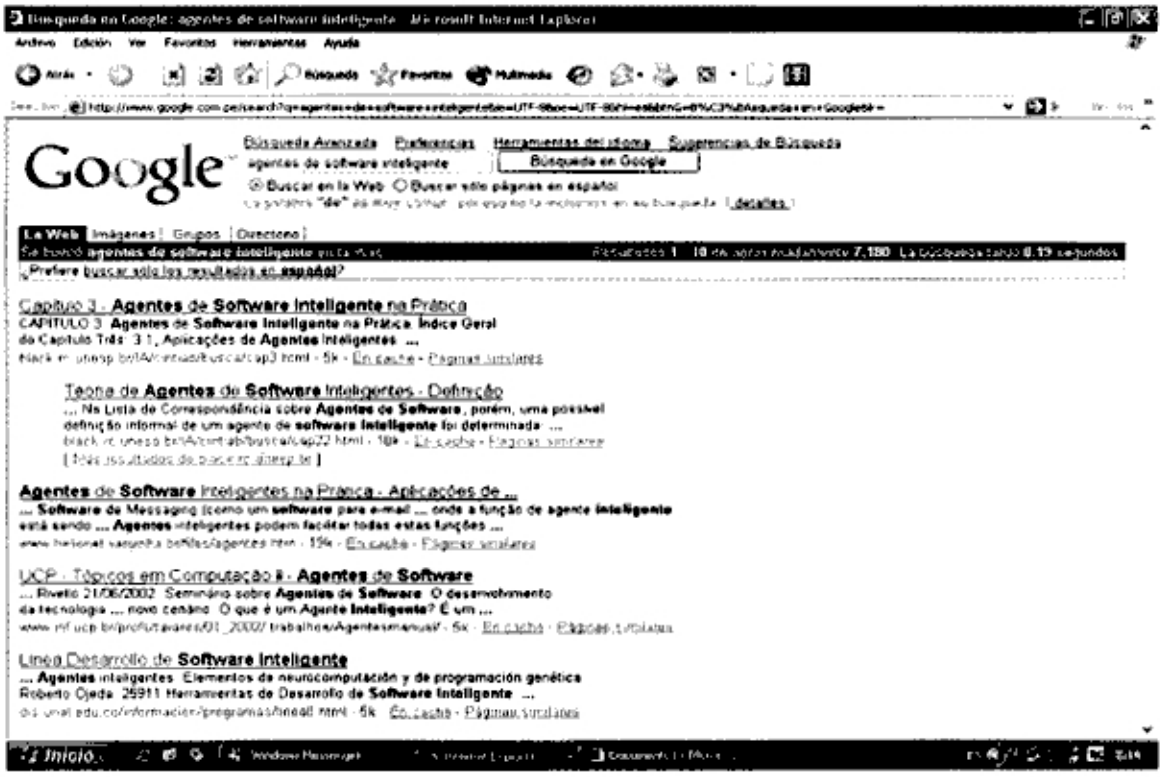

Fig. 6.6: Página de respuesta de Google a la consulta "agentes de software inteligente".

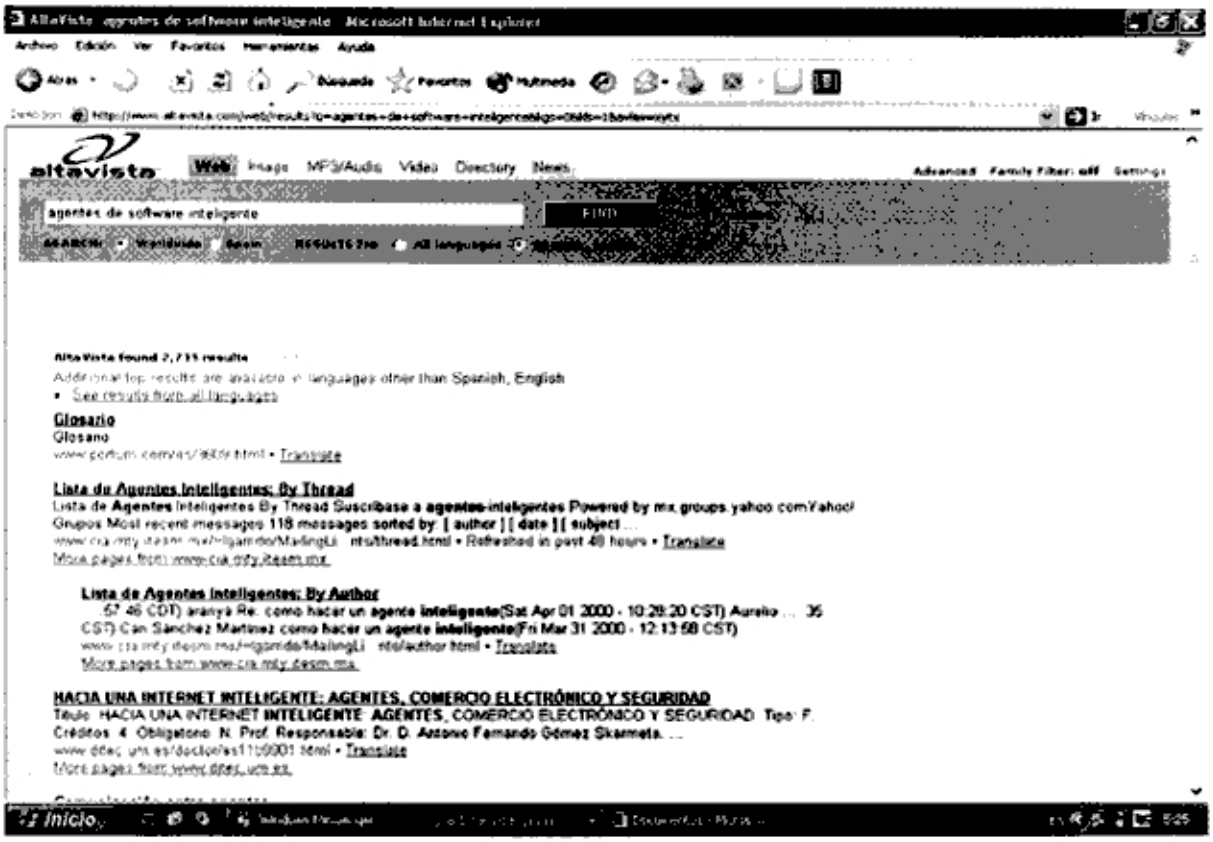

Fig. 6.7: Página de respuesta de Altavista a la consulta "agentes de software inteligente".

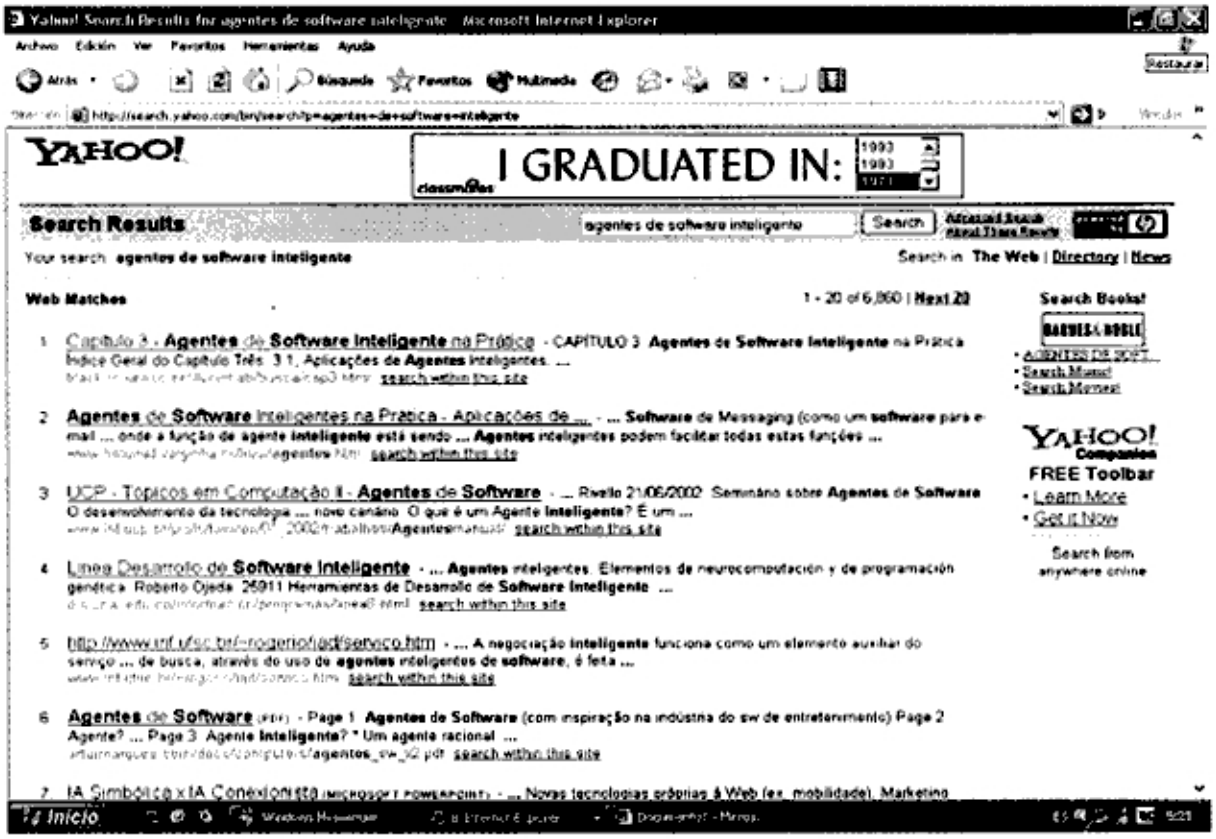

Fig. 6.8: Página de respuesta de Yahoo a la consulta "agentes de software inteligente".

6.3 Extractor

6.3.1 Arquitectura

Para lograr una cuantificación de la importancia de la página hallada por el motor de búsqueda con respecto al perfil del usuario, hubo la necesidad de construir un extractor de texto. Para el proceso de extracción se hace uso del modelo de espacio vectorial (Cap. 5.4) para procesamiento de texto por la cual un documento HTML es transformado desde una versión llena de texto a un vector documento que describe los contenidos del documento. El procedimiento para generar el extractor **(Fig. 6.9)** puede ser descrito en dos etapas.

- **La Indexación del Documento.**
- **Ponderación de Términos.**

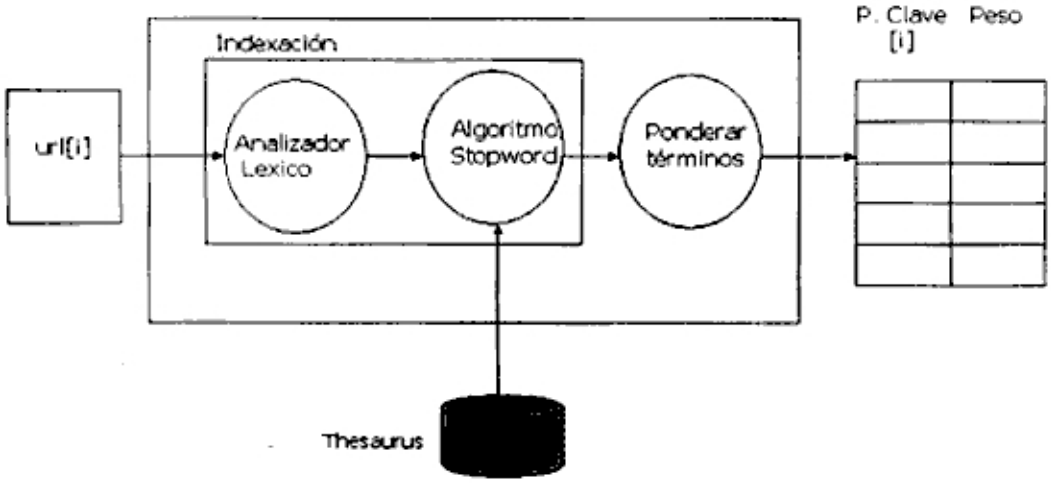

Fig. 6.9: Arquitectura del Extractor.

Indexación del Documento
Se realiza a través de un Analizador Léxico y un Algoritmo de Supresión de Palabras de alta frecuencia, por el cual los términos más relevantes son extraídos del texto del documento.

- **Analizador Léxico**
  En esta fase la página HTML de entrada que viene con una serie de contenidos sin importancia es fraccionada en pequeños elementos para lo cual se:
  - •Remueve los HTML tags
  - •Remueve los comandos de lenguaje Script

- **Algoritmo de Supresión de Palabras de alta frecuencia (Stopword)**
  Una vez obtenido en el paso anterior de una serie de elementos que no dependan del lenguaje Web se procede a remover:
  - Palabras de alta frecuencia como "el", "la", "los", etc.
  - Palabras comunes como días, meses y todos los números.Signos Ortográficos (-, /,", etc.).Para cumplir con este objetivo usamos un archivo Thesaurus (ver

Apéndice F) que esta almacenada en una base de datos MYSQL. Se toma cada uno de los elementos del vector documento y se hace la consulta a la base de datos, si el elemento está entonces se descarta sino se almacena en un array.

Tabla 6.1: Estructura de Thesaurus

Campo	Tipo	Clave	Predeterminado	Extra
Espanolid	Int(10) unsigned	Primaria	null	Auto_increment
Palabras	Varchar(30)			

Ponderación de Términos

Permite mantener una estadística simple de los términos del documento. Para ponderar los términos del vector documento en este trabajo se esta considerando el factor de frecuencia de termino(tf) igual a uno, es decir todas las palabras tienen la misma importancia; el factor de frecuencia de documento(df) igual a uno ya que solo usamos un documento y no hago uso del factor de normalización debido a que el proceso se haría muy lento. Por lo que para calcular el peso de un término solo es necesario ver cuantas veces se repite el término en el array de la primera etapa. (Fig. 6.11).

6.3.2 Interfase del Extractor

La Figura 6.10 muestra la interfase del Extractor, dentro del campo de texto debemos indicar la dirección URL de la página Web a ser analizada, la Figura 6.11 nos muestra la respuesta del Extractor a la extracción de la URL: http://www.uni.edu.pe/, la palabra clave encontrada esta a la izquierda y el número de veces que ésta se repite en la página se muestra a la derecha.

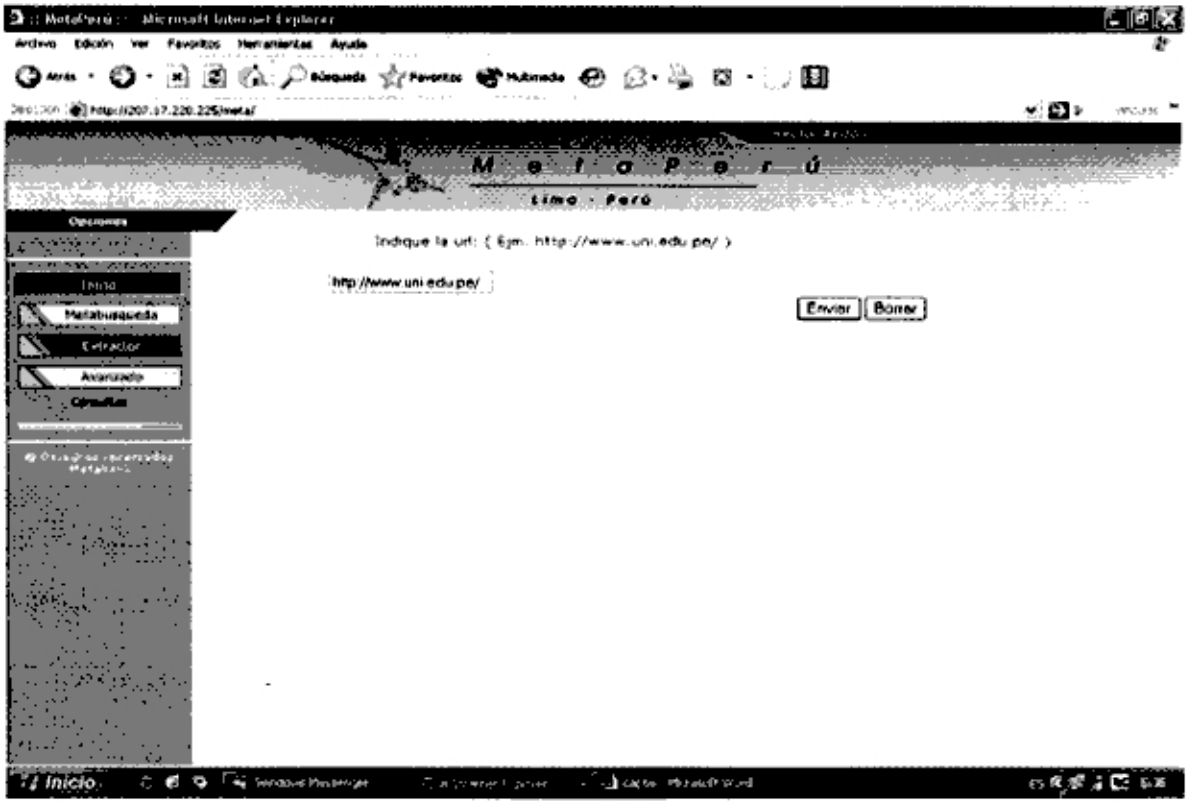

Fig. 6.10: Interfase de consulta para el Extractor.

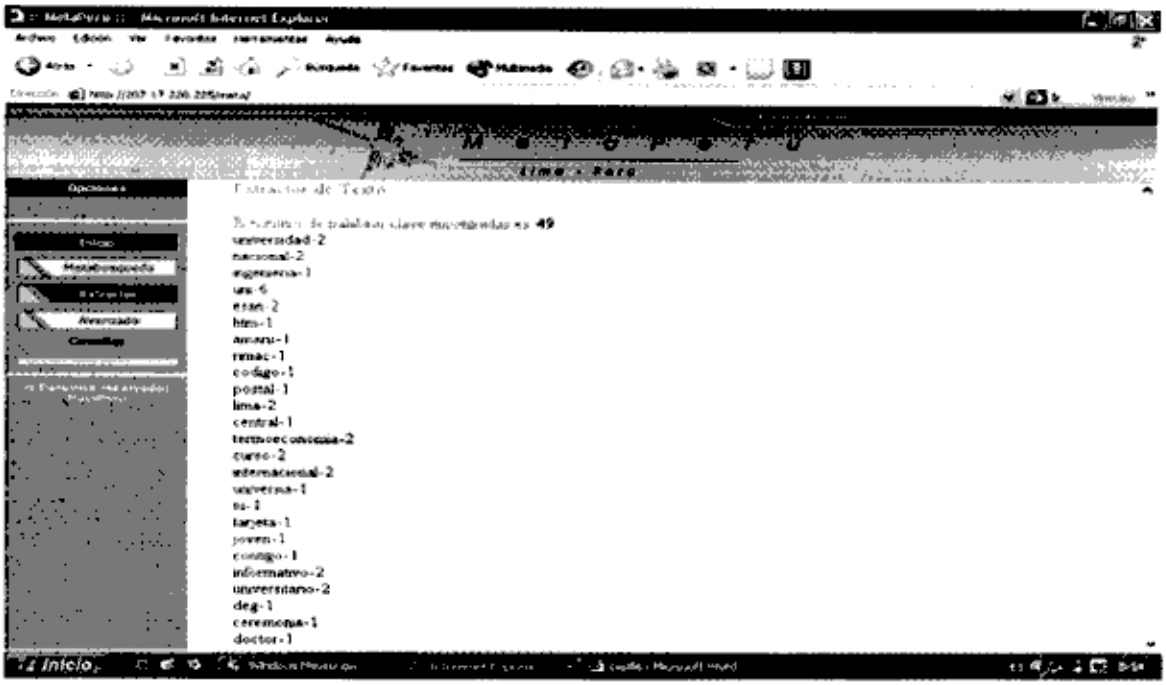

Fig. 6.11: Página de respuesta del Extractor a la URL "http://www.uni.edu.pe/".

6.4 Avanzado

6.4.1 Arquitectura

Esta componente implementa técnicas evolutivas venidas del campo de la Inteligencia Artificial para explorar áreas de interés del usuario, es decir el perfil del usuario y sugerir las mejores URLs que se ajustan a este perfil. Considera un conjunto de agentes que interactúan dos de los cuales son similares a las anteriormente analizadas (el metabuscador y el extractor), tal como se ve en la figura 6.12. A continuación vamos a ver su composición.

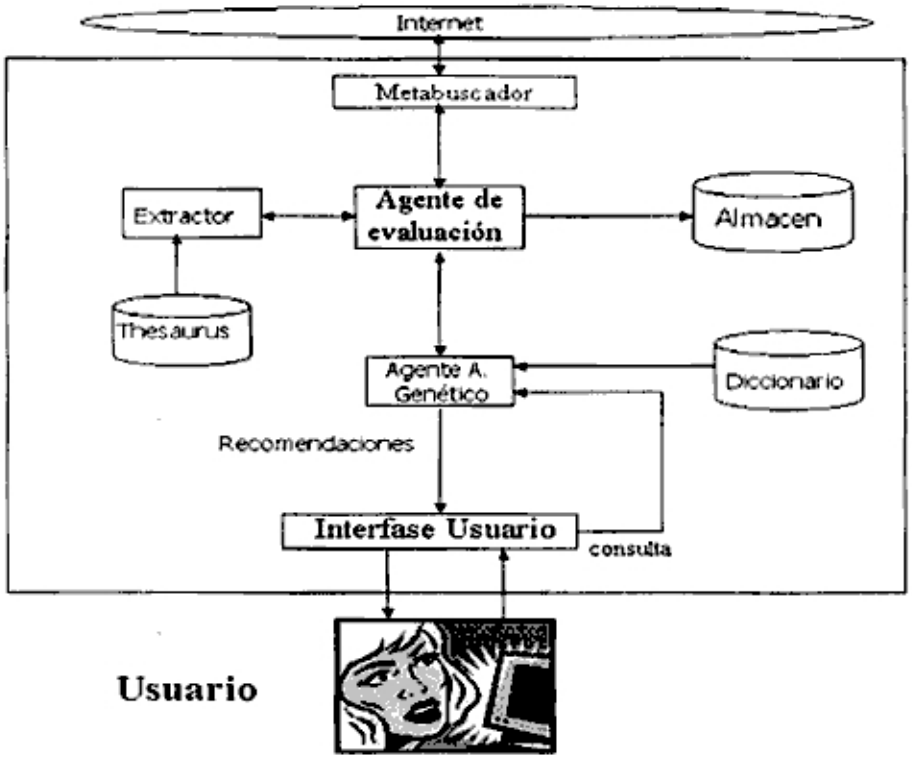

Fig. 6.12: Arquitectura de Avanzado.

Extractor

Cuantifica la importancia de la pagina entregada por el agente de evaluación para su análisis. Recibe un URL como entrada y entrega la frecuencia de palabras del documento a página al agente de evaluación. El archivo Thesaurus es el mismo que se usa en la opción Extractor (Apéndice F).

Metabuscador

El Metabuscador usa un solo motor de búsqueda para la opción Avanzado, se eligió Google por ser el más popular, para la tarea de colección de información pero que puede fácilmente extenderse a los otros dos restantes, el problema es que extender la consulta a mas motores de búsqueda aumentaría el tiempo de respuesta. Esta componente recibe como entrada la consulta del agente de evaluación y entrega los URLs de respuesta en un máximo de siete al agente de evaluación.

Agente de Algoritmos Genéticos

En primer lugar el agente de algoritmos genéticos (Fig. 6.13) realiza la fase de *inicializar* la población, basada en el perfil de usuario. Todas las palabras incluidas en el documento *diccionario* y aquellas que el usuario ingresa a través del formulario de consulta (interfase de usuario) forman el espacio de búsqueda (población).

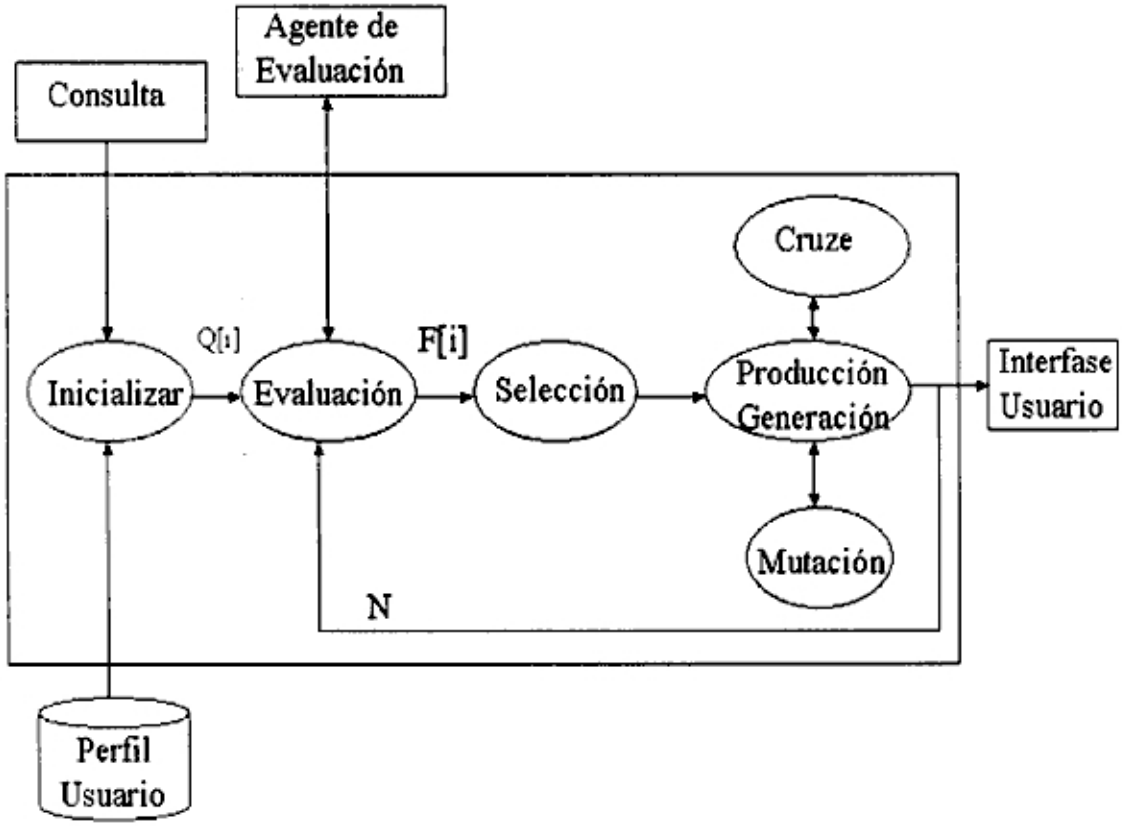

Fig. 6.14: Arquitectura del Agente de Algoritmos Genéticos.

El pseudo código específico usado en la implementación de este agente es el siguiente:

```
{
generacionx= 0;
inicializarOrganismo (P, generacionx)
repetir hasta (generacionx = numgen)
        {
         generacionx = generacionx +1;
         evaluación (P, generacionx);
        guardarURLs(P,generacionx);
        selección de los padres (P, generacionx);
        cruce(P, generacionx);
        mutación (P, generacionx);
        siguienteGeneracion (P, generacionx);
        }
}
```

donde:

generacionx- Generación actual;
numgen – Numero de generaciones para finalizar el algoritmo;
P - Población

Ahora para elegir el tamaño de mi población, es decir el número de palabras de la página de interés que escoja y que va a conformar mi *perfil de usuario*, he tenido que hacer un análisis previo. Voy a hacer un análisis de la frecuencia con que aparecen las primeras 30 palabras para cuatro documentos Web cualesquiera, luego los voy a normalizar en forma individual cada pagina con respecto a la palabra que más se ha repetido y obtengo los resultados mostrados en la figura 6.14, de este grafico se puede observar que las primeras 16 palabras son mas frecuentes en un documento con respecto al resto que mantiene una frecuencia normalizada de aproximadamente 0,2. Esto me permite tomar a este número como limite superior de palabras relevantes. En base a esto he escogido tomar para generar mi primera población solo las diez primeras palabras de un documento que significan casi el 70 % de palabras más relevantes. Las estadísticas obtenidas se muestran en el apéndice D.

El algoritmo crea consultas complejas (individuos) en un numero de cuatro, usando palabras del diccionario y operadores lógicos (Y, O), las *palabras claves* que más se repiten tienen mayor probabilidad de ser escogidas para formar las consultas, las cuales son enviadas al *agente de evaluación*. Realizada la fase de evaluación se pasa a la fase de *selección* donde se elige al azar por el método de la ruleta, dos individuos. Ya elegidos los individuos pasamos a la fase de producción de la siguiente generación para los cuales hacemos uso de los operadores de cruce y mutación. Se verifica el número de generaciones permitidas, si la actual es menor al permitido, se regresa y se hace un nuevo proceso de evaluación, selección y producción, esto se repite hasta que el número de generaciones sea igual al permitido. Una vez alcanzado este punto se buscan las siete (puede variar) URLs con mas alto valor de similitud y se recomiendan al usuario (Fig. 6.17).

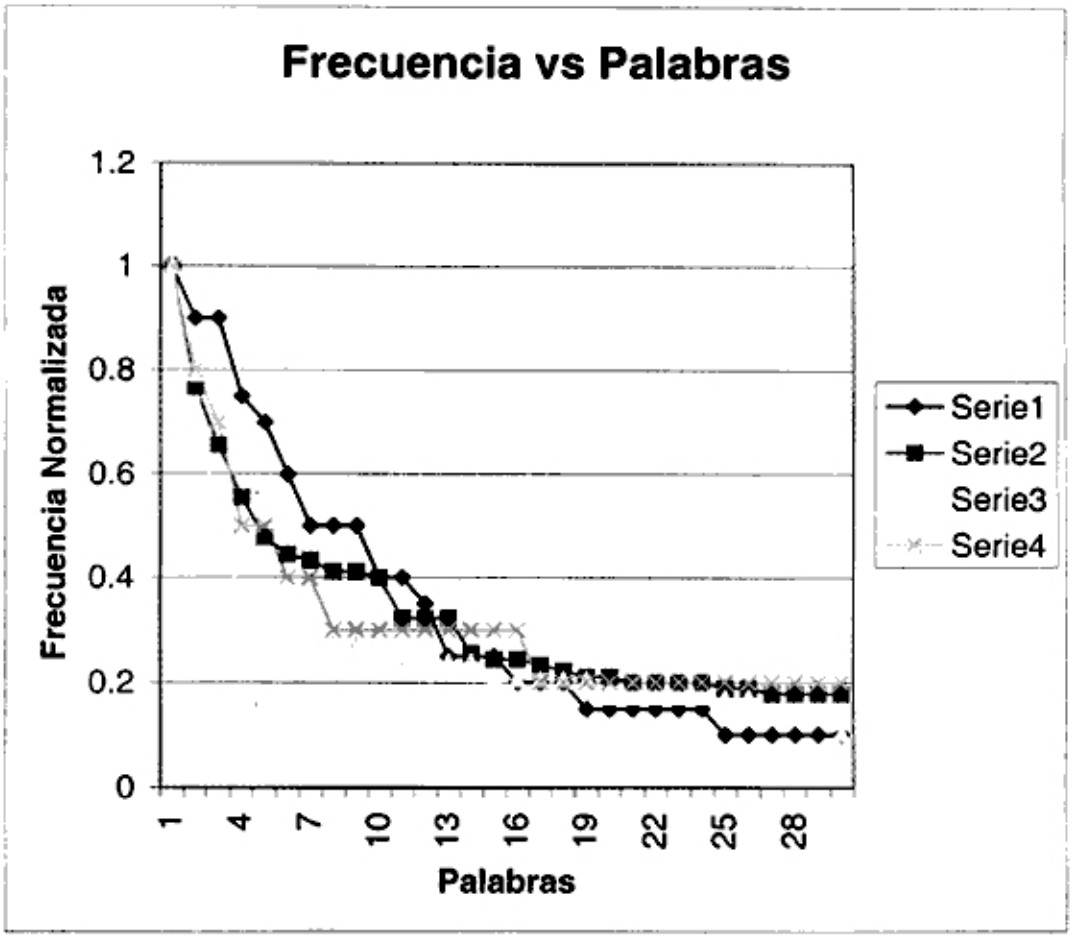

Fig. 6.14: Análisis de frecuencia de las primeras 30 palabras que más se repiten en las páginas Web.

Agente de Evaluación

La primera fase de este agente (Fig. 6.15) es la de *consulta* en el cual se recibe la "consulta" del agente de Algoritmos Genéticos y se la envía al metabuscador. Una vez realizada la consulta en la fase de selección de URL se cogen los tres primeros URLs de respuesta del metabuscador, los cuales se envían al Extractor de uno en uno. Una función similitud se usa para evaluar la similitud de cada documento que envía el extractor con respecto al perfil del usuario. Una vez analizados los URLs se suman sus similitudes para dar un valor de similitud total F[i] a la consulta (individuo) y se pasan al agente de Algoritmos Genéticos.

Aquí en esta fase los URLs activos se guardan en el archivo almacén (Apéndice C) y aquellos cuyos enlaces están rotos se descartan.

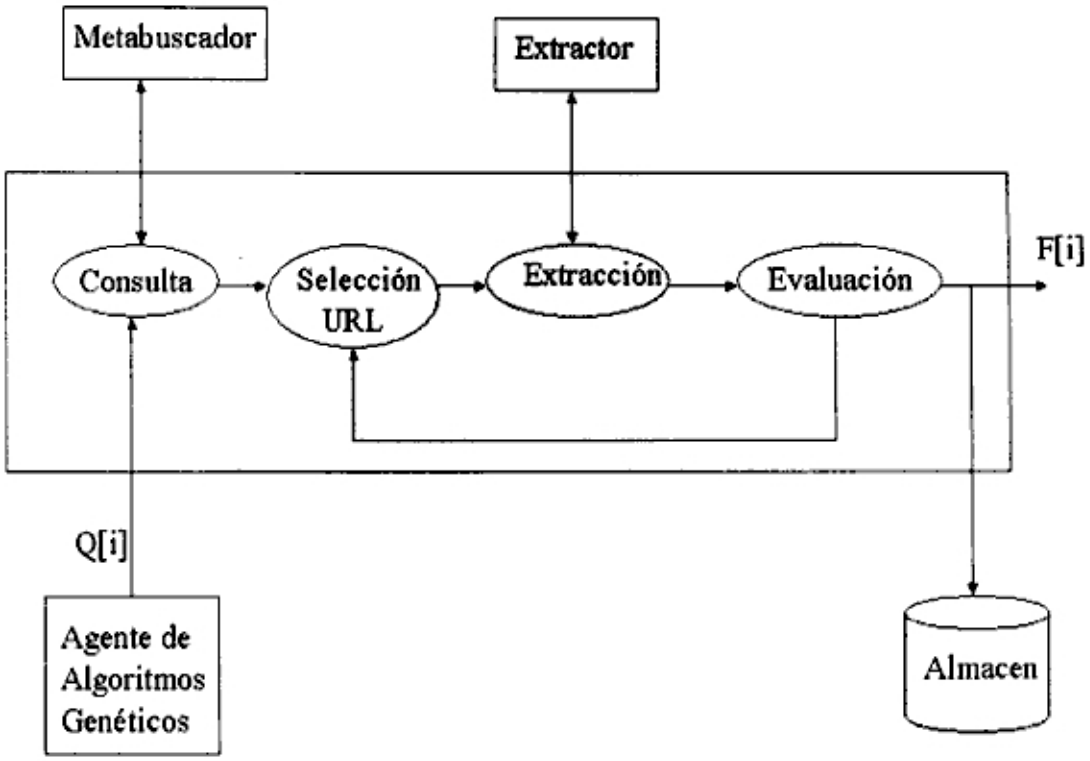

Fig. 6.15: Arquitectura del Agente de Evaluación.

6.4.2 Interfase de Avanzado

Los usuarios proveen un conjunto de palabras que describen apropiadamente su interés en un tema especifico, esto se hace a través de un formulario que permite ingresar cuatro palabras clave en la *interfaz de usuario* (Fig. 6.16) que se agregan a las ya existentes en el diccionario (Apéndice B).

La interfaz nos da la opción también de indicar el número de generaciones que el algoritmo va a iterar antes de arrojar una respuesta al usuario.

Esta componente después de ejecutar la búsqueda basada en el perfil de usuario ingresado trata de hallar las páginas que mejor se ajusten al perfil tal como se observa en la figura 6.17.

A continuación vamos a mostrar y analizar los resultados obtenidos por nuestro sistema en la opción Avanzado. Para esto vamos dividir nuestro análisis en dos partes, la primera *sin realimentar* el sistema y la segunda *realimentando* el sistema.

En esta primera parte, vamos a realizar una serie de medidas experimentales en dos tipos de contexto, el primero en el tema de *Física I* cuyos resultados se aprecian en la tabla 6.2 y la figura 6.18, de allí podemos ver como a medida que el número de generaciones se incrementa la similitud crece lo que implica que las páginas web encontradas se ajustan mejor al perfil del usuario.

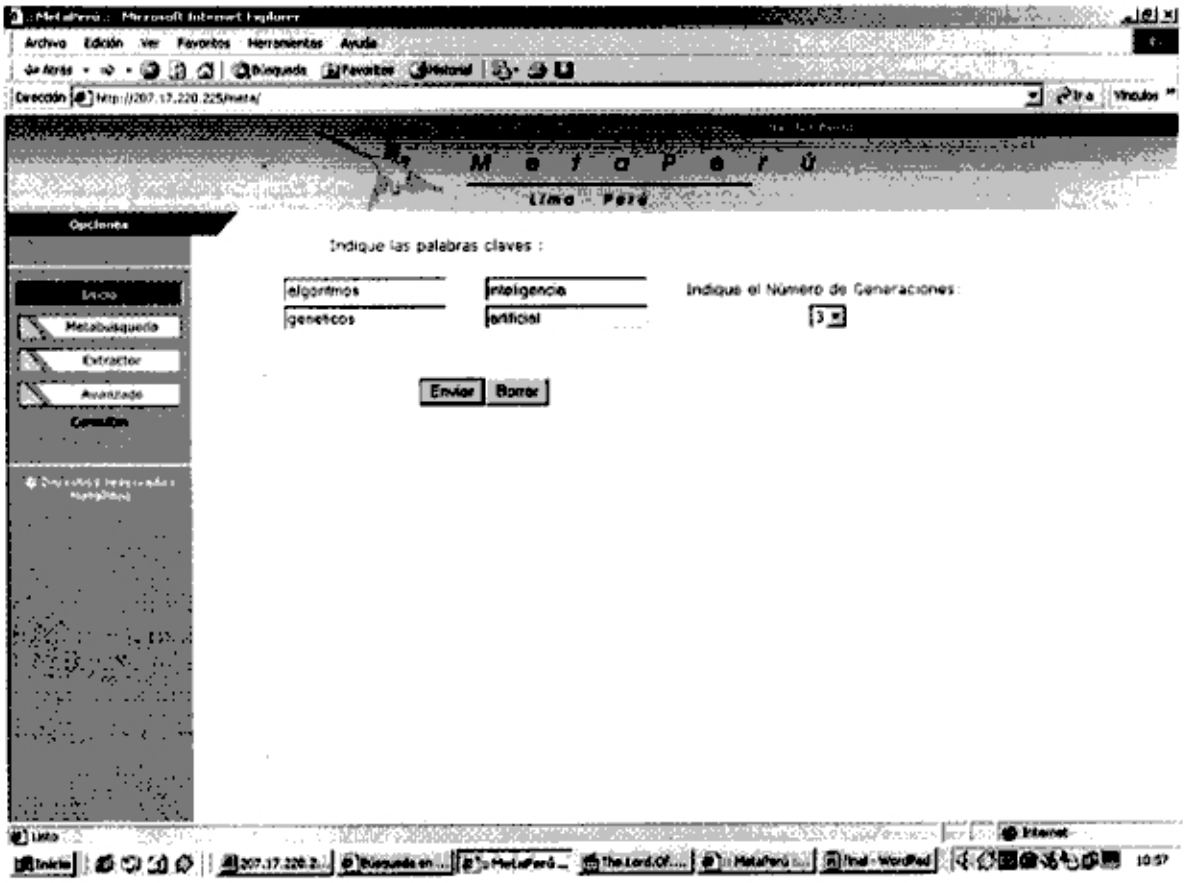

Fig. 6.16: Interfase de consulta para Avanzado.

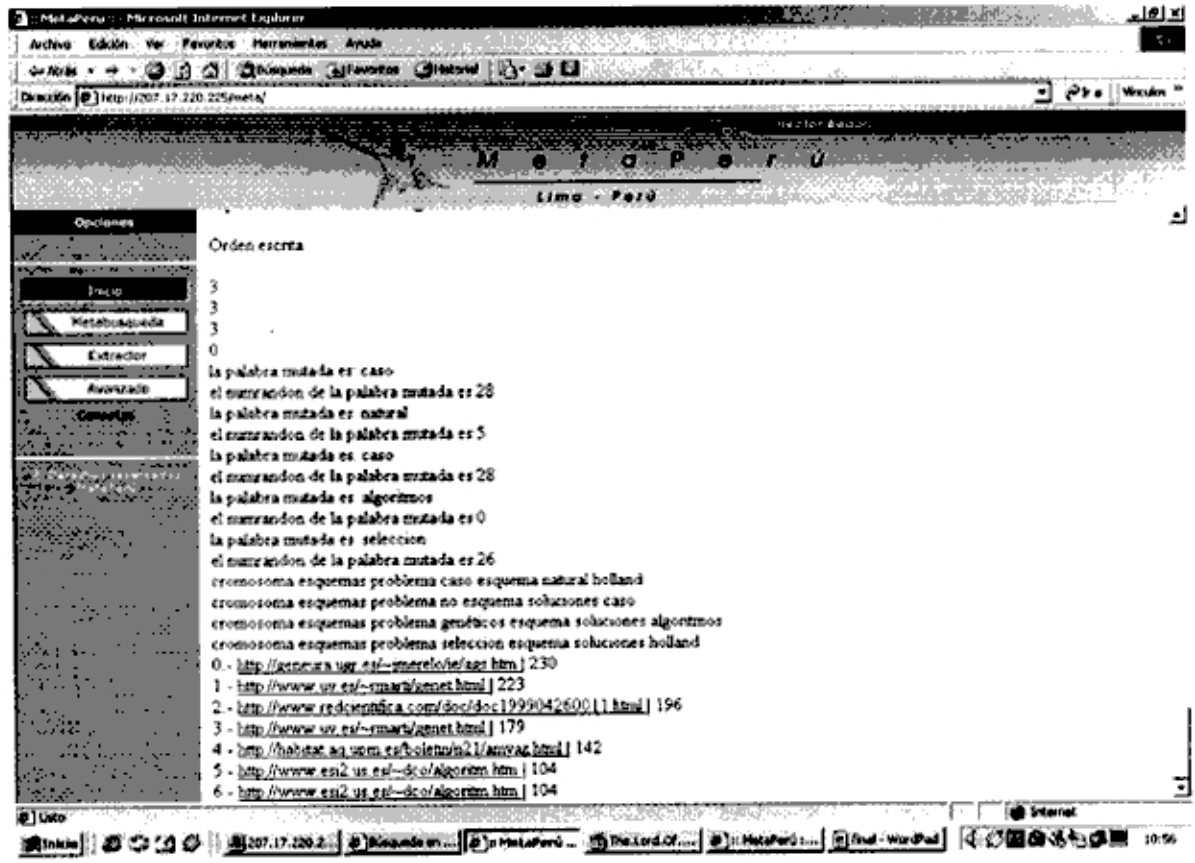

Fig. 6.17. Página de respuesta para Avanzado.

Tabla 6.2: Tema de búsqueda: *"Física I"*

Consulta:	fuerza, aceleración, masa y fricción.

	Generaciones				
No Organismos	1	2	3	4	5
1	68	94	94	56	250
2	101	161	286	285	219
3	0	285	170	94	380
4	323	177	304	286	268
Similitud Promedio	492	717	854	721	1117

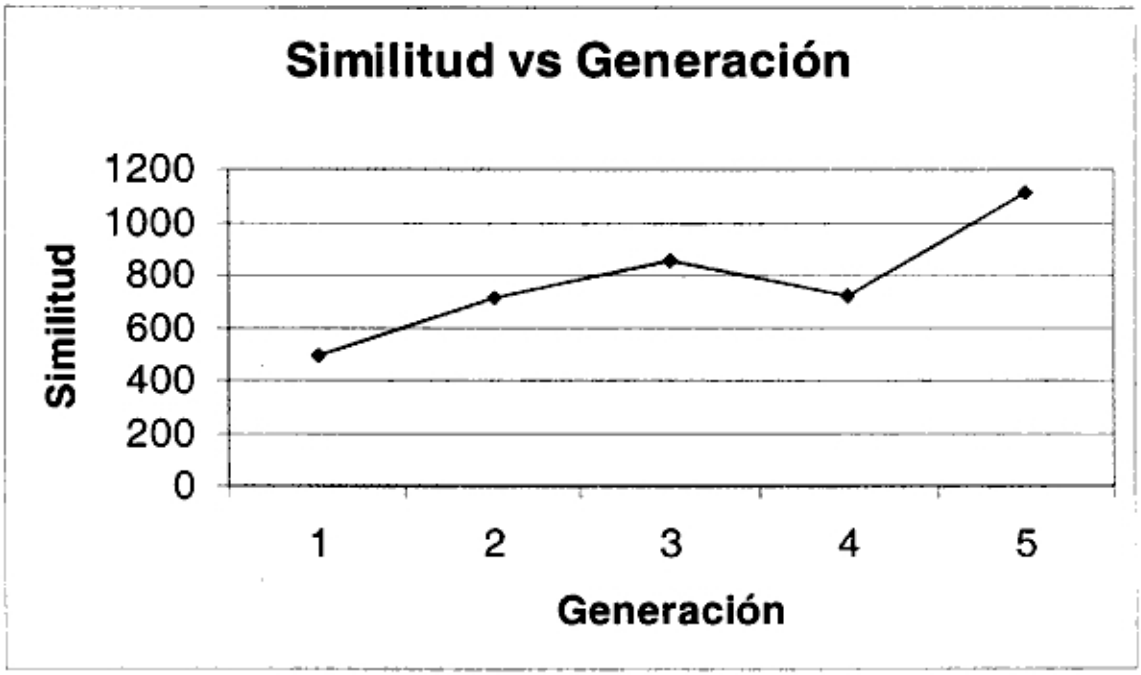

Fig. 6.18: Evolución de la Similitud través de las generaciones para la consulta: fuerza, aceleración, masa y fricción.

Y el segundo en el tema de *Algoritmos Genéticos*, aquí he variado las palabras de ingreso través de la interfase pero manteniendo las mismas palabras en el diccionario, y se ha obtenido los resultados que se muestran en las tablas 6.3 y 6.4 y las figuras 6.19 y 6.20 respectivamente.

De aquí se puede ver que el comportamiento es similar al primer tema es decir aumenta con el incremento del numero de generaciones pero con distinto comportamiento lo que sugiere que el sistema es muy dependiente de las palabras de ingreso del usuario, es decir un ingreso de palabras que no reflejen bien el interés del usuario nos lleva a un comportamiento no muy óptimo del sistema.

Tabla 6.3: Algoritmos Genéticos

Consulta:	algoritmos, geneticos, evolucion y natural

	Generaciones				
No Organismos	1	2	3	4	5
1	195	363	561	363	363
2	316	12	0	363	0
3	371	0	0	444	363
4	143	363	363	363	363
Similitud Promedio	1025	738	924	1533	1089

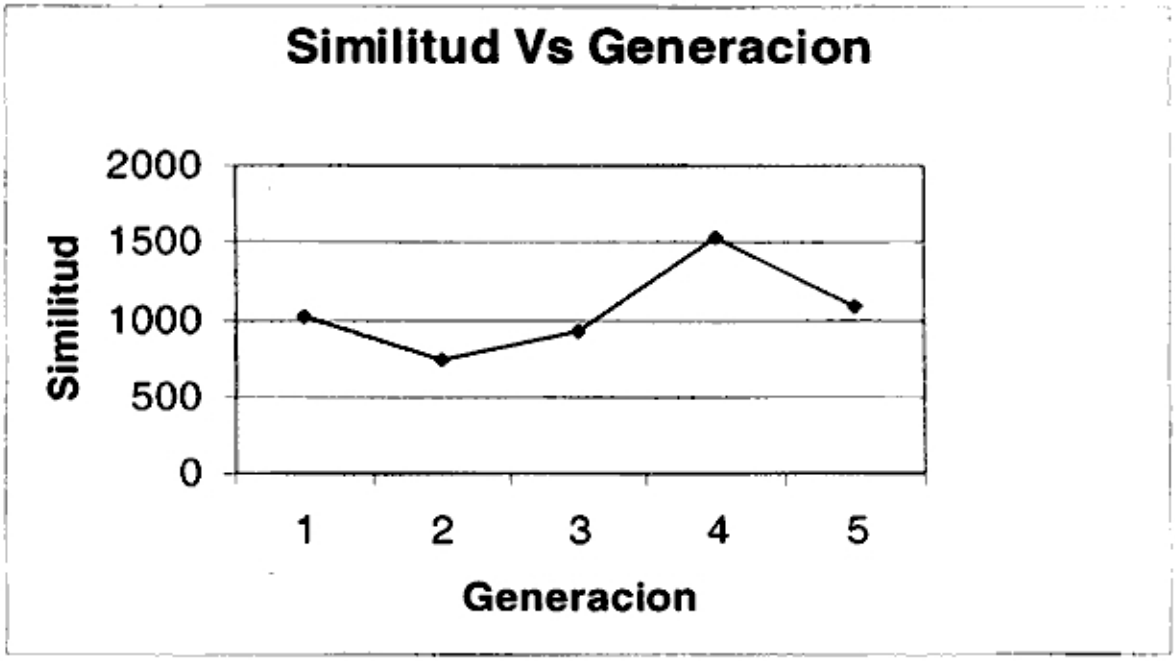

Fig. 6.19: Evolución de la Similitud través de las generaciones para la consulta: algoritmos, geneticos, evolucion y natural

Tabla 6.4: Algoritmos Genéticos

Consulta:	algoritmos, holland, geneticos y proceso.

	Generaciones					
No Organismos	1	2	3	4	5	6
1	0	227	227	227	0	280
2	233	287	0	0	273	280
3	0	0	0	327	280	280
4	0	0	227	273	236	0
Similitud Promedio	233	514	454	827	789	840

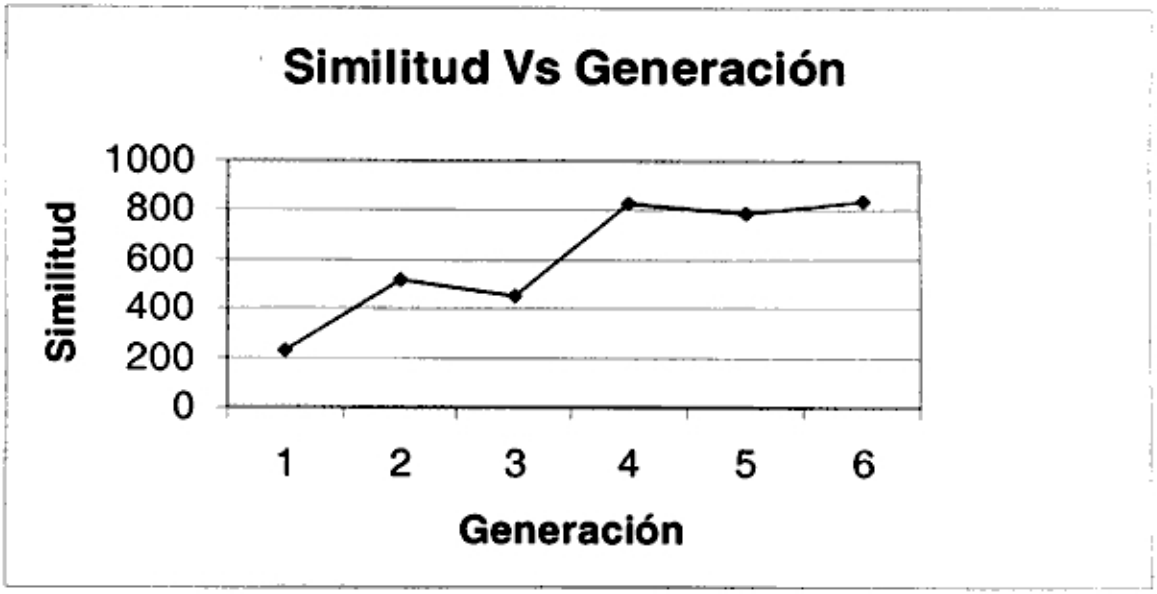

Fig. 6.20: Evolución de la Similitud través de las generaciones para la consulta: algoritmos, holland, geneticos y proceso.

En esta segunda parte voy a hacer un análisis de cómo evoluciona el sistema a medida que este se realimenta por paginas Web seleccionadas por el usuario y como se va diferenciando de los resultados obtenidos por el buscador Google. En nuestro sistema tanto la primera pagina que escoja el usuario como población inicial como las paginas que se usen luego para realimentar el sistema son muy importantes, ya que tienen que ser paginas que reflejen el interés del usuario, porque sobre la base de estas paginas el sistema arrojara direcciones URL de paginas Web que se le asemejen.

La consulta que se va a hacer tanto a mi sistema como al buscador Google es: ***"algoritmos geneticos"***, los resultados y la comparación se muestran en la figura 6.21 y en el apéndice E. Las primera página web usada como generadora de la primera población inicial es:
http://www.uv.es/~rmarti/genet.html

El análisis se ha hecho sobre las primeras 7 URLs mostradas por mi sistema y el buscador Google.

La figura 6.21 muestra como en el sistema la tasa de similitud va aumentando a medida que se va realimentando el sistema. La tasa de similitud la defino como la similitud total dividida entre el número de palabras que conforman la población inicial en cada realimentación. También se puede observar como en el caso de Google esta tasa disminuye lo que sugiere que los resultados de Google se van alejando del perfil dinámico del usuario en cada realimentación. Para el cálculo de similitud se ha tomado en consideración el hecho de que la similitud aumenta cuando una palabra es repetida en la población inicial por lo que se ha recalculado en cada nueva realimentación la similitud arrojada por Google.

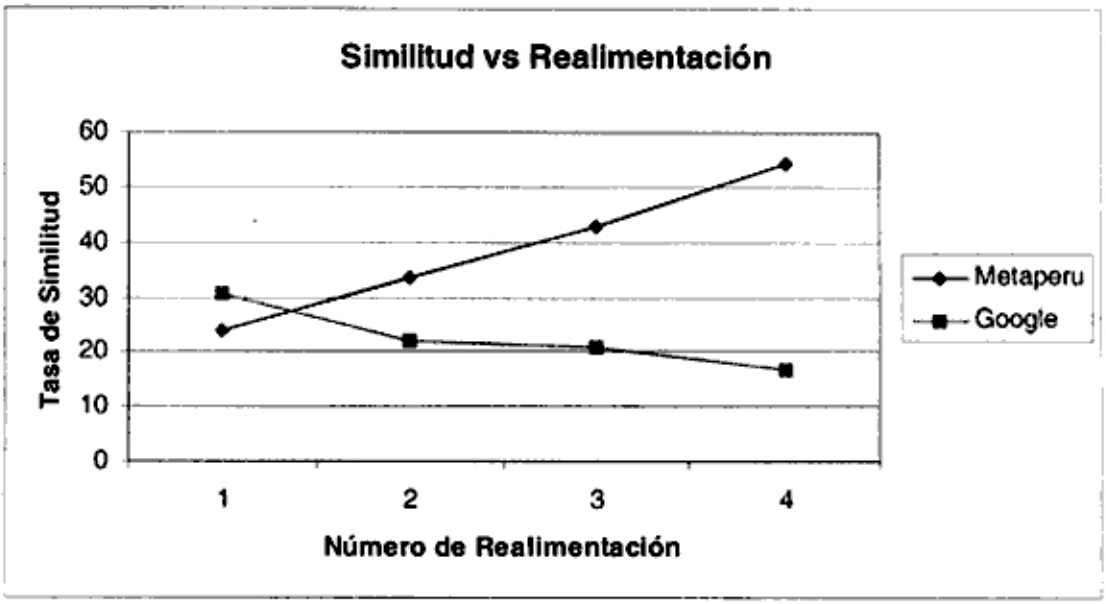

Fig. 6.21: Evolución de la Tasa de Similitud través de cada realimentación para la consulta:"algoritmos geneticos" versus los resultados obtenidos por Google.

Finalmente vamos a evaluar los tiempos de respuesta del sistema por URL mostrado (TR/UM) a medida que se va incrementando el numero de generaciones, aquí el TR/UM nos da una medida efectiva de cuanta tarda el sistema para mostrar al usuario una URL, el tiempo se midió con el programa restando los valores al inicio y al final del procesamiento, lo que se verifico con el tiempo medido por el browser Netscape 7.0.

Se puede observar(Tabla 6.5 y Fig. 6.22) que el TR/UM se incrementa ligeramente con el incremento del numero de generaciones y esto es lógico ya que el número de iteraciones que realiza el programa por generación se incrementa, pero en promedio podemos considerar un tiempo de 28 segundos por URL mostrado. También se puede observar que el TR/UM es dependiente del ancho de banda ya que si observamos la figura 6.23 podemos ver que el incremento en el TR/UM corresponde a una caída en el Ancho de Banda. La tasa promedio a las que se realizaron las pruebas fue de 66 Kbps, estas medidas se hicieron usando la herramienta WebSpeed [36].

Tabla 6.5: Tiempo de Respuesta

Generación	Tiempo de Respuesta (TR) (s)	URLs Mostrado (UM)	TR/UM
1	34.3	2	17.15
2	54.78	5	10.96
3	229.17	4	57.29
4	112.61	4	30.65
5	135.41	6	22.57

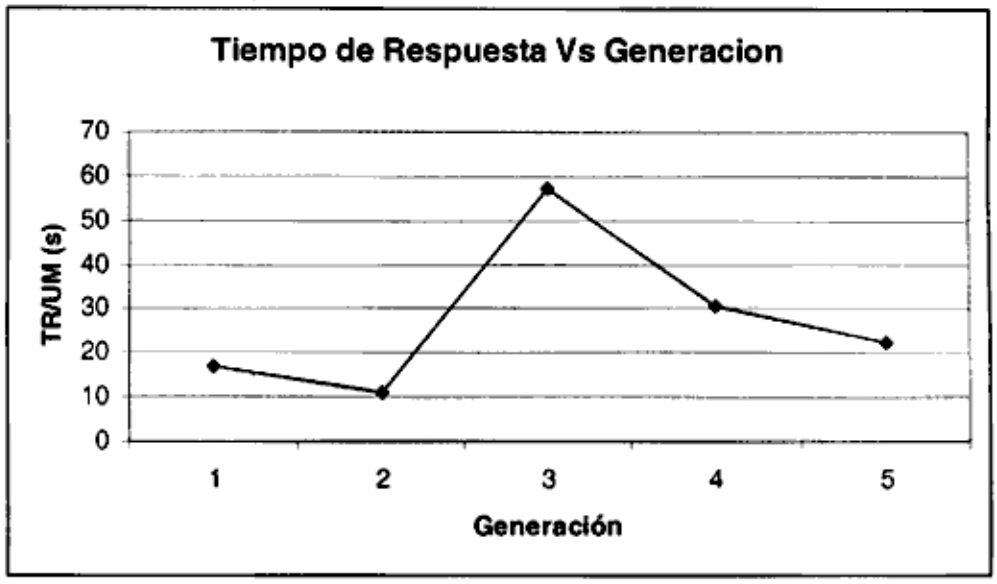

Fig. 6.22: Tiempo de Respuesta por URL mostrado versus el Número de Generaciones.

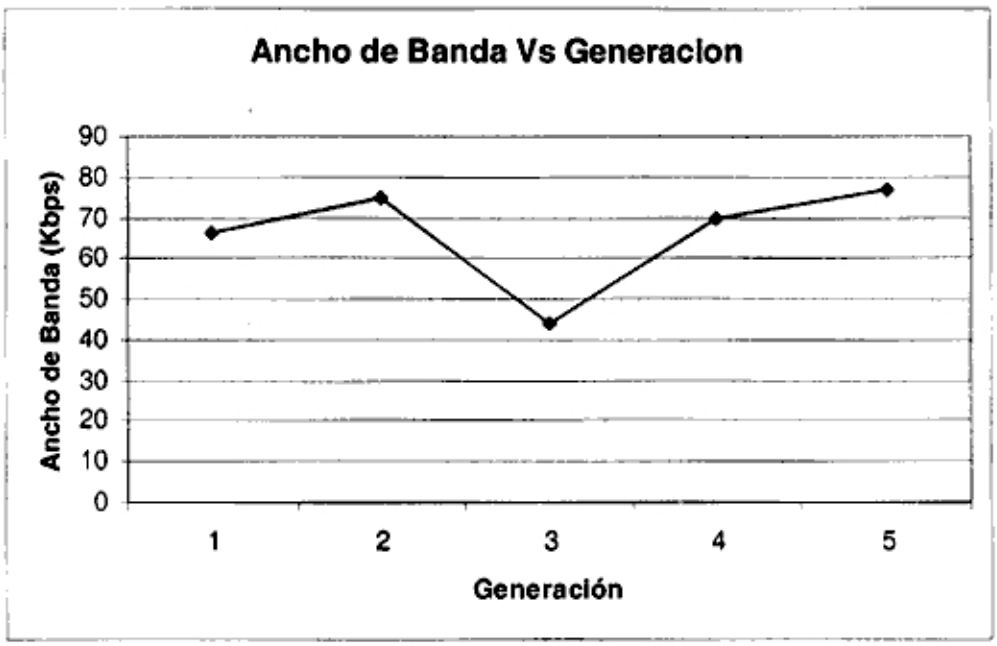

Fig. 6.23: Ancho de Banda usado versus el Número de Generaciones.

Capítulo 7

Conclusiones y Sugerencias

7.1 Conclusiones

Esta tesis ha alcanzado los siguientes objetivos:

1) Diseño e implementación de una nueva herramienta de búsqueda integrada, el cual permite optimizar la búsqueda de información en Internet sobre tópicos específicos de una manera personalizada.

2) Mostrar como funciona un Metabuscador, el cual bajo una sola interfase enlaza a tres de los más importantes buscadores: Google, Altavista y Yahoo, con el consiguiente ahorro de tiempo y un aumento en la diversidad de la información, al no estar restringida a un solo motor de búsqueda.

3) Diseño e implementación de un Extractor de Texto basado en el Modelo de Espacio Vectorial, optimizado para el idioma español el cual haciendo uso de algoritmos de indexación y ponderación de términos, permite extraer palabras relevantes de cualquier página HTML, así como el número de veces que estas se repiten en la página, ordenadas en forma decreciente. Gracias a esto se ha podido hallar cierto patrón en las páginas Web en las que aproximadamente las primeras 16 palabras que más se repiten en las páginas Web son las más frecuentes (Fig. 6.14).

4) Diseño e implementación de un prototipo de búsqueda inteligente de nivel 1(ver Pág.23), que basado en algoritmos genéticos obtiene un conjunto de URLs activos que se ajustan al perfil del usuario. Los logros obtenidos lo podemos dividir en dos partes:

    a. En la primera parte bajo un esquema sin realimentación, se ha podido observar que la similitud aumenta en los dos tipos de contextos analizados, hasta en un 400% (Fig. 6.20) con el incremento del número de generaciones, tendiendo a un valor máximo, lo que demuestra la correcta operación del algoritmo, ya que a medida que el numero de generaciones se incrementa la similitud obtenida se ajusta más al perfil del usuario.

    b. Se ha podido observar también que bajo un esquema de realimentación el sistema evoluciona en función de las preferencias del usuario, ya que se ajusta mejor al contexto del usuario (Fig. 6.21), lo que lo diferencia de muchos sistemas de búsqueda en los que cada nueva consulta no tiene nada que ver con la búsqueda pasada.

5) Constituirse como una herramienta pionera en el Perú en el proceso de Gestión de Información.

Entre los problemas encontrados se pueden mencionar a los siguientes:

- El factor de mantenimiento del software, que esta relacionado con los motores de búsqueda. Aunque estos están muy bien mantenidos el problema es que regularmente cada 3 o 4 meses cambian los formatos de entrada y salida, lo que nos obliga a estar pendientes de estos cambios para la actualización del código.

- Tiempo de respuesta, nuestro prototipo es muy dependiente del ancho de banda de la red local, cada búsqueda en el avanzado puede durar entre 17 a 58 segundos por URL mostrado (Fig. 6.22 y 6.23).

- No existe una normalización en el proceso de ponderación del documento vectorial por lo que los documentos más grandes tienen mayor probabilidad de ser escogidos.

- El usuario debe tener cierta experiencia con respecto al contexto de la búsqueda que esta realizando.

7.2 Sugerencias para Trabajos Futuros

Se pueden sugerir los siguientes puntos para mejorar la performance del sistema:

1. Aumentar el número de motores de búsqueda enlazados en el componente de Metabusqueda.

2. Integrar un algoritmo dentro del componente Extractor, para reducir la redundancia de palabras claves para el idioma español así como un algoritmo de supresión de sinónimos.

3. Aumentar el número de contextos en el que se realiza el proceso de búsqueda.

4. Migrar el prototipo de solo una aplicación enteramente en servidor, a una aplicación mixta en la que el Metabuscador pueda seguir corriendo en servidor pero el modulo de búsqueda Avanzada se corra en una maquina cliente, esta componente se podría trabajar en Java.

5. Aplicar un factor de normalización al proceso de ponderación de términos en el módulo Extractor de Texto.

6. Debido a que el prototipo se encuentra sobre una plataforma Linux se puede implementar un cluster de computadoras en un entorno Mosix para mejorar el rendimiento del servidor y por ende el tiempo de respuesta del proceso de búsqueda.

REFERENCIAS

[1] S. Lawrence and C. Lee Giles. Accesibility of information on the web.Nature, 400, July 1999, p.p 107-109.

[2] Google: http://www.google.com

[3] Altavista: http://www.altavista.com

[4] Yahoo: http://www.yahoo.com

[5] Internet Software Consortium: http://www.isc.org/

[6] NIC-Mexico: http://www.nic.mx/nic/plsql/Estadisticas.a9902

[7] OSIPTEL, Organismo Supervisor de Inversión Privada en Telecomunicaciones: http://www.osiptel.gob.pe

[8] Webseek: http://www.webseek.com/index.html

[9] WebCrawler: http://webcrawler.com

[10] Weiyi Meng, Clement Yu and King-Lup Liu, "Building Efficient and Effective Metasearch Engines", http://citeseer.nj.nec.com/376145.html.

[11] Wen-Chen Hu, 'World Wide Web Search Technologies', capitulo del libro, Shi Nansi (Ed.), 'Architectural Issues of Web-Enabled Electronic Business', Idea Group Publishing.

[12] Sergey Brin and Lawrence Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine", Computer Networks and ISDN Systems, 30:107-117,1998.

[13] L. Gravano, K. Chang, H. Garcia-Molina, C. Lagoze, A. Paepcke, "STARTS: Stanford protocol proposal for internet retrieval and search", Proceedings of the ACM SIGMOD International Conference on Management of Data, 1997.

[14] Lycos: http://www.lycos.com

[15] J. Xu y B. Croft, "Cluster-based Languaje Models for Distributed Retrieval", ACM SIGIR, 1999.

[16] Atsushi Sugiura y Oren Etzioni, "Query routing for Web search engines: Architecture and experiments", Proceedings of the 9th International World Wide Web Conference, Amsterdam, Netherlands, May 2000.

[17] R. Svidzinska, "A World Wide Web Meta Search Engine Using an Automatic Query Routing Algorithm", master's thesis, http://citeseer.nj.nec.com/svidzinska01world.html

[18] NEC Research Institute CiteSeer, http://citeseer.nj.nec.com

[19] S. Lawrence, "Context in Web Search" , http://citeseer.nj.nec.com/lawrence00context.html

[20] E. Selberg and O. Etzioni, "The MetaCrawler Architecture for Resource Aggregation on the Web", http://citeseer.nj.nec.com/selberg97metacrawler.html.

[21] http://activist.gpl.ibm.com:81/WhitePaper/ptc2.htm.

[22] P. Maes, "Agents that Reduce Work and Information Overload", Communications of the ACM, 37(7) (July 1994), 31-40.

[23] S. Russell, P. Norvig, Inteligencia Artificial, Un enfoque moderno, Prentice Hall, p. 33, 1996.

[24] D. Smith, A. Cypher and J. Spohrer, "KIDSIM: Programming agents without a programming languaje", Communications of the ACM, 37(7):55-67, July 1994.

[25] http://dollar.biz.uiowa.edu/~fil/Class/6k260/PDF/lec9.pdf

[26] Melanie Mitchell,An Introduction to Genetic Algorithms, First MIT Press paperback edition, p.p. 21-23, 1998.

[27] Carpineto, C. and Romano, G. *Effective reformulation of Boolean queries with concept lattices.* Datalogiske Skrifter, Issue.78, p. 83-94, Univ. Roskilde, 1998.

[28] Bordogna, G.; Carrara, P. and Pasi, G. ***Extending Boolean information retrieval: A fuzzy model based on linguistic variables.*** IEEE International Conference on Fuzzy Systems, p. 769-776, IEEE, 1992.

[29] Srinivasan, Padmini. *Query expansion and MEDLINE.* Information Processing and Management, 32 (4) , p. 431-444, 1996.

[30] Gallant, S. I. *A practical approach for representing context and for performing word sense disambiguation using neural networks.* Neural Computation, 3 (3), p. 293-309, 1991.

[31] Salton, Gerard. *Introduction to Modern Information Retrieval.* McGraw-Hill, 1983.

[32] Van Rijsbergen, C. J. *Information retrieval.* Butterworths, 1979.

[33] HyperText Transfer Protocol (ver 1.0): http://www.ietf.org/rfc/rfc2616.txt

[34] Zacharis Z. Nick, Panayiotopoulus Themis,*Web Search Using a Genetic Algorithm,* IEEE Internet Computing, p. 19-26, March-April 2001.

[35] Holland J.H., *Adaptation in natural and artificial system,* Ann Arbor, The University of Michigan Press, 1975.

[36] WebSpeed: http://www.numion.com.