Universidad de la Habana

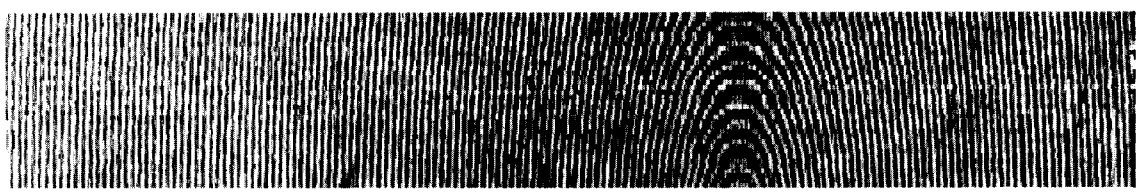
Facultad de Matemática y Computación

MATCOM

Facultad de Matemática - Computación

Spider UCI

Trabajo de Diploma



Autor: Jhonlier Suárez Molina

Tutor: Ing. William Azcuy Morales

UCI

Universidad de las Ciencias Informáticas

Ciudad de la Habana,

Junio 2004

Existe la jerarquía de mi afecto:

*a **Gessliam**, **Ana María** y **Gaspar**.*

*a **Mami Nena**, porque su nombre real no le gustaría leerlo acá.*

Mis tíos, tías, primos y primas por conformar esa nube confortable llamada familia de la que es tan difícil bajarse.

***Jaime** y **el Yocet**, mis hermanitos.*

*Y a la doble **A** que ha marcado mi vida como una letra escarlata.*

Agradecimientos

No quiero que se quede nadie, así que será largo:

En la UO: Mis profesores, en especial a Aurora, Alexis, Mirtha (quién se lo iba a decir), Aguilera, Adriano (que ya no está), Juanita y Hebert. Solo quiero excluir expresamente a Magalis. A mis tres grupos, pues cada uno me enseño algo distinto: el primero la unidad y el dolor; el segundo, la soledad y el tercero, la fuerza y la alegría. Todos, excepto contadas excepciones, que ellos mismos deben saber, quedan en la memoria como único puedo archivar a la gente: cuando sonríen. Ahí también está enmarcado una serie de amigos y amigas y fastidiosos compañeros de cuarto. Un mensaje telepático para ellos: la verdad está allá afuera.

En Santiago: La fauna, la AHS, Carlos Torres, David y Whitman por las buenas charlas. Otto, lo tuyo era nada más el corazón, compadre. A Ileana, un caso especial entre las mujeres maravillosas de mi vida.

En Holguín: A Luis Yussef, Yenny, Carlos Alberto, Garayalde, Víctor, Yamili, Puppy, Magdeline, Yoe, Osean y su primo.

En el Cayo: A Olguita, Liuba & Cía. A Héctor y Alexis Legrá. A Betty, Santa y Editha. A las playas y a la gente.

La gente del Onelio que conforman toda una categoría.

Los cuartos: el 207-3D y el 1030 de la UCI.

En la UCI: Jorge E. Hurtado, Leonardo, Tomás, Milagros, Rislaidy, Chirino y Tony.

Mis amigos: Rafael Sariol, Trujillo, Gizeh, Jessie, Pablo, Mildre, Laritza, Meisbel, Yaniela, Haydee, Irela, la Nani más gorda y a los otros, porque es verdad que la verdadera memoria está en el corazón.

Índice

CAPÍTULO II CARACTERÍSTICAS, ANÁLISIS Y DISEÑO DEL SISTEMA 26

Resumen

El presente trabajo de diploma expone detalles de la elaboración de un robot recuperador de información para la Intranet de la Universidad de las Ciencias Informáticas (UCI), que no contaba con este servicio, con vistas de una futura extensión a convertirse en el *spider* de CubaSearch, el motor de búsqueda de CubaSí. Quedan expuestos los pormenores del diseño e implementación: modelos, algoritmos, diseño, clases y herramientas empleadas, etc. Sin excluir un acercamiento teórico a los métodos de Recuperación de Información vigentes y realizar una panorámica del estado del arte de la materia en el mundo.

Abstract

Current document exposes the design and implementation details of forthcoming Informatics Sciences University of Cuba's (UCI in Spanish) spider, due to that center, although the rising amount of information published in it Intranet, does not have one yet. The future application is supposed to become in the CUBASEARCH's spider, the autonomic Information Retrieval System hosted by CubaSí. Design aspects such as: models, algorithms, class design and used tools are shown in these pages. It also includes an theoretical approach to the Information Retrieval models and an analysis of the actuality of the theme in the world.

Introducción

El acelerado crecimiento del monto de información publicada en Internet pone a los usuarios ante la disyuntiva de no saber dónde encontrar con exactitud lo que buscan. Máxime, cuando de una misma temática pueden aparecer miles de archivos dispersos a todo lo largo de la red de redes, en puntos tan geográficamente distantes como Europa y Antártida y a veces, la información buscada forma parte de un documento que mayoritariamente trata de asuntos que no tienen que ver del todo con el objeto de nuestra consulta. ¿Cómo puede entonces encontrarse lo deseado o al menos, un conjunto dónde, entre otros, esté lo que necesitamos?

Tal es el trabajo de los buscadores (motores de búsqueda) y directorios temáticos, los cuales deben mantener un repositorio de direcciones de acceso a los documentos indexados según algún criterio de clasificación, por lo general, por categorías de conocimiento (arte, deportes, computadoras) o desglosados según los términos que aparezcan dentro de ellos. Ambos son sistemas de recuperación de información que entre cuyos propósitos se encuentra, mediante diversas técnicas, poner mecanismos de acceso a información dispersa de cualquier tipo: textos, imágenes, categorías, sonidos.

Los mismos son cada vez más reclamados como punto de partida de la mayoría de las navegaciones que realiza un usuario común. De este modo, cuando no se está claro, al menos se tiene una opción más estricta que escribir una dirección al azar y esperar que el producto sea el esperado.

Así, debido al volumen de información que se ha ido acumulando en la Intranet de la Universidad de las Ciencias Informáticas (UCI) y por la importancia de la misma para todo el personal, venía necesitándose del servicio de búsqueda y por tanto la creación de una Directorio de los documentos publicados, pues no existía ninguno. Como dicho volumen es además muy dinámico, es necesario que dicha herramienta actualice regularmente el repositorio. Por ello se hace necesaria la utilización de un motor de búsqueda, en especial de un *spider* para ese trabajo.

Por añadidura, Cuba, pese a tener una cantidad en crecimiento de información dispuesta en Internet, tampoco cuenta con un servicio de este tipo que a los usuarios nacionales y

foráneos les facilite la navegación y búsqueda de la información solicitada en el ámbito nacional. De modo que en una segunda entrega, pudieran extenderse los resultados y experiencias acá obtenidos a un servicio que extienda la Internet cubana para todo el mundo desde las páginas de CubaSí.

Esto sin mencionar que los buscadores comerciales que aparecen en Internet, sin duda, los más eficientes, en su mayoría hay que pagarlos y los que no, no respondían a las especificidades exigidas por la UCI y CubaSearch, quienes necesitaban que sus aplicaciones corrieran sobre la novedosa plataforma .NET y con las bases de datos en Microsoft SQL Server.

De ahí que nos propusiéramos como objetivo el diseño e implementación de un robot que indexe periódicamente los documentos dispersos en la Intranet de la UCI y como futura extensión, que recupere información en el dominio .CU con interfaz en las páginas de CubaSí y que las almacene en un repositorio digital ordenado para las futuras consultas de los usuarios del sistema total.

De cualquier modo, del presente trabajo se espera, más que una solución definitiva, una apertura para llegar a una versión satisfactoria del programa, delimitar las posibilidades de creación del sistema y determinar un diseño sobre el cual se apoyen futuros trabajos más abarcadores.

El documento consta de dos capítulos distribuidos de la siguiente manera:

1. Capítulo 1, en el mismo se hace un estudio de la actualidad de los Sistemas de Recuperación de Información en el Mundo, así como se valoran los diferentes modelos y conceptos que serán utilizados a los largo del documento.
2. Un segundo capítulo donde se abordan las principales características del sistema a elaborar, particularidades del diseño y algunos detalles del uso de las técnicas de Recuperación de Información descritas en el capitulo teórico para la elaboración de la parte funcional de la aplicación.

Capítulo I Fundamentación Teórica

1.1 Estado del arte

Uno de los terrenos de la Inteligencia Artificial donde se está trabajando más constantemente es en el de la Recuperación de Información en Internet. No cabe dudas del motivo de tal necesidad: el vertiginoso ritmo de crecimiento de la red de redes amerita la implementación de mecanismos de búsqueda y recuperación de documentos, no solo como un magnifico punto de partida para la navegación de cualquier usuario ávido de información dispersa (asunto bastante común en estos días); sino también debido a que la elaboración de herramientas de indexación de documentos publicados en Internet, aparte diríamos que es cada vez más imprescindible, debido a la demanda de información y la vastedad y dispersión de lo publicado.

Esto sin dejar de analizar que, por los motivos de la globalización de la información, cada país, institución e incluso las personas naturales se ven en la necesidad de tener su propia representación en Internet, lo cual hace apetecible la posibilidad de existencia de un sitio que devuelva esas informaciones divididas en categorías, ayudando de esta manera a usuarios interesados en encontrar más información acerca de temas específicos.

Para lograr el éxito de cualquier consulta de búsqueda, debe primero existir una colección ordenada de referencias a documentos indexados por palabras, categorías, etc.; pero antes esa colección o base de datos ha de ser suministrada y los encargados de esa operación son o bien operadores humanos (en el caso de los directorios temáticos), o bien programas, en este segundo caso se encuentran los *spiders*, objetos del presente trabajo.

Antes de entrar en detalles acerca de los principales elementos teóricos que forman parte de *spider*, sería bueno analizar una definición de Recuperación de Información (Information Retrieval en inglés), tomándose la misma como la representación, almacenamiento, organización y el acceso a elementos de información, que pueden ser textos, pero también imágenes, sonidos, etc. Visto de un modo menos abstracto, la RI es una operación mediante la cual se interpreta una necesidad de información de un usuario y se seleccionan los documentos más relevantes, capaces de solucionarla, es decir, consiste en buscar

documentos que exhiban un mayor parecido a la pregunta formulada. En el contexto de la WWW, se puede definir el objetivo de la recuperación de información como la identificación de una o más referencias de páginas Web que resulten relevantes para satisfacer una necesidad de información del usuario [Codi--].

Otra cuestión medular en el diseño del motor de búsqueda es saber qué tipo de información va utilizar la interfaz de búsqueda para poder elaborar una implementación coherente que facilite en el repositorio la mayor cantidad de datos que ayuden en el trabajo de la aplicación de tratamiento y respuesta de la consulta del usuario.

No hacemos referencia a los directorios temáticos, pues aunque son dos de las grandes clases de herramientas de recuperación de información en Internet, no son el objeto de estudio de este trabajo.

1.1.1 Motores de búsqueda

Los mismos están compuestos básicamente por [Javo--]:

- Un *spider* o robot que analiza la Internet desde ciertas entradas predefinidas y así analiza los documentos publicados y luego se expande por los enlaces del documento a otros documentos en el mismo sitio o en otros lugares.
- Un algoritmo de indexado que organiza de modo predeterminado la información recuperada.
- Una interfaz que interactúa con el cliente para realizar la consulta y que analiza los resultados volcados por el robot en la base de datos.

Todo esto con ciertas especificidades como [Bae02]:

- Realizar la exploración de los sitios indexables de modo automático.
- Indexar la información sin la intervención del ser humano.
- La representación de la información es explícita: mediante palabras clave, operadores, separadores, etc.

- El resultado de la consulta son páginas que se crean dinámicamente para cada consulta, de modo que aunque se pierda en la precisión del resultado, se gane en el número de documentos candidatos.

Y como tal, los motores tienen por función mostrarle al usuario dónde pueden estar los documentos más relevantes para su búsqueda.

Pero para llegar a esos documentos relevantes deberán enfrentar los siguientes problemas:

- La información normalmente está desordenada.
- Es cambiante.
- Es común encontrar varios sitios diciendo lo mismo de un tema (redundancia).
- Gran cantidad de datos a manejar.
- ¿De qué manera ha de representarse la información para su ulterior procesamiento?
- La determinación de la relevancia del resultado emitido.

Para los cuales se han desarrollado diversas estrategias, herramientas y técnicas en vista de solucionar dichos problemas, pero es la última la que representa el mayor escollo para la efectividad de los motores de búsqueda, pues sin dudas es el campo más trabajoso, ya que los problemas de velocidad y potencia pueden resolverse con los medios materiales adecuados y con ciertos algoritmos altamente optimizados y muchas veces recurriendo al paralelismo, el cual será vital a tener en cuenta en buscador respetable. Desafortunadamente todas esas soluciones, fundamentalmente en el área del software, aunque haya algunas documentadas, son de carácter privado y comercial.

Sin embargo la relevancia de la consulta dependerá de la estrategia desarrollada para el buscador, su política de determinación de la temática, de la importancia de las palabras de la consulta en el documento, entre otras así como de la retroalimentación provista por el usuario (*recall*).

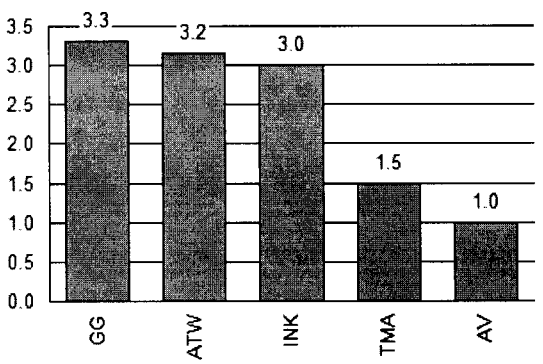
Existen en el mundo muchos motores de búsqueda comerciales y de distribución gratuita (los menos y de peor calidad), cada uno con sus soluciones respectivas a los problemas planteados, entre otros se encuentran los muy conocidos:

- Google (www.google.com)
- Altavista (www.altavista.com)
- HotBot (www.hotbot.com)
- Excite (www.excite.com)
- Lycos (www.lycos.com)
- Infoseek (www.infoseek.com)
- Northern Light (www.northernlight.com)
- All the Web
- Inktomi

Y es interesante para tener una idea de la potencia de los mismos, la encuesta realizada por www.searchenginewatch.com en que arroja a Google como el puntero en cantidad de páginas indexadas, seguido por el reciente All the Web [06].

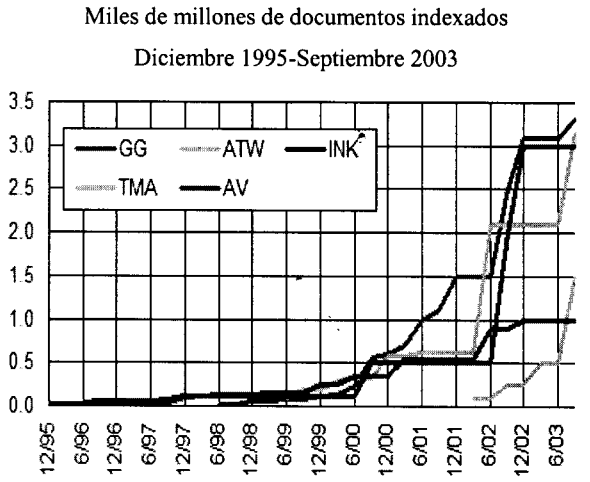
Miles de millones de páginas indexadas

Sept 2, 2003



Leyenda: GG=Google, ATW=AllTheWeb, INK=Inktomi, TMA=Teoma, AV=AltaVista.

Y para tener una idea del crecimiento de los directorios de estos gigantes esta gráfica publicada en el mismo sitio:



Sería interesante analizar los dos casos que encabezan ambas listas.

1.1.1.1 Google.

De todos los motores existentes el de más éxito por el momento es Google, con 4 000 millones de páginas indexadas hasta el momento, este buscador basa su funcionamiento en una red de miles de máquinas de bajo costo que trabajan en paralelo. Muestra muchos resultados, entre los que se encuentran: el título del documento, fragmentos que contengan los términos de la consulta, la URL de la página, la descripción, el tamaño del fichero y una opción llamada *Páginas similares* con la que se pueden localizar documentos relacionados.

Google fue fundado en septiembre de 1998 por Larry Page y Sergey Brin, dos estudiantes de doctorado de Stanford [1]. El nombre proviene de un juego de palabras con el término "googol", acuñado por Milton Sirotta, sobrino del matemático norteamericano Edward Kasner, para referirse al número representado por un 1 seguido de 100 ceros. El uso del término refleja la misión de la compañía de organizar la inmensa cantidad de información disponible en la web y en el mundo.

En este buscador la principal causa de éxito (según sus propios creadores) es su sistema de clasificación PageRank [2], encargado de establecer un orden de relevancia entre las páginas web. Este algoritmo usa los enlaces existentes entre las páginas como base para calcular el valor o relevancia de ellas, Google interpreta un vínculo desde la página A hacia la página B como un voto de A por la página B, a su vez analiza la relevancia de la página

que emite el voto. Los votos emitidos por páginas que son en sí mismas "importantes" pesan más y ayudan a convertir a otras páginas también en "importantes".

1.1.1.2 All the Web.

AlltheWeb apareció en agosto de 1999 como fruto del trabajo de los estudiantes y profesores de la Universidad Noruega de Ciencia y Tecnología [3]. Inicialmente su nombre fue FAST (Fast Search & Transfer), pero al cambiar su diseño en el verano del 2001 el nombre Alltheweb se convirtió en el definitivo. Dispone de una de las bases de datos más importantes en Internet, siendo una de las que se actualiza más frecuentemente (aproximadamente un 30% de sus registros se indexan semanalmente). Se utiliza en Lycos, Scirus y otros motores de búsqueda, muchos de ellos europeos. Sus otras bases de datos (noticias, imágenes, audio, video y FTP) también son de las más amplias de la Red. El buscador funciona bajo servidores Dell PowerEdge 4300 y sistemas de almacenamiento PowerVault, con el sistema operativo FreeBSD.

Como fácilmente se pudo apreciar en las gráficas anteriores, constituye el más serio competidor de Google, con más de 3 000 millones de páginas web indexadas, en muy poco tiempo de existencia. Su principal ventaja sobre este es que actualiza sus bases de datos con una periodicidad semanal, mientras Google lo hace mensualmente.

1.1.1.3 Criterios de evaluación de la eficiencia.

Existen varios criterios para evaluar la eficiencia de la respuesta de un sistema recuperador de información. Entre los más importantes tenemos [Ana02]:

- *Eficiencia de la ejecución*: Toma como medida el tiempo de respuesta del sistema a una consulta.
- *Eficiencia de almacenamiento*: Se basa en la relación entre la cantidad de espacio en memoria que debe reservarse para indexar un documento y el tamaño del documento en sí.
- *Efectividad de la recuperación*: Depende de la relevancia para el usuario de la información recuperada.

Como puede apreciarse fácilmente, la importancia o no de la información de un documento en dependencia de lo que busca un usuario es más bien subjetiva y por tanto, establecerla por mecanismos automáticos es un problema sin solución aún. Sin embargo muchas investigaciones sostienen la posibilidad de al menos establecer un criterio computable que dé alguna medida acerca de este tópico. Para ello se habla de *relevancia* (*recall*) y *precisión*.

La relevancia es la proporción entre la cantidad de documentos relevantes para el usuario que fueron arrojados por la encuesta y la cantidad total de documentos relevantes que hay en la base de datos, o sea [Pons00]:

$$recall = \frac{doc_{relevantes.consulta}}{doc_{relevantes}}$$

Este medidor normalmente será estimado o aproximado mediante expresiones calculadas por muestro u otro método.

La *precisión* vendrá dada por la proporción entre el número de documentos recuperados con relevancia para el usuario y la cantidad de documentos recuperados en la consulta.

$$\text{precisión} = \frac{doc_{relevantes.recuperados}}{doc_{recuperados}}$$

El rango de ambas medidas está entre 0 y 1.

1.2 Modelos de RI

Existen varios modelos de Recuperación de Información, todos con la intención de representar los elementos y mecanismos que inciden en la respuesta y retroalimentación de un sistema recuperador de información. De modo abstracto pudiera representarse como la tupla <D, Q, F, $R(q_i, d_j)$>, donde:

- D es la representación de los documentos.
- Q es la representación de la consulta y sus requerimientos.
- F es el marco para modelar documentos, consultas y sus relaciones.
- $R(q_i, d_j)$ es la función de ranking.

Algunos ejemplos de modelos de recuperación de información son:

- Modelo booleano.
- Modelo vectorial.
- Modelo estadístico.
- Modelos basados en inferencias.
- Modelos de indexado de semántica latente.
- Modelos de búsqueda de cadenas.
- Modelos basados en redes neuronales.
- Modelos estructurados

Pero normalmente, para fines sencillos, se utilizan los dos primeros por su simplicidad y facilidad de implementación.

1.2.1 Modelo booleano.

Es el más simple de los modelos de recuperación de información y consiste en que todo documento está conformado por una serie de palabras claves a las que se le asocia valores de 1, si se encuentra y 0, si no. Las consultas consisten en expresiones booleanas tradicionales donde las variables son las palabras en cuestión, conectadas mediante operadores lógicos.

La semejanza entre la consulta y el documento estará dada por el resultado de la expresión sobre el documento, siendo 1 si es recuperable y 0 si no cumple con la consulta.

Entre las ventajas con que cuenta este modelo podemos citar:

- Fácil de implementar.
- Tiene en el Álgebra de Bool un fundamento teórico muy sólido.

- Provee de grandes niveles de eficiencia en materia de tiempo de ejecución y en la creación de los índices.

Y como desventajas:

- Valores pobres de relevancia y precisión.
- Recuperación deficiente, malo para obtener consultas satisfactorias.
- Como el sistema es binario, es más de recuperación de datos que de información.
- Muy estricto con las consultas, aunque una parte arroje resultados verdaderos si no se satisface la expresión completa, posiblemente no se recupere un documento relevante.
- Todos los términos poseen igual importancia.
- No se puede establecer un orden entre los documentos.

Dos de las formas más utilizadas de este método vendrían a ser:

- Vectores de bits
- Hashing
- Conjuntos Difusos

Esta última es una variante, donde a cada término del documento se le asigna un valor (peso) entre 0 y 1, que indicará el grado en que el término indexa al documento. En este caso, la recuperación del documento relevante a la consulta dependerá de la similitud entre ambos, solo que los operadores no serán los tradicionales pues habrá que redefinirlos para que trabajen en un entorno continuo.

1.2.2 Modelo vectorial.

En el modelo vectorial cada documento se representa como un vector n-dimensional donde n es el cardinal del conjunto de los términos de indexación elegidos para la colección, donde cada elemento tomará un valor de relevancia (peso) a medida de cuan representativo sea o no en el documento. Un término que no aparezca no tendrá representatividad, por tanto, su peso será 0.

Esta sería la representación del documento i-ésimo, y $w(t_j, d_i)$ es el peso del término t_j en el documento d_i:

$$d_i \longrightarrow \vec{d_i} = (w(t_1, d_i), \ldots, w(t_k, d_i))$$

Existen varias fórmulas para calcular este peso, entre las más populares se encuentra:

$$w(t_r, d_i) = w_{r,i} = \frac{tf_{r,i} \times idf_r}{|\vec{d_i}|} = \frac{tf_{r,i} \times \log \frac{N}{n_r}}{\sqrt{\Sigma_{s=1}^{k} \left(tf_{s,i} \times \log \frac{N}{n_s}\right)^2}}$$

Donde $tf_{r,i}$ es la cantidad de veces que aparece el término r-ésimo aparece en el documento i-ésimo y n_r es la cantidad de documentos en los que aparece dicho término.

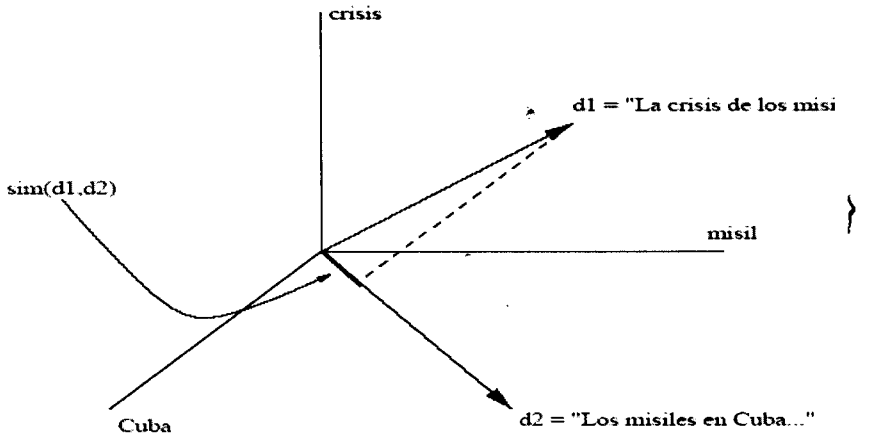
Entonces, se deberán establecer algunas estrategias a la hora de la recuperación que favorezcan los niveles de efectividad y relevancia. Entre otras podríamos citar:

- Si un término aparece mucho en un documento su peso aumentará; pero si aparece en muchos documentos, disminuirá, pues este término no es muy útil para distinguir un documento de otro.
- Los vectores se deben normalizar para no favorecer documentos más largos.

La similitud entre dos documentos vendrá dada por *la distancia coseno*:

$$sim(d_i, d_j) = \vec{d_i} \cdot \vec{d_j} = \sum_{r=1}^{k} w_{r,i} \times w_{r,j}$$

que no es más que el seno del ángulo entre ambos vectores.



de ahí que el valor de la similitud entre documentos esté entre 0 y 1. O sea, dos documentos iguales tendrán similitud 1 y dos documentos ortogonales, no tendrán similitud alguna. De modo, que una consulta podrá considerarse también un documento y se recuperarán del directorio digital aquellos documentos cuya similitud sobrepase un umbral determinado.

Por tanto, podría calcularse la relevancia de la recuperación de información relativa a una consulta q, por la fórmula:

$$sim(d_i, q) = \sum_{r=1}^{k} w_{r,i} \times w_{r,q} \sim \sum_{t_r \in q} w_{r,i} \times idf_r$$

que no es exactamente igual a la anterior, pero permite establecer un orden en la respuesta ofrecida.

Este método es el más popular debido entre otras a las siguientes ventajas:

- Permite que la consulta sea tomada como un documento.
- Permite hacer *clustering* entre los documentos.
- Arroja buenos resultados en el *recall* y la *precisión*.
- Facilidad de implementación.
- Los niveles de eficiencia de ejecución son satisfactorios.

1.3 Formas de Almacenamiento de la Información.

En el caso del *spider* cuya función es la de suplir los datos del repositorio que va a utilizar el motor de búsqueda, es vital la determinación de qué se va a guardar de los ficheros y cómo va hacerse. En mucho de los casos dependerá del modelo utilizado, pues hay informaciones que el robot habrá de colectar acerca del contenido y el formato del documento, en dependencia de los requerimientos de la interfaz recuperadora.

En caso de que la información se guarde indexadamente, puede organizarse con alguno de los siguientes métodos:

- Archivos invertidos.
- Archivos de firmas.
- Árboles PAT.

Si de lo contrario, la información no se fuera a almacenar de forma indexada, pueden utilizarse los ficheros planos.

No caben dudas que factores como la periodicidad de la actualización de la información en el repositorio y el tamaño del mismo, influirán por escoger el almacenamiento indexado. Así, una colección de documentos que varía muy frecuentemente no parece adecuada para ser indexada, ya que no va a dar tiempo a amortizar el esfuerzo invertido en la indexación de la colección de documentos con el tiempo ganado a la hora de resolver las consultas. Por otra parte, en el caso de que el tamaño de la colección de documentos sea grande, el único modo de responder a las consultas en un tiempo razonable será disponiendo de un índice. En el caso de una colección de documentos muy grande, con una tasa de actualización muy alta se tendrán que utilizar técnicas de actualización de índices o alguna solución de compromiso.

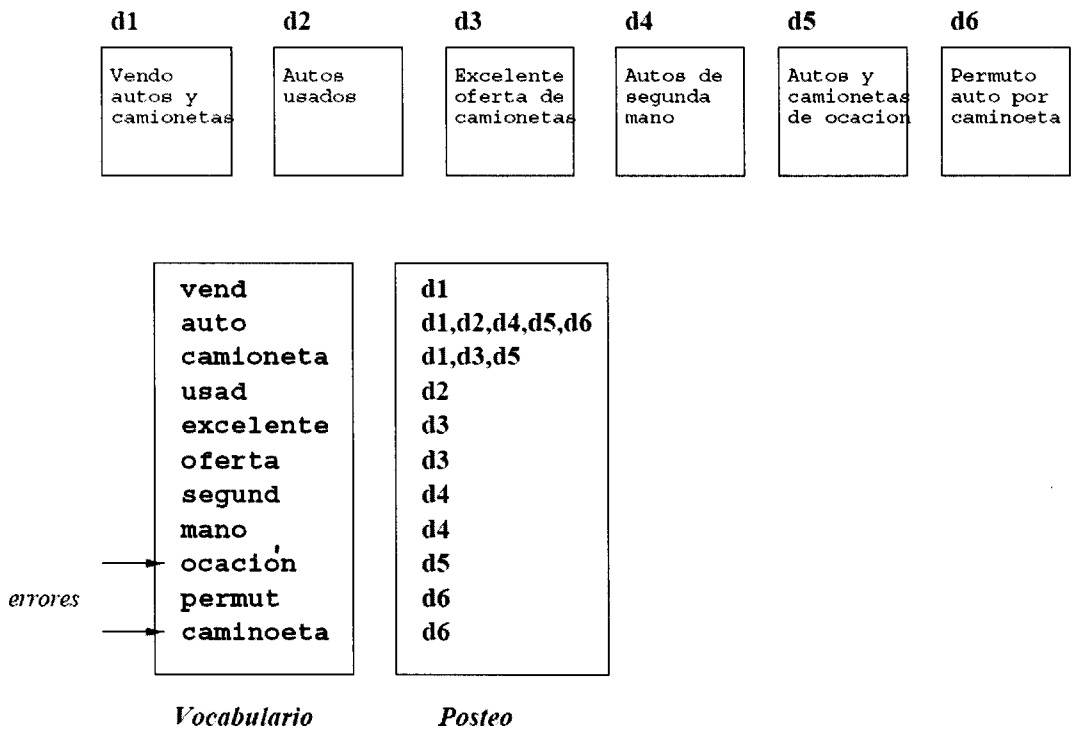
1.3.1 Ficheros planos.

Es la manera más elemental de guardar la información recopilada, según la cual uno o más documentos se almacenan en un archivo (por lo general un archivo de texto ASCII), que trae ciertas consecuencias sobre la búsqueda, como que ésta se realice mediante localización de patrones de texto. Puede emplearse en el modelo booleano y para almacenar

el contenido completo del documento para otros usos (muestra del texto exacto en la respuesta de la consulta).

1.3.2 Ficheros Invertidos.

En su forma más básica los ficheros invertidos representan la información asociando a cada término los documentos donde aparece.



Por supuesto que en este caso, si no se cuenta con más información, si no se almacenó nada más de los documentos, solo podrá recurarse si los términos están o no en un documento, lo cual es más aproximado al modelo vectorial, y por lo general se quiere hacer consultas más complejas. Por eso una representación más general y satisfactoria podría ser:

idtérmino	*iddocumento*	*idcampo*

donde *idtermino* es el identificador del término, así como *iddocumento,* que definen unívocamente al registro en cuestión, los cuales mientras que *idcampo* representa otras informaciones que se almacenarán del término en ese documento.

Se han descrito una serie de variantes que podrán observar los ficheros invertidos en dependencia del tipo de modelo que se desee implementar con ellos. Estas estrategias y cambios de estructura, incluyen ordenamientos, pero fundamentalmente almacenar información extra que será utilizada por la interfaz de búsqueda en la recuperación de los documentos, para alimentar las fórmulas de cada modelo, etc.

Como principal ventaja pudiéramos mencionar el aumento de la eficiencia de ejecución, pues el sistema no habría de recorrer todo el documento en busca de los términos sino que sabría directamente por el término qué documentos lo contienen. Por otro lado, es muy fácil de usar e implementar.

Entre sus desventajas contamos con el costo de actualización y de reorganización del índice.

1.4 Métodos de indexado.

1.4.1 Definiciones.

La indexación es la operación destinada a representar los resultados del análisis del contenido de un documento o de una parte del mismo, mediante elementos (denominados genéricamente *términos de indexación*) de un lenguaje documental o natural, generalmente para facilitar la recuperación.

La indexación puede realizarse utilizando diversos métodos [Vill98]:

- Indexación manual con vocabulario controlado: consiste en la asignación por los expertos de los términos de indexación a partir de un conjunto finito de los mismos.
- Indexación automática con vocabulario controlado: automáticamente se escogen los términos de los existentes en un conjunto finito.
- Indexación libre: son los propios autores de los textos los que asignan los términos de indexación pensando en los términos que utilizarían los usuarios si buscasen información sobre los temas de sus textos.
- Indexación a través del lenguaje natural: los términos de indexación los elige la computadora a partir de un análisis de los textos. Este último tipo aparece hoy en día debido a la gran variedad de textos disponibles en formato electrónico y es el centro de atención en las investigaciones actuales en Recuperación de Información.

El método más empleado en este tipo de indexación es tomar como índices las palabras del texto. La gran dificultad de este enfoque es que si indexamos un texto basándonos en las

palabras del lenguaje natural, podemos obtener decenas de miles de características, una cantidad más allá de la que podríamos caracterizar como óptima. Además, en una representación basada en palabras hay una redundancia considerable, en el sentido de que aparecerán conjuntos de características sinónimos o casi sinónimos. Si dos palabras son sinónimas podríamos pensar que deberían estar en los mismos documentos y añadirlas como características. Otro efecto perjudicial es la ambigüedad en su significado, entendiéndola como la existencia de dos o más significados para una misma palabra o expresión lingüística. La ambigüedad debemos evitarla para las representaciones de los textos, puesto que es de suponer que es poco probable que todos los significados sean de interés en un texto determinado.

1.4.2 Niveles de Indexación.

Los niveles de indexación pueden lograrse de varias maneras. Algunas de las más utilizadas son:

- Por palabras.
- Por conceptos.
- Por hipervínculos.

En el primero de los casos, el más utilizado, el índice se construye mediante los términos clave que han sido obtenidos de los documentos recuperados por el robot, y normalmente constituyen índices para definirlos unívocamente. Se basa fundamentalmente en análisis morfológicos y estadísticos, que en términos de RI se traduce en verificar la similitud de las palabras con la estadística de la presencia de las mismas en los documentos a recuperar.

La indexación por conceptos, recurre a la clasificación conceptual de la información y para ello requiere de complejos procedimientos de construcción de bases de datos conceptuales que apelan a engorrosos métodos lingüísticos y de Inteligencia Artificial, que a partir de análisis estadísticos trata de determinar la presencia, adyacencia y otras características de temas dentro de los documentos recuperados.

El último proviene de representar los documentos como vértices de un grafo cuyos arcos son los enlaces, de modo que dos documentos con enlaces comunes supuestamente deberán tratar temas similares.

1.4.3 Procesamiento de los términos de los documentos.

En los documentos, por lo general, aparecen datos de formato, enlaces, informaciones del autor, temas, palabras clave, títulos..., que aunque pueden contribuir a esclarecer la naturaleza del tema central del documento, el contenido mismo, o sea el texto puro es el eje central de la RI por lo que luego de obtener la información de relevancia contenida en los metadatos, debe seguirse un proceso de filtrado que incluye algunos pasos como los siguientes:

- Purificar el texto de etiquetas, descripción de formato, etc., para finalmente obtener el texto puro.
- Eliminar las *stopwords*, la cuales pueden abarcar un espacio considerable de las palabras del texto en cuestión. Esta técnica se utiliza para eliminar palabras cuya frecuencia de aparición es alta, pero de escaso valor semántico en el texto. Dichas palabras tampoco son muy útiles para diferenciar un documento de otro.
- Reconocer estructuras multipalabra como frases sustantivas, nombres propios, etc.
- Lematización. Con esta técnica se pretende agrupar los términos por lexemas o raíces y familias de palabra. Se determinan las partes comunes de las palabras, forma canónica, obteniéndose finalmente un término, que representa la base de una familia de términos relacionados semánticamente.
- Truncamiento. El truncamiento se refiere a la "mezcla" o "unión" de términos, utilizando caracteres especiales en la palabra, de ahí que el término truncado se referirá a muchas palabras. Lo más común es que se refiera a palabras con una raíz común. Por ejemplo *"info"* se referirá a palabras tales como *informativo*, *informadores*, *información*, etc.
- Utilización de un tesauro. Un tesauro proporciona una agrupación o clasificación de términos en un determinado dominio o área, en categorías denominadas clases. Un

tesauro permite identificar términos lexicográficamente diferentes pero equivalentes semánticamente (sinónimos). Así se elegirían, como términos de indexación, un representante de cada clase del tesauro.

Esto no quiere decir que todas las operaciones se llevarán a cabo con cada término o documento. Luego que el texto sea depurado de etiquetas y palabras vacías, los términos candidatos podrán pasar por algunas de estas acciones para así facilitar el proceso de indexación o para aumentar su representatividad en la indexación de los documentos, como algo mayoritario podría decirse que a los términos se les extraerá la raíz, que será quien se indexe en lugar del término propiamente dicho. No hay que decir que los términos deben homogenizarse: no distinción entre mayúsculas y minúsculas, no vocales acentuadas, etc.

1.4 Robot, Crawler *o Araña.*

Dentro del motor de búsqueda la tarea de llenar el directorio virtual, corre a cargo de un robot o *spider*, que es una aplicación que recorre automáticamente y periódicamente Internet, visitando de manera recursiva las páginas y almacenando todo la información pertinente. Esta información se indexa y se pone en una base de datos, que después será recorrida por la aplicación del buscador. Es también acción suya investigar si el documento continúa en el mismo lugar y si ha sufrido cambios.

No existe una fórmula específica para el trabajo del robot, de hecho las grandes empresas de la búsqueda guardan celosamente dicha información, sin embargo se tiene una idea consensuada de la composición de los mismos y de su comportamiento, pese a que cada robot guarde información diferente, según el modelo e intereses que se tengan.

Al principio, apenas almacenaban palabras de partes específicas del documento (título, campo de palabras clave, autor, etc.), para después expandir su acción al contenido del documento.

Normalmente el robot comienza su rastreo desde una lista de direcciones base suplida manualmente y que él mismo con el tiempo irá engrosando, normalmente estas direcciones base son directorios o grandes portales, desde los cuales, podrá accederse, visitando recursivamente los URLs que vayan apareciendo, una gran porción de la Red de redes. Por supuesto, que se corre el riesgo de tratar de indexar un conjunto clausura de sitios y que se

queden fuera sitios de importancia para muchas temáticas. Esto de cualquier manera es inevitable y es por ello que todavía hoy gigantes buscadores no indexen sino una porción bastante irrisoria de Internet.

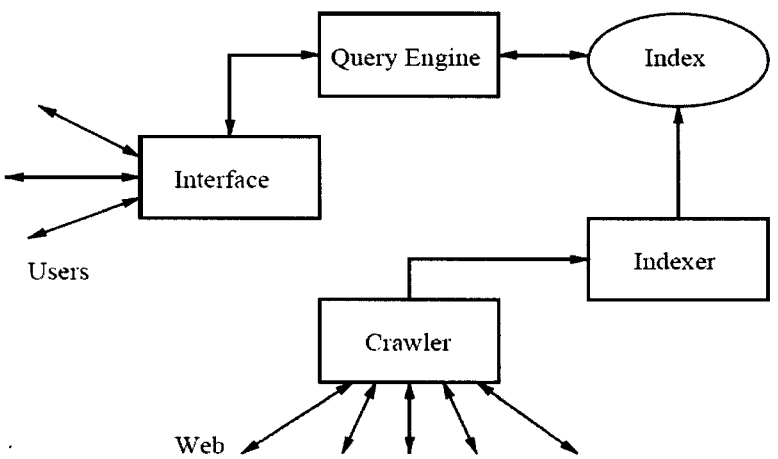
La exploración puede hacerse primero a lo ancho o primero en profundidad, en el primero de los casos, los enlaces se ponen en cola esperando su turno para ser analizados. Este proceso se activa o bien de modo manual o bien cada un tiempo especificado.

1.4.1 Arquitecturas de spider.

Normalmente se conocen dos tipos de clases de arquitectura de spiders: centralizadas o distribuidas. En ambos casos la composición del spider se puede descomponer en dos grandes elementos: el *crawler* encargado de recorrer Intranet y de descargar los documentos y el *indexador* que indexa la información descargada.

1.4.1.1 Arquitectura centralizada.

Las primeras son aplicaciones más o menos monolíticas que corren en una máquina y que solicitan las páginas a los servidores y las almacenan y procesan localmente, tributando a una base de datos [Bae02].



Esta arquitectura tiene que enfrentar varias desventajas:

- Problemas de potencia y velocidad de las máquinas donde corren. La exploración es la actividad más lenta para que se haga de modo centralizado, aparte que el volumen de información a manipular y la velocidad a la que se requiere que se haga puede

traer serios problemas para que se realice en el futuro bajo una filosofía centralizada.

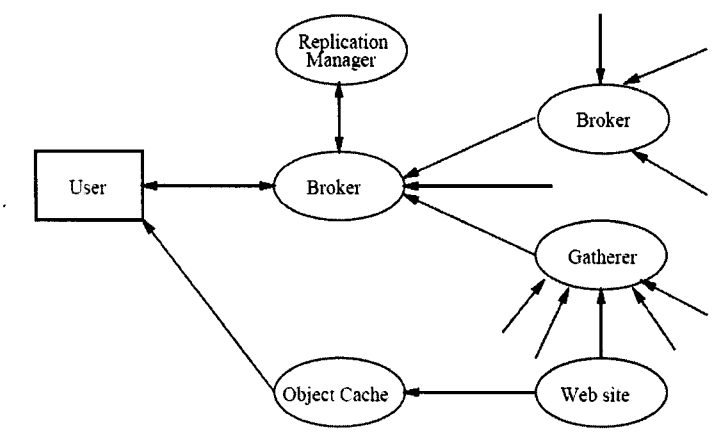
- Por las solicitudes del *crawler* los servidores Web pueden saturarse.
- La red puede saturarse tras el traslado de documentos cuyo contenido mayoritario será desechado tras el proceso de filtrado e indexación.
- No hay cooperación entre los robots.
- Deben cumplirse las normas de cortesía (metaetiquetas, robot.txt).

1.4.1.2 Arquitecturas distribuidas.

Los *crawlers* distribuidos se basan en dos componentes fundamentales [Bae02]: los recolectores y los *brokers*. Los primeros recolectan la información a indexar desde Internet. Los segundos, indexan la información de otros recolectores o *brokers* y responde a las consultas.

El modo de organización puede ser bastante variable, pues cada quien estructura según sus intereses, o según el modelo escogido y la información que desea indexar.

Por ejemplo un recolector puede trabajar de manera local y así no saturar demasiado los servidores Web y entregar los documentos a diversos *brokers*, evitando difundir información repetida. Mientras tanto, los *brokers* construirán los índices y los propagarán a otros *brokers*, mezclando cuando haga falta, tratando de evitar indexar tanto y emitir tanta información.



1.4.2 Algoritmo general del robot.

El modo de funcionamiento general de un robot podría sintetizarse de la siguiente manera:

1. Se arranca tomando el primer URL explorable (esto dependerá de la respuesta del servidor Web) de la base datos de URLs. Dicha lista deberá haberse llenado la primera vez con una serie de sitios lo suficientemente grandes como para que la búsqueda recursiva por ellos arroje la mayor cantidad de nuevos sitios, se preferirán los grandes directorios y portales.
2. Se escoge un URL de la lista, si no ha sido analizada y si ha cambiado, entonces se almacena el documento.
3. Se analiza el documento para recuperar la información contenida en ella y los posibles enlaces a nuevas páginas y luego se indexan.
4. Los enlaces encontrados se almacenan en la lista de URLs se marca el visitado para que no vuelva a ser recorrido.
5. Si en la lista todos los enlaces quedaran marcados o si ocurriera algún tipo de evento final (tiempo de espera agotado, espacio insuficiente, número máximo de índice alcanzado,... etc.), el algoritmo finaliza. En caso contrario, se regresa al paso dos.

El robot en cada iteración solicita un documento en formato HTML u otro y lo descarga como mismo lo haría un *browser*, solo que una vez en disco, el documento será procesado (parseado del contenido, proceso donde se separan las órdenes de formato del contenido puro) para obtener informaciones que le sirvan para llenar los respectivos registros de la base de datos que usará después el buscador. Dichas informaciones serían el título, autor, palabras clave, todas obtenidas del texto, reconociendo en el caso de los dos primeros por las etiquetas o campos correspondientes del documento.

Una vez indexado el documento, el robot, apoyado por el subsistema localizador y analizador, identifica los enlaces que contiene el documento con otras partes del mismo documento u otros y de forma recursiva, procede a recuperar los documentos referenciados, procediendo a su indexación, obtención de nuevas referencias, etc.

Cada implementación es libre de hacer variaciones de los pasos antes referidos pero en general son esos los mecanismos.

1.4.3 Zonas de prohibición para robots.

Existen las llamadas políticas de cortesía para motores de búsqueda, que han de ser tomadas en cuenta a la hora de indexar esos documentos, pues por lo general es intención de los autores o publicadores, que no sea de conocimiento general, por su carácter especial. Existen divergencias al respecto, pero el problema está ahí.

Una solución está en leer el documento *robot.txt* que normalmente se coloca en la raíz de los servidores de publicación Web e incluso se ponen en la base del sitio en cuestión, donde excluyen documentos de los que no se desea sean indexados. De este modo, se prescribe la exclusión de esos documentos de la acción de los robots recuperadores.

Existe un acuerdo descrito desde 1994, tras su acuerdo en el foro de robots-request@nexor.co.uk en junio de ese año, bajo el nombre de *Standard for Robot Exclusion* [Kost94].

Sin embargo, pocos son los servidores que tienen el fichero en su base, con lo cual impiden el principio de exclusión y se arriesgan a que los *crawlers* que admiten este protocolo en su comportamiento indexen y por tanto, publiquen contenido de alguna manera clasificado o viole las políticas de privacidad del sitio.

1.5 Tecnología .NET.

En la implementación del sistema será utilizada la plataforma .NET de Microsoft, tecnología que dado el nivel de integración y facilidades de desarrollo que posee resulta muy conveniente.

Microsoft .NET es un conjunto de nuevas tecnologías que persiguen obtener una plataforma sencilla y potente para distribuir el software en forma de servicios que puedan ser suministrados remotamente y que puedan comunicarse y combinarse unos con otros de manera totalmente independiente de la plataforma, lenguaje de programación y modelo de componentes con los que hayan sido desarrollados [Gonz01].

1.5.1 Lenguaje Común de Rutinas (CLR).

El Common Language Runtime (CLR) es el núcleo de la plataforma .NET. Constituye el motor encargado de gestionar la ejecución de las aplicaciones para ella desarrolladas, a las que ofrece numerosos servicios que simplifican su desarrollo y favorecen su fiabilidad y seguridad. El CLR presenta diversas características algunas de las cuales son:

Modelo de programación consistente: A todos los servicios y facilidades ofrecidos por el CLR se accede de la misma forma: a través de un modelo de programación orientado a objetos. Esto es una diferencia importante respecto al modo de acceso a los servicios ofrecidos por los algunos sistemas operativos actuales (por ejemplo, los de la familia Windows), en los que a algunos servicios se les accede a través de llamadas a funciones globales definidas en DLLs y a otros a través de objetos (objetos COM en el caso Windows)

Modelo de programación sencillo: Con el CLR desaparecen muchos elementos complejos incluidos en los sistemas operativos actuales (registro de Windows, GUIDs, HRESULTS, IUnknown, etc.).

Ejecución multiplataforma: El CLR actúa como una máquina virtual, encargándose de ejecutar las aplicaciones diseñadas para la plataforma .NET. Por lo que, cualquier plataforma para la que exista una versión del CLR podrá ejecutar cualquier aplicación .NET.

Integración de lenguajes: Desde cualquier lenguaje para el que exista un compilador que genere código para la plataforma .NET es posible utilizar código escrito en otro lenguaje para el que también exista un compilador de .NET.

1.5.2 ADO .NET

Microsoft ADO .NET es la opción dentro del .NET framework para el acceso a bases de datos. Ofrece un mecanismo de acceso a diferentes gestores de bases de datos y lo hace sobre un modelo escalable. Aunque ADO.NET conserva algunos de los conceptos del anterior ADO se ha mejorado considerablemente para permitir el acceso a información estructurada procedente de distintas fuentes a través de un consistente modelo de programación estandarizado. Presenta además un modelo desconectado de acceso a los

datos que le permite trabajar aún cuando se haya perdido la conexión con el origen de datos.

1.5.3 ASP .NET

La tecnología ASP .NET es la sección del *framework* destinada al desarrollo de aplicaciones para la Web, según las bondades y restricciones que impone .NET. Constituye un marco de trabajo sin precedentes en cuanto a facilidades de desarrollo y posibilidades de escalabilidad de aplicaciones empresariales, permite además implementar soluciones informáticas para una diversidad de dispositivos electrónicos y dar en general cualquier servicio de información a través de la Internet, en fin ha sido creada con el ojo puesto en sacarle el mayor provecho a las potencialidades alcanzadas por la tecnología [08].

1.6 Microsoft SQL Server.

Este es el servidor de bases de dato por excelencia de varias empresas e instituciones de nuestro país y el mundo. Es, junto con el Oracle, uno de los dos tipos de especializaciones que incorporan las componentes de ADO .NET para trabajo con sistemas de gestión de bases de datos.

Entre sus principales ventajas se tienen el alto nivel de integración con la plataforma Windows y la característica ya mencionada acerca de su afinidad con el *framework* .NET, sin embargo se le atañen serios problemas de prestación cuando el número de solcitudes y el tamaño de las mismas crece, fundamentalmente si se les compara con Oracle y MySQL.

Sin embargo, para los fines que se proponen las instituciones que dirigen la presente investigación, constituye una herramienta satisfactoria.

Capítulo II Características, Análisis y Diseño del Sistema

Introducción

En este capítulo se produce la descripción de las diferentes características del sistema, el objeto de estudio, situación problémica, propuesta del sistema, así como la relación de los principales requisitos funcionales que deberá cumplir el producto final.

2.1 Preliminares, problemática y objetivos.

La búsqueda de información es un asunto medular para cualquier usuario de Internet, pues él tiene ante sí todo un océano de documentos dispersos y por tanto, encontrar lo que necesita es harto difícil sin la asistencia de un buscador. Este problema puede verse también en ámbitos más estrechos, especialmente en el caso de los centros de educación superior donde también se llevan a cabo procesos de digitalización de los documentos existentes y se publican a lo largo de la intranet, fundamentalmente en sitios *web*, aunque no se excluyen sitios *ftp* o carpetas de recursos compartidos. ¿Qué deben hacer los usuarios de estos lugares para acceder a la información propuesta si pudiera ser que una parte del tema esté en un lugar y otro, más lejos? Para ellos que la universidad incorporase un buscador sería una solución ideal.

2.1.1 Situación problémica.

En nuestro caso, aparece la necesidad de la implementación de un robot recuperador que extraiga de Internet o Intranet los documentos y los indexe y almacene en una base de datos para su posterior recuperación por parte de la interfaz de respuesta a las consultas.

La Universidad de las Ciencias Informáticas (UCI), pionera en Cuba en varias prácticas de informatización del acceso al conocimiento, no cuenta con un *spider* que indexe los documentos publicados en el ámbito de su Intranet. Otro tanto le sucede a nuestro país, que aún no cuenta con un motor de búsqueda capaz de mostrarle a usuarios nacionales y extranjeros la información publicada a lo largo de la red nacional que puede resultar de importancia para sus fines.

2.1.2 Objeto de estudio.

De ahí que pueda definirse como objeto de estudio la recuperación e indexación de la información en los ámbitos de la Intranet de la UCI y la Internet cubana mediante un spider preferiblemente centralizado.

2.1.3 Problema.

Como problema podemos definir las crecientes disponibilidades y demandas de información en la UCI y la red cubana, que han llegado a un límite donde la búsqueda se hace engorrosa; sobre todo, si se depende de lo que otros buscadores han logrado indexar o si, como es el caso de la UCI, la red no tiene salida y por tanto no puede ser analizado por motores externos. Entonces se impone la necesidad de localizar esa información y disponerla en un directorio para su posterior consulta.

2.1.4 Información que se maneja.

Aunque en una primera aproximación se propuso la idea de tratar toda una amplia gama de documentos que aparecen normalmente publicados en Internet (pdf, doc, rtf, txt, xml, vrml, html, shtml), sin embargo tras un análisis de las posibilidades reales del sistema en vista del tiempo con que se contaba para acometer el proyecto, se determinó solo atender el caso de las extensiones del Hyper Text Mark up Language y texto plano, pues el resto, como se verá más adelante, solo tendrá que tenerse en cuenta durante el proceso de filtrado del texto puro y visto de modo independiente como un proceso que acepta el fichero en cuestión y lo parsea devolviendo el texto puro más los otros campos que solicite el subprograma de indexación.

2.2 Propuesta del Sistema.

Es intención del presente proyecto arribar a un diseño e implementación de un robot recuperador e indexador de documentos para la UCI y el proyecto CUBASEARCH de CubaSí de ETECSA. Su funcionamiento deberá permitir la actualización automática de la información del Directorio, así como garantizar mecanismos para lograr otras extensiones.

2.2.1 Requerimientos funcionales del sistema.

Para ello, según lo expuesto en el capítulo previo, el sistema deberá cumplir a grandes rasgos los siguientes requisitos de funcionamiento:

1 Incorporar nuevos hipervínculos al directorio.

1.1 Se solicita la nueva dirección y si no existe en el directorio se incorpora al final del mismo.

2 Gestión de información en el Índice (Directorio).

2.1 Deben garantizarse mecanismos para insertar, borrar y actualizar los distintos elementos del Directorio Virtual (Documentos, Palabras, Stop Words, Ficheros Invertidos, etc.).

2.2 Elaboración de Web Services que cubran estas operaciones.

3 Obtener el documento a partir de una URL del Directorio.

3.1 El sistema solicita al servidor Web el URL en cuestión y como respuesta deberá recibir el correspondiente documento.

4 Indexar el documento obtenido.

4.1 Se parsea el documento y se filtran etiquetas y delimitadores de formato para obtener el texto puro y el resto de los campos que serán tenidos en cuenta para su posterior indexación.

4.2 Indexar, según alguno de los criterios antes expuestos.

2.2.2 Requisitos no funcionales del sistema.

Los mismos describen cualidades adicionales que debe presentar la solución del sistema y que aunque puedan influir en las funcionalidades principales del sistema, no son precisamente el núcleo del funcionamiento de la aplicación.

En el caso de nuestro *spider*, que no cuenta con ninguna interfaz visual específica, no son necesarias hacer caracterizaciones de la misma, excepto para el caso en que se quieran detallar ciertas configuraciones o agregar manualmente un nuevo enlace de partida. Sin embargo, entre las facilidades que debería observar el proceso de diseño implementación

debería responderse a favores factores como la velocidad de ejecución, intentar no sobresaturar con solicitudes excesivas al servidor de bases de datos y el de páginas web. Sin descontar que deben respetarse los principios de zonas de exclusión para robots descritos en el capítulo I.

2.2.3 Definición de casos de uso.

2.2.3.1 Definición de los actores de los casos de uso del sistema.

Actores	Justificación
Administrador	El mismo intervendrá en los procesos de inserción de una nueva URL de partida y en el arranque del sistema
Tiempo	Aunque de por sí mismo no es un personaje real, cómo el sistema puede activarse de manera automática, cada cierto intervalo de tiempo, se considera como un actor más del sistema.

2.2.3.2 Caso de uso.

CU-1	Exploración y indexado
Actores	Administrador y Tiempo
Descripción	Activa la exploración, pide los URLs de la tabla de URLs y los explora recursivamente, en nuestro caso primero a lo ancho, en busca de actualizaciones o nuevos documentos que luego serán indexados al Directorio.
Referencia	R1

2.2.3.3 Diagrama del caso de uso.

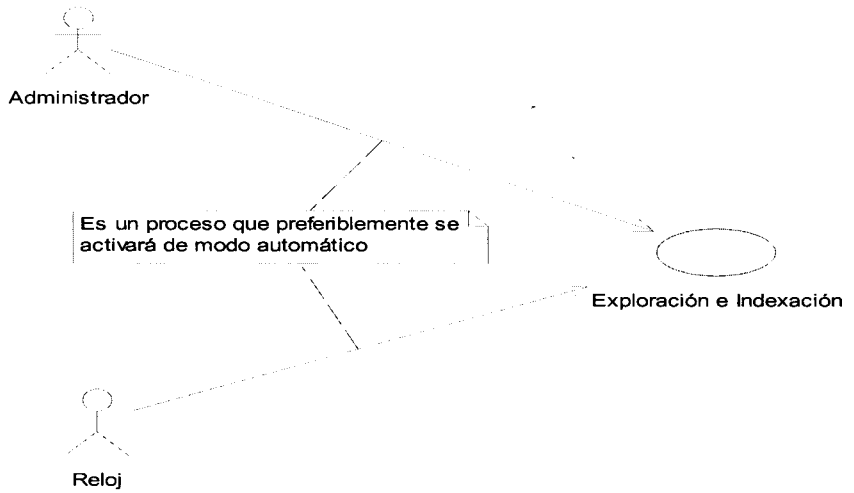


Fig. 2.1 Diagrama de caso de uso

2.3 Análisis y diseño del sistema.

2.3.1 Arquitectura del sistema.

Según lo visto en el primer capítulo, donde se hacía una revisión de los distintos elementos teóricos que componen las bases de los Sistemas de Recuperación de Información, se analizó las potencialidades y desventajas de las distintas arquitecturas que siguen la implementación de dichos sistemas, en nuestro caso apostamos por una solución relativamente simplista, en vista de los problemas de tiempo con que contamos: el modelo centralizado, dejando para una futura revisión las posibilidades de una arquitectura distribuida.

De modo que la arquitectura del sistema podría resumirse en abstracto con el siguiente gráfico:

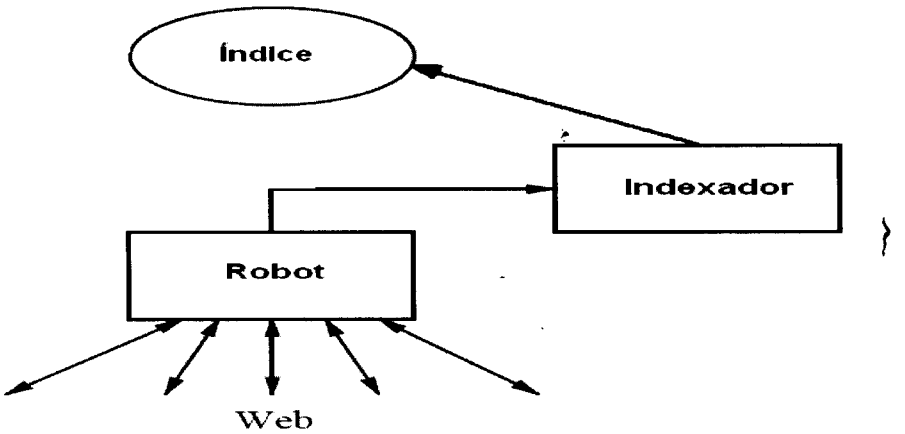


Fig. 2.2 Robot de arquitectura centralizada

La cual en una expansión, pudiera expandirse a la siguiente que da una idea más detallada de los subsistemas del robot.

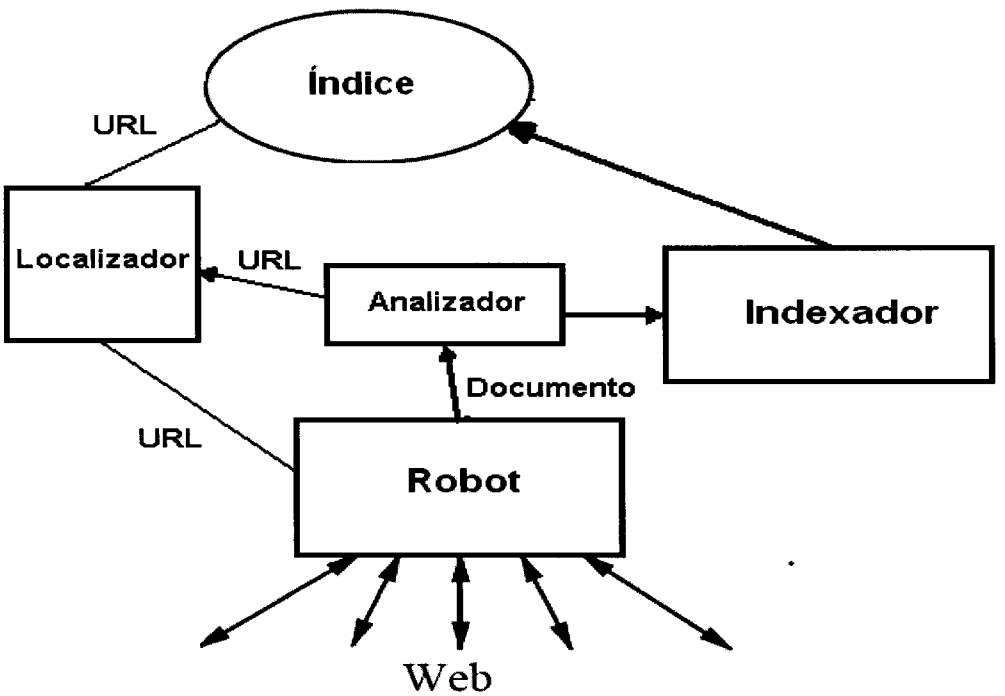


Fig. 2.3 Arquitectura del sistema. Subsistemas.

Podríamos detenernos a explicar, a partir del diagrama representado en la figura 2.3 cuales son las funciones generales de cada subsistema.

2.3.1.1 Robot.

El robot tiene como función tomar un enlace servido por el localizador y hacer la solicitud al Servidor Web de la página o documento correspondiente. Posterior a la recuperación del documento de Internet, el mismo se entrega íntegramente o con un mínimo de procesado al analizador.

2.3.1.2 Localizador.

Entre las funciones del subsistema localizador se encuentran la de obtener la lista de enlaces a recorrer, su inicialización... Tendrá a su cargo además la tarea de alimentar el robot con enlaces y se relaciona con el analizador pues este puede arrojar nuevos enlaces que serán agregados a la lista de enlaces.

2.3.1.3 Analizador.

En esta componente se realiza gran parte del trabajo duro del motor, pues entre sus ocupaciones se encuentran desbrozar el texto de información de formateado, determinar el o los idiomas del documento, hacer ciertos cálculos estadísticos de frecuencia de partición de los términos y debe como fin obtener los nuevos enlaces para el Localizador y los términos y metadatos que serán la entrada del subsistema encargado de Indexar la información recuperada.

2.3.1.4 Indexador.

El subsistema de indexado crea a partir de los términos y metadatos asociados del documento actual, los correspondientes registros que serán almacenados en el Índice y directorios. Por supuesto, que en este caso deberá seguirse algún criterio de indexado, según lo visto en el capítulo I.

2.3.2 Diagrama de secuencia.

Fig. 2.4 Diagrama de secuencia

2.3.5 Diagramas de clase.

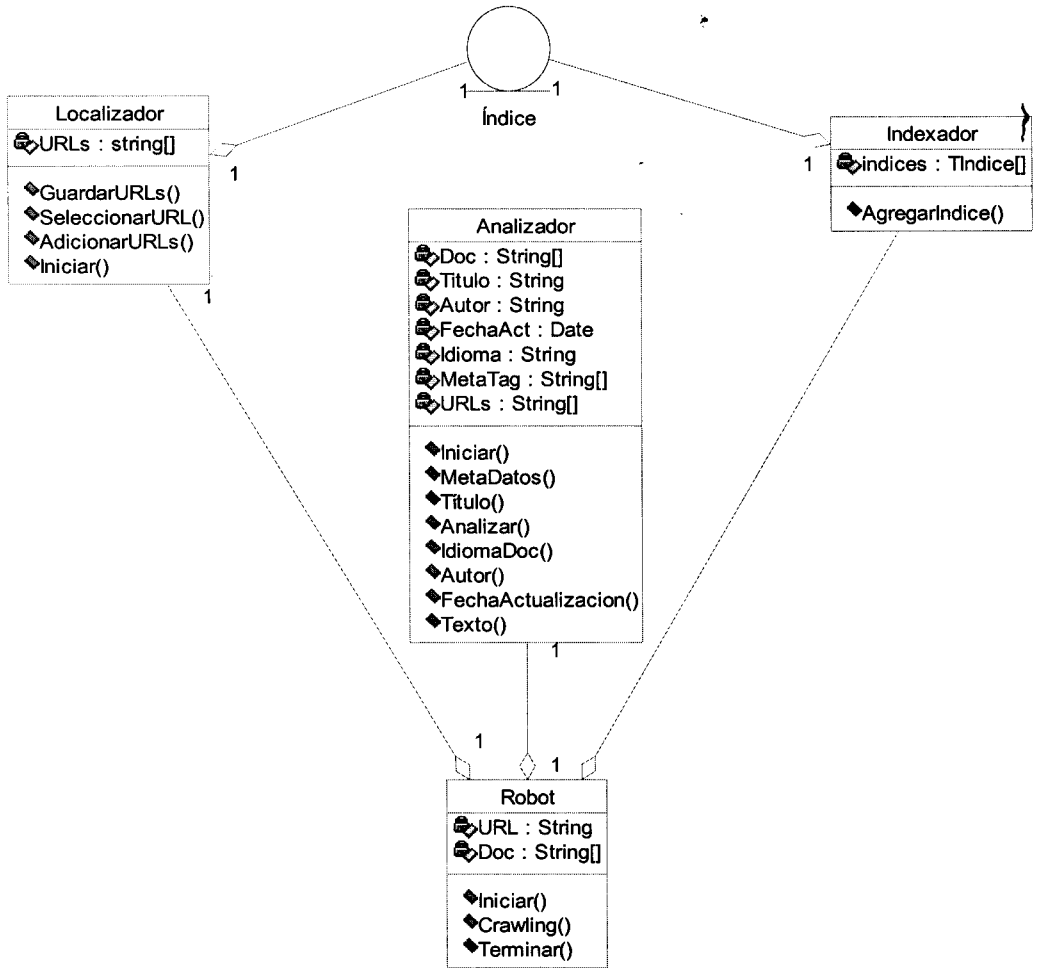


Fig. 2.5 Diagrama de clase.

2.3.5.1 Descripción general de las clases.

Nombre: Robot	
Tipo de clase: Controladora	
Atributo	**Tipo**
URL	String
Doc	String[]

Para cada responsabilidad:	
Nombre:	**Descripción**
Iniciar()	Realiza todo el proceso de *inicialización* del crawler y llama al proceso de inicialización del resto de las clases.
Crawling()	Realiza todas las tareas concernientes a la recuperación de los documentos de Internet o Intranet, su análisis e indexación, es la acción principal.
Terminar()	Detiene la actividad del robot y libera todos los recursos empleados por él.

Nombre: Localizador	
Tipo de clase: Controladora	
Atributo	**Tipo**
URLs	String[]
Para cada responsabilidad:	
Nombre:	**Descripción:**
Iniciar()	Realiza una serie de operaciones de preparación para el proceso de recorrido. Carga los enlaces del Índice y los ordena en una cola de prioridades.
SeleccionarURLs(): String	Selecciona un enlace de la cola de URLs, si el resultado fuera una cadena en blanco, no hay más enlaces para visitar y por tanto, el trabajo del *crawler* ha terminado.
GuardarURLs()	Almacena en el índice la lista de enlaces base que existe en el localizador en ese momento.
AdicionarURLs(String[])	Permite incorporar nuevos enlaces a la cola de enlaces.

Nombre: Analizador	
Tipo de clase Controladora	
Atributo	**Tipo**
Doc	String[]
URLs	String[]
Titulo	String
Autor	String
FechaAct	Date
Idioma	String
MetaTags	Strings[]
Para cada responsabilidad:	

Nombre:	**Descripción:**
Analizar(String[]): boolean	Dado un documento cualquiera el analizador, lo procesa y obtiene de él el texto libre de marcas de formato e información extra sobre el mismo que va a formar parte del índice o los enlaces que aparecen dentro del documento y que serán entregados al localizador.
Titulo : String	Una vez analizado el documento, devuelve el título del mismo si el formato así se lo permite.
Idioma: String	Una vez analizado el documento, devuelve el título del mismo si el formato así se lo permite.
. . .	El resto de las responsabilidades hacen más o menos lo mismo con cada uno de los atributos, después que estos son extraídos con el proceso de Analizar.

Nombre: Indexador	
Tipo de clase Controladora	
Atributo	**Tipo**
índices	TIndices[]
Para cada responsabilidad:	
Nombre:	**Descripción:**
AgregarIndice()	A partir de una serie de parámetros obtenidos por el analizador, construye los nuevos índices que serán incorporados al Directorio.

2.3.6 Sistema de almacenamiento.

Indiscutiblemente es medular la adecuada selección del sistema de almacenamiento del Directorio, que indiscutiblemente puede verse como una base de datos que servirá más tarde para realizar las consultas en el motor de búsqueda. Para el caso específico de nuestro *spider* se siguió el sistema de almacenamiento por índice, más específicamente el de índices o ficheros invertidos.

2.3.6.1 Fichero invertido.

El modelo de fichero invertido que fue tratado en el capítulo dedicado a los aspectos teóricos de la construcción de los elementos de los Sistemas Recuperadores de Información, es el modelo que se empleó para contener el índice elaborado por el robot que nos ocupa. Es una base de datos que se conforma con los registros que elabore el robot para un documento recuperado arbitrario.

Hubo varios criterios que apuntaron hacia los ficheros invertidos, los fundamentales se basan en la simplicidad de la solución, la rapidez de búsqueda que se alcanza y que es para

la mayoría de los motores comerciales un punto común, aunque la información que se almacene difiera de uno a otro SRI.

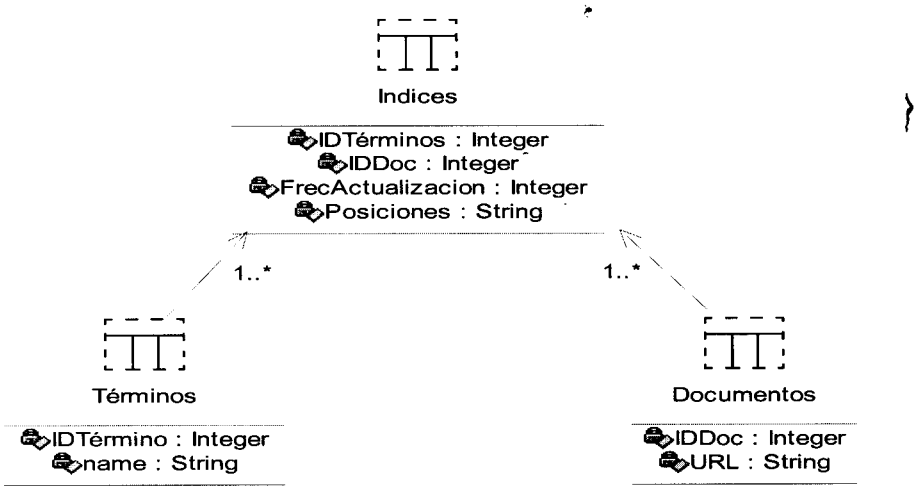


Fig. 2.5 Modelo general de Fichero invertido.

Lo representado en la figura 2.5 es sin dudas una abstracción del problema, pues para el diseño de la base de datos final deberán atenderse las restricciones del modelo de índices invertidos, nombre por el que se le conoce también a este mecanismo de almacenamiento, más las especificidades del modelo teórico que se utilizará para el proceso de RI y otras características adicionales que se prevean en la política de trabajo del Buscador.

En nuestro caso es bueno aclarar que nos acomodamos al modelo vectorial y por tanto, hace falta guardar ciertas informaciones estadísticas aparte, del registro de que si un término aparece o no en el sistema. Algunas de esas informaciones son la frecuencia de aparición del término, las posiciones donde aparece el término, y otros detalles como si el término aparece en el título o no. Por ello, se hace necesario detallar más el modelo antes visto y agregar otras entidades que serán esenciales para el trabajo general del *spider*.

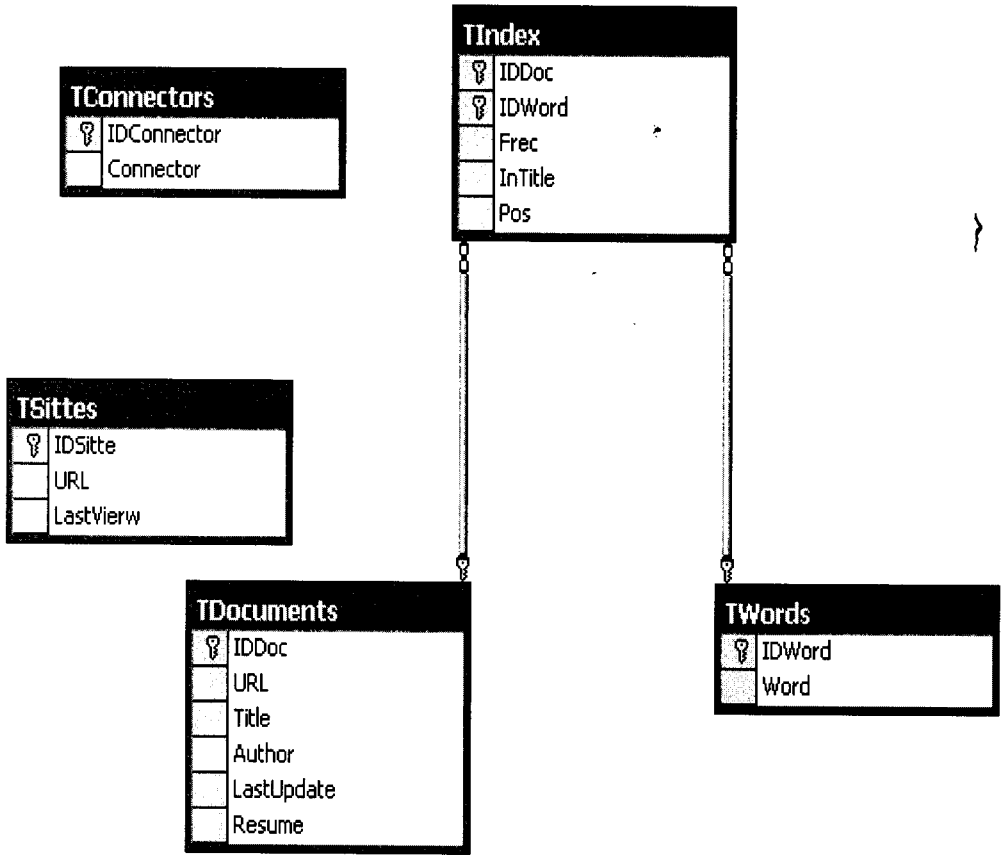


Fig. 2.7 Diagrama de relación de entidades del Índice.

Las mismas pueden describirse de la siguiente manera:

Nombre: TWords		
Descripción: Es la tabla de los términos de indexación. Cada entrada representa un término extraído por el Analizador y procesado por el subsistema de Indexado.		
Atributo	**Tipo**	**Descripción**
IDWord	int	Es el identificador numérico del Término en cuestión, es también la llave.
Word	nvarchar	Es la palabra o término en sí.

Nombre: TConnectors		
Descripción: Es la tabla de las Stop Words, o sea aquellos términos que en dependencia del idioma del documento en cuestión deberán ser eliminados del texto del documento pues las mismas no son representativas para indexar documentos.		
Atributo	**Tipo**	**Descripción**
IDConnector	int	Es el identificador numérico de la Stop Word en cuestión, es también la llave.
Connector	nvarchar	Contiene el Stop Word.

Nombre: TSittes		
Descripción: La presente tabla es el repositorio de los enlaces de partida con que el localizador alimentará el robot y con el que se		
Atributo	**Tipo**	**Descripción**
IDSitte	int	Es el identificador numérico del enlace y a su vez la llave de la tabla.
URL	nvarchar	Es la cadena que describe el enlace
LastView	datetime	Fecha de la última vez que el robot visitó e indexó los documentos en ese enlace.

Nombre: TDocuments		
Descripción: Es el repositorio de los documentos que han sido indexados.		
Atributo	**Tipo**	**Descripción**
IDDoc	int	Este atributo es la llave de la tabla y a su vez es el identificador numérico de los documentos
URL	nvarchar	Es el enlace de documento en Internet.
Title	nvarchar	Es el título extraído del documento
Author	nvarchar	Nombre y Apellidos del o los autores del documento.
LastUpdate	datetime	Representa la última vez que se registró una actualización por parte del analizador
Resume	nvarchar	Es un fragmento o el texto íntegro con detalles de formato para que sea mostrado por el sistema de respuesta a la consulta.

Nombre: TIndex		
Descripción: El la tabla estilo Fichero Invertido que contiene para cada término y el respectivo documento donde aparece una serie de datos útiles para el proceso de recuperación y *ranking*.		
Atributo	**Tipo**	**Descripción**
IDDoc	int	Son las llaves foráneas de las tablas de Documentos y de Palabras.
IDWord	int	
Frec	int	Es la cantidad de veces que aparece el término en el documento.

InTitle	bool	Es verdadero si el término aparece en el Título del Documento
Pos	nvarchar	Es una cadena qué contiene una secuencia de las posiciones relativas en el texto purificado de detalles de formato y metadatos.

De manera, que puede apreciarse en esta última descripción, aparte de lo que es típico de la estructura de Ficheros Invertidos, o sea la determinación de si el término aparece en el documento o no, otras características que apoyarán el trabajo del motor de búsqueda como son la frecuencia, las posiciones, entre otras.

2.4 Características de funcionamiento.

A continuación se expondrá más detalladamente el comportamiento de los subsistemas que integran el robot, que aunque ya se había visto de modo general, ahora se pasa a analizar cuestiones particulares del diseño que se ha tenido en cuenta para desarrollar dichas componentes.

2.4.1 Detalles del comportamiento de los subsistemas del *spider*.

2.4.1.1 El robot.

Como ya habíamos dicho anteriormente, el robot tiene a su cargo obtener de Internet los documentos según los enlaces seleccionados por el localizador, así como respetar la política de exclusión de robots.

Al inicio, parte de una serie de direcciones base que se encuentran en la Base de Datos o Índice y que como ya hemos dicho, normalmente deberán ser sitios que contengan muchos enlaces a nuevos sitios o a páginas del mismo sitio con enlaces a otros sitios. Qué sitio será analizado cada vez lo determina el subsistema de localización, por ello las solicitudes de nuevos enlaces se canalizarán por esa vía. El documento íntegro o ligeramente procesado se entrega al analizador quién lo procesará en busca de enlaces, términos y metadatos. Los primeros serán enviados al localizador para su análisis futuro. Los términos y metadatos

extraídos son procesador por el indexador que posteriormente los almacenará en la base de datos.

Este es, grosso modo, el comportamiento general del robot, pero ya desde el diagrama de interacción veíamos que al ser iniciado el robot se desencadenaban una serie de eventos que iban dirigidos fundamentalmente al Índice y al localizador. Al primero se le vacía para que durante el proceso vuelva a actualizarse; pero esto podría verse de otro modo, ya que se ha tenido la idea de conservar una copia del documento para compararse con el que se acaba de descargar y así determinar si se produjeron cambios o no en el mismo. Esta opción permite que no tenga que procesarse aquellos documentos que no han cambiado desde la última exploración del *crawler* y así ahorrar tiempo. En el caso del segundo, se debe cargar la lista de enlaces base, por lo general consistente en las URL de servidores web, sin embargo trayéndolo a un ámbito más local, o sea la UCI, podría ser la página inicial de la intranet.

Luego se produce el proceso de actualizar la información, durante el cual se exploran los enlaces que provea el localizador en busca de los correspondientes documentos, quien seleccionará las direcciones respondiendo al criterio de recorrido primero a lo ancho, así hasta llegar a un nivel de profundidad que se preestablece general o particular para cada servidor a visitar.

Es válido mencionar que durante la actualización antes de verificar cada servidor se analiza si está en el él el documento "robot.txt", de donde se carga la lista de exclusiones y si alguna dirección servida por el localizador coincide, simplemente se pasa a la próxima para cumplir con el Principio de Exclusión de Robots.

Durante el proceso de actualización pudieran generarse tres procesos que impidan el cumplimiento absoluto de la meta del robot:

- No hay conexión con alguno de los servidores de la cola de direcciones.
- Tiempo de ejecución excedido.
- Fallo del sistema inesperado.

En el primero de los casos, pudiera devolverse al final de la cola de enlaces del localizador y ver si más tarde pudiera continuarse indexado los documentos disponibles en él. Si

sucediera que se termina el tiempo previsto o hay algún fallo del sistema donde está corriendo el robot; en el primer caso se devuelve el control al sistema con el mensaje de error correspondiente y en el segundo normalmente no podrá hacerse nada sino volver a arrancar la actualización del Directorio.

Siempre puede saberse si hubo cambios, motivos de actualización en el Directorio, comparando la cola de enlaces del localizador contra la copia que quedó en la base de datos. Por dichos cambios y verificando que se llegó al final de la cola puede darse por concluido el trabajo del robot.

Se ha habilitado un modo de configuración mediante el cual al robot se le especifica el nivel de profundidad máximo a alcanzar en cada servidor y eliminar, actualizar o insertar enlaces en la tabla de enlaces de partida.

2.4.1.2 El localizador.

La función primordial del localizador es la de servir, según un criterio de selección predeterminado, los enlaces al robot, así como aceptar los enlaces que extraiga el analizador de los documentos.

Durante el proceso de *inicialización*, el localizador carga de la base de datos la lista de partida guardada la última vez, y se la va sirviendo al robot para que este haga sus solicitudes, si la dirección es válida, se mantiene, si por algún motivo ha desaparecido y después de una cantidad de exploraciones totales, sigue sin posible conexión con el sitio, se elimina el enlace de la tabla de enlaces iniciales.

Al agregársele un posible nuevo enlace, el localizador deberá comprobar que no haya sido visitado, de ser así se coloca no necesariamente al final de la cola, sino atendiendo al criterio de recorrer cada sitio por el método de primero a lo ancho. Así que él localizador encontrará el lugar que debe tener el URL en la cola y lo inserta allí.

El proceso de selección con semejante criterio de ordenamiento puesto, es muy sencillo, consistente solo en tomar el primero que esté en cola y desplazar la marca de "exploración" un lugar más arriba. Cuando dicha marca haya abarcado todos los elementos, se habrá recorrido toda la Web indexable por nuestro *spider*. Esta sería la condición de parada con éxito.

2.4.1.3 El analizador.

Es en este subsistema donde recae la mayor parte del peso del trabajo, pues como veremos a continuación acá es donde deben implementarse los métodos más engorrosos: procesamiento de texto natural, el parseo de los distintos tipos de documentos, la determinación del idioma del documento, la extracción de los enlaces, y otras muchas operaciones que representan una parte importante del *performance* de la aplicación en general.

Podemos distinguir con facilidad algunas etapas dentro de la acción del subsistema:

- Determinación del formato del documento.
- Parseo.
- Análisis del idioma del documento.
- Añadir los enlaces extraídos durante el parseo a la cola de enlaces del localizador.
- Extraer los términos y calcular los datos estadísticos que intervendrán en la formación del registro de Fichero.

La determinación del formato del documento y por tanto, su respectivo mecanismo de análisis sintáctico es vital una vez obtenido el documento, una solución simplista pudiera ser discriminar por la extensión del archivo: txt, html o htm, pdf, doc, shtml, xml, etc. Esto es válido si siempre se respeta dicho protocolo, de lo contrario habría que establecer un mecanismo para revisar una parte del documento hasta que coincida y pueda ser parseado con uno de los analizadores sintácticos, de lo contrario el documento se desecha por no ser "legible".

Después o como parte misma del proceso anterior, vendría el parseo, que acepta un documento, lo procesa y devuelve tres cosas: los enlaces contenidos en el documento, los metadatos y el texto puro. Como parte de este proceso pueden también eliminarse las *stopwords* del texto puro.

¿Cómo se realizaría el análisis del documento?

En caso de que el documento sea tipo Word hará falta un analizador de ficheros Word, pero la idea más ilustrativa y simple sería simplemente analizar un documento HTML, pues son los más abundantes y es el lenguaje por excelencia de Internet.

Primero se determina si es una página con marcos pues entonces el resultado del texto vendría a ser la unión de todos los textos de los documentos formados por los marcos por lo que si tenemos una estructura de marcos como la siguiente:

<frameset rows = "30%, 30 %,">*

<frame href= "1.html">

<frame href= "2.html">

<frame href= "3.html">

</frameset>

Deberá procesar cada una de los documentos y unir el resultado como uno solo.

Posteriormente deberá extraer la información relacionada con el título, esto es algo muy sencillo pues basta o bien con buscar el texto entre las etiquetas *<title>...</title>*, o en caso de no aparecer, simplemente se tomarán los primeros n caracteres alfanuméricos que aparezcan en el cuerpo del documento.

Un tercer paso del parseo lo constituye la extracción de los metadatos que normalmente viene contenidos en las etiquetas:

<meta name= "nombre de la etiqueta" content = "contenido de metadato">

Luego se procede a eliminar todas las etiquetas y los signos de puntuación y sustituir los símbolos descritos al estilo HTML ("á" en lugar de "á") para dejar solamente el texto puro.

En los casos de otros tipos de documentos (por ejemplo .doc o .pdf) simplemente se procesan con un convertidor de esos formatos a texto plano y luego se depuran de signos de puntación.

Luego vendría el paso de determinar o mejor, estimar en qué lenguaje mayoritariamente se escribió el documento.

Para ello sería bueno antes aclarar que el idioma de un texto determinado puede determinarse con bastante regularidad mediante el uso de trigramas, que son conjuntos cuyos elementos son tríadas de caracteres consecutivos y que permiten por comparación determinar si un texto a partir de sus trigramas pertenece a una lengua o no.

Supongamos que se tienen almacenados en un fichero los trigramas T_1 y T_2 correspondientes a dos idiomas arbitrarios, luego se extraen los trigramas T del documento que estamos analizando y se comparan los resultados de

$$f(T, T_1) \text{ y } f(T, T_2)$$

donde $f(T, I)$ es la función que me calcula el nivel de semejanza entre los trigramas del documento T y los del idioma I y se define como:

$$f(T, I)=2\frac{|T \cap I|}{|T|+|I|} \text{ donde } f \text{ toma valores entre } 0 \text{ y } 1$$

y se comparan si

$$f(T, T_1) > f(T, T_2)$$

se infiere que el documento está escrito en el primer idioma; de lo contrario, es de suponer que está en el segundo. Lo ideal sería contar con toda una serie de trigramas que permitan valorar entre varios idiomas y para ello, una vez obtenido el vector los valores de $f(T, T_i)$ se ordenan y se toma el de mayor valor. Si coincidieran varios, simplemente no se escoge ningún idioma pues no puede inferirse a partir de la información extraída del texto.

Posteriormente, se envía la lista de enlaces al localizador para que este lo incorpore a la cola de enlaces a visitar. De ahí, se pasa a enviar al indexador la información extraída del documento: título, url, autor, fecha de actualización, texto.

2.4.1.4 El indexador.

Es el indexador el encargado de que, una vez le sean entregado los datos de salida del analizador, conformar con ellos los distintos registros que serán incorporados al índice. Uno de los pasos a ejecutar es la eliminación de las *stopwords* y las palabras que queden serán los términos que habrán de emplearse para la fase de construcción de los registros del Índice.

En este subsistema se calcula la frecuencia de aparición de un término, y las posiciones relativas (debe recordarse que las *stopwords* fueron sesgadas y por tanto, solo puede ser estimado la cantidad de estos términos poco significativos había entre dos términos definitivos cualesquiera) dentro del texto depurado. Una vez hechos estos cálculos las palabras se van eliminando del texto.

El indexador actualiza posteriormente el Directorio Virtual con los registros formados y responde al robot que el almacenamiento de los registros se realizó de modo exitoso, para que aquel sepa que debe continuar con él próximo documento. De suceder algún tipo de imposibilidad de conexión sería muy útil almacenar localmente esos registros hasta su posible incorporación en la tabla.

2.1.4.5 El subsistema de almacenamiento o Índice.

Con el Índice solo actúan directamente el localizador y el indexador. El primero, para extraer la lista de enlaces base a visitar y para agregar nuevas URLs de partida al sistema; toda vez que el segundo inserta o actualiza los registros de índice invertido del Directorio.

Sin embargo, el directorio no sería tal si no se implementan una serie de facilidades para que se produzca un intercambio dinámico entre el robot, que muy probablemente corra en una máquina distinta a donde se encuentra la base de datos y su gestor.

Podríamos decir que para ello sería muy útil la creación de una doble capa de datos, una para que suponiendo que al gestor puede accederse en la misma intranet y una segunda posibilidad para que pueda accederse como un Web Service y que tanto el robot como el directorio puedan estar en lugares geográficos distintos, de modo que pueda configurarse de dónde va a leerse la información y el sistema cambiará al mecanismo de conexión seleccionada.

Conclusiones

En vista a los objetivos propuestos a lo largo del siguiente informe, se puede concluir que el presente trabajo de diploma intentó más que dar una solución a la problemática de construir un Directorio Virtual y un *spider* que lo mantuviera actualizado con los índices extraídos de los documentos recuperados de la intranet de la UCI y en una futura expansión, posterior a un estudio más ambicioso, a formar parte del buscador CubaSearch; se abrió las puertas con análisis, estudios, etc., para acometer en un futuro mediato las tareas de poner en punta ambas herramientas y comprobar el alcance de los resultados.

El punto de producción alcanzado impidió la conclusión completa de los objetivos, por lo que se presenta un pequeño prototipo centralizado que permite:

- Recorrer la Intranet primero a lo ancho con un nivel preestablecido de profundidad límite.
- Formar la base de datos de ficheros invertidos solo con las palabras extraídas de los documentos, la frecuencia de aparición en los documentos y la posición relativa donde se encuentra en el texto purificado.
- Agregar o eliminar uno de los enlaces a visitar la próxima vez que el robot se active.
- Activar un servicio que dependa del reloj del sistema para que de modo automático se inicie el proceso de indexación.

Recomendaciones

Es propósito de este acápite, resaltar algunas sugerencias y recomendaciones para la futura continuación del presente trabajo, que es sin dudas el objetivo fundamental de la Universidad de las Ciencias Informáticas y la división de CubaSí de ETECSA.

- Avanzar en el estudio real de las posibilidades de C#, MS SQL Server y .NET para la elaboración de un SRI de uso comercial.
- Completar los procedimientos y mejorar la velocidad del *crawling*, manteniendo la arquitectura centralizada pero configurando de modo cooperativo varios robots para indexar la Web.
- Implementar el estimador de idiomas y si es posible agregar la mayor cantidad de estos al sistema, para facilitar el trabajo del indexador y del motor que deberá recuperar documentos del Directorio.
- Robustecer al indexador con mecanismos más complejos, asistiéndose del tratamiento de lenguaje natural, entiéndase Análisis textual, ***Segmentación del texto, Filtrado de información no relevante,*** Análisis morfosintáctico, **Desambiguación morfosintáctica (*pos tagging*), Agrupación sintáctica (*chunking*)**, diccionarios y otros que puedan arrojar luz sobre el asunto..

Referencias Bibliográficas

[Ana02] Anaya, Henry y Trujillo Rasúa, Rafael Arturo. Trabajo de Diploma "Un sistema de recuperación de información para una intranet". Universidad de Oriente, Santiago de Cuba, 2002.

[Bae99] Baeza-Yates, R. y Ribeiro-Neto, B. "*Modern Information Retrieval*". Addison-Wesley, 1999.

[Bae02] Baeza-Yates, Ricardo y Navarro, Gonzalo. "Recuperación de la Información: Algoritmos, estructuras de datos, y aplicaciones en el Web". Addison-Wesley, 2002.

[Codi--] Codina, L. "Teoría de recuperación de información: modelos fundamentales y aplicación a la gestión documental".

[Jav--] Javocho. "Diseño e implementación de una Arquitectura multiplataforma para el estudio de Motores de búsqueda en Internet".

[01] Ficha técnica de Google

http://www.infobuscadores.com/google_02.htm (18/03/04).

[02] ¿Por qué usar Google?

http://www.google.com/intl/es/why_use.html (19/03/04).

[03] Ficha técnica de Alltheweb

http://www.infobuscadores.com/alltheweb_02.htm (18/03/04).

[04] Frakes, W.B.; Baeza-Yates, R. "*Information Retrieval: Data Structures and Algorithms*". Prentice-Hall. Englewood Cliffs, N.J. 1992.

[05] Greengrass, Ed. "*Information Retrieval: a Survey*". November, 2000.

[06] Search Engines Watch

http://www.searchenginewatch.com/

[07] González Seco, José Antonio "*El lenguaje de programación C#*", 2001.

[Kost94] Koster, Martijn. *"A Standard for Robot Exclusion"*. 1994. http://info.webcrawler.com/mak/projects/robots/norobots.html

[08] Parihar, Mridula. *"La Biblia de ASP .NET"*. Anaya Multimedia. 2002.

[Pons00] Pons Porrata, Aurora. "Agrupamiento de documentos", Universidad de Oriente, Santiago de Cuba, 2000.

[Vill98] Villena Román, Julio. *Sistemas de Recuperación de Información*. Curso de Doctorado. Departamento de Ingeniería de Sistemas Telemáticos. U.P.M. 1998.

Bibliografía

1. Anaya, Henry y Trujillo Rasúa, Rafael Arturo. Trabajo de Diploma "Un sistema de recuperación de información para una intranet". Universidad de Oriente, Santiago de Cuba, 2002.
2. Baeza-Yates, Ricardo y Navarro, Gonzalo. "Recuperación de la Información: Algoritmos, estructuras de datos, y aplicaciones en el Web". Addison-Wesley, 2002.
3. Baeza-Yates, R. y Ribeiro-Neto, B. "*Modern Information Retrieval*". Addison-Wesley, 1999.
4. Greengrass, Ed. "Information Retrieval: A Survay", November 2002.
5. Ingwersen, Peter. "Information Retrieval Interaction", First Edition. Taylor Graham Editing, 1992.
6. Pons Porrata, Aurora. "Agrupamiento de documentos", Universidad de Oriente, Santiago de Cuba, 2000.
7. Pons Porrata, Aurora. "*Folleto del Curso de Maestría*". Universidad de Oriente. 2001.
8. Serler, E. and Etzioni, O. "Multi-search and comparison using the MetaCrawler", Department of Computer Sciences and Engineering, University of Washinton, October, 1995.
9. Rodríguez, Horacio. "Técnicas básicas en el tratamiento informático de la lengua", UNAM, 2000.
10. Villena Román, J. "Sistemas de Recuperación de Información", Curso de Maestría y Doctorado, Departamento de Ingeniería de Sistemas Telemáticos. U.P.M. 1998.