





# **Búsqueda eficaz de información en la Web**



**José Angel Olivas Varela**

# **Búsqueda eficaz de información en la Web**



**cacic2011** XVII CONGRESO ARGENTINO DE CIENCIA DE LA COMPUTACIÓN

**XV ESCUELA INTERNACIONAL DE INFORMATICA**



FACULTAD DE INFORMÁTICA  
Universidad Nacional de La Plata



RedUNCI



Olivas, José A.  
Búsqueda eficaz de información en la web. - 1a ed. -  
La Plata : Universidad Nacional de La Plata, 2011.  
130 p. ; 24x16 cm.

ISBN 978-950-34-0763-9

1. Informática. 2. Recuperación de Información. I.  
Título  
CDD 005.3

## **Búsqueda eficaz de información en la Web**

**José Angel Olivas Varela**

Diseño y diagramación: Andrea López Osornio  
Cuidado de la edición: Magdalena Sanguinetti



Editorial de la Universidad Nacional de La Plata (Edulp)  
47 N° 380 / La Plata B1900AJP / Buenos Aires, Argentina  
+54 221 427 3992 / 427 4898  
editorial@editorial.unlp.edu.ar  
www.editorial.unlp.edu.ar

Edulp integra la Red de Editoriales Universitarias (REUN)

Primera edición, 2011  
ISBN N° 978-950-34-0763-9

Queda hecho el depósito que marca la Ley 11.723  
©2011 - Edulp  
Impreso en Argentina

A mi madre, e *in memoriam* a mi padre,  
le hubiera gustado leerlo.

*«Existing search engines —with Google at the top— have many remarkable capabilities; but what is not among them is deduction capability, the capability to synthesize an answer to a query from bodies of information which reside in various parts of the knowledge base».*

LOTFI A. ZADEH



## **Agradecimientos**

Mi más sincero agradecimiento a los miembros del grupo de investigación SMILe (*Soft Management of Internet and Learning*) de la Universidad de Castilla-La Mancha (España) por su amistad, apoyo y contribución al desarrollo de todas las investigaciones en el ámbito del libro. En particular a Javier de la Mata, por las aportaciones en el tema de Recuperación de Información, a Manuel Fernández en la parte de operadores y a Francisco P. Romero, Jesús Serrano, Cristina Puente y Andrés Soto en lo que concierne a nuevas propuestas. También quiero agradecer a mis allegados, en especial a G. Bustarviejo, su paciencia y apoyo.



# Índice

Prólogo	13
Capítulo I: Sistemas de recuperación de información	15
Recuperar información no es recuperar datos	16
Etapas en el proceso de recuperación de información	17
Indexación	18
Preprocesado de documentos	19
Estructuras de indexación clásicas	22
Consulta	23
Evaluación	25
PageRank	26
HITS (Hypertext Induced Topic Selection)	27
SALSA (Stochastic Approach for Link Structure Analysis)	28
Capítulo II: Herramientas de búsqueda en la Web:	
Directorios, buscadores y metabuscadores	29
Breve historia de las herramientas de búsqueda en la Web	29
Directorios	31
Buscadores Web	32
Ventajas e inconvenientes de los buscadores Web	32
Operadores de búsqueda	35
Operadores lógicos	36
Otros operadores	37
Arquitectura de los buscadores	46
Metabuscadores	51
Agentes inteligentes	57
Posicionamiento Web-SEO (Search Engine Optimization)	59
Capítulo III: El lenguaje y los usuarios en los buscadores Web	61
Semántica	62

Capítulo IV: Las técnicas de Soft Computing en la recuperación de información	77
El papel de la lógica borrosa en los Metabuscadores	82
El uso de diccionarios de sinónimos y tesauros (ontologías): búsqueda conceptual	82
Sentencias de búsqueda y capacidades deductivas	84
Combinación de valores borrosos	86
Resultados del proceso de <i>clustering</i> borroso	87
Arquitectura de un Metabuscador	88
Capítulo V: Aportaciones y Propuestas	91
Representación de documentos mediante el modelo FIS-CRM	91
Integración del modelo FIS-CRM en sistemas de búsqueda	93
Ejemplo de búsqueda con el modelo FIS-CRM	94
GUMSe: Metabuscador basado en agentes	96
Cálculo de distancias borrosas entre los términos de una ontología	100
FzMail: Una Herramienta para la gestión inteligente del correo electrónico	103
BUDI: Plataforma de Metabúsqueda para la búsqueda difusa de información	105
FOG: Generación automática de Ontologías Borrosas	108
Estudio y uso de las relaciones causales	111
Ejemplo	114
Conclusiones y líneas de actuación	117
Bibliografía	119

## Prólogo

En los últimos años, en especial desde la aparición de los computadores personales, la cantidad de información digital generada y almacenada ha crecido de forma vertiginosa, alcanzando niveles jamás conseguidos con anterioridad. Con la aparición de Internet, y en particular con el uso generalizado de las páginas Web, se abre la posibilidad de acceder (habitualmente de forma gratuita) a toda esa información generada, que puede ser vista como una base de datos distribuida, dónde la mayoría de la información se almacena en forma de texto (esto está cambiando rápidamente hacia la información multimedia), en multitud de lenguas y formatos, y que hoy por hoy ya tiene un tamaño inmanejable en su totalidad. Diversas estimaciones fiables indican que en la última década es accesible en la Web un volumen de datos mayor a toda la información generada por la humanidad desde la antigüedad hasta el final del siglo xx.

Gestionar toda esta información representa un gran desafío científico y tecnológico en muchos aspectos, en particular para manipular, buscar y recuperar la información y el conocimiento contenido, teniendo en cuenta que los datos que provienen de la Web no sólo contemplan el contenido de las páginas y los enlaces entre ellas, sino que también pueden incluir ficheros (*logs*) sobre el uso de la mismas. Se podría decir entonces que la Web es una estructura de grafo (enlaces) con un nivel de conectividad muy alto, diferentes “categorías” de nodos con contenidos muy heterogéneos y poco estructurados (texto, multimedia, etcétera), donde resulta impracticable cualquier intento de estandarización masivo y cuyos datos lejos de ser estáticos, son tremendamente cambiantes en el tiempo. Los motores de búsqueda comerciales actuales (con Google a la cabeza) suelen ser muy eficientes, pero los resultados que proporcionan no suelen ser plenamente satisfactorios (relevantes) para los usuarios (demasiada información no buscada o errónea, falta de información relacionada recuperada).

Además, recientemente se ha generalizado el uso de las denominadas “redes sociales” (Facebook...), que permiten el envío de mensajes, los

foros de opinión, compartir información multimedia, realización de actividades en grupo, etcétera, que deben cambiar radicalmente la filosofía de los sistemas de búsqueda y acceso a la información/conocimiento/opiniones/comportamientos contenidos, ya que no sólo se manejan contenidos “descriptivos” (como es habitual en la Web), sino que las “conversaciones”, comportamientos y opiniones almacenadas, gestionadas de forma “inteligente” pueden suministrar un conocimiento muy valioso en cuanto a tendencias, intenciones, aspiraciones... de la sociedad en su conjunto o de determinados ámbitos del comportamiento social.

El procesamiento inteligente del lenguaje natural juega un papel primordial en la mejora de la eficacia en el uso de las herramientas disponibles para el acceso y la búsqueda de información en la Web, tanto desde el punto de vista del propio usuario como del de las propias herramientas, que de una forma automática o semi-automática pueden proporcionar ayudas muy apreciables (por ejemplo usando sinónimos a las palabras contenidas en nuestra petición de búsqueda) ya que habitualmente los sistemas comerciales de búsqueda sólo contemplan aspectos lexicográficos en el lenguaje (tanto en el de los documentos almacenados como en el de las propias consultas) y se olvidan de sus aspectos semánticos.

Por todo lo anteriormente reflejado, en este trabajo se describe someramente lo que es un Sistema de Recuperación de Información, para posteriormente poder profundizar en algunos aspectos específicos. Se presentan las herramientas de búsqueda Web más usadas actualmente, haciendo especial hincapié en los buscadores y en los metabuscadores, con el fin de proporcionar ciertos “trucos” para ayudar a mejorar nuestro acceso y búsqueda en los contenidos de la Web (por ejemplo explicando el uso de algunos operadores de búsqueda, cómo funcionan los algoritmos de ranking, como mejorar la posición de una página Web en los buscadores o cuáles son las peculiaridades de las arquitecturas computacionales de algunos motores de búsqueda). Finalmente, se propone el desarrollo y pruebas de mecanismos más “inteligentes” de acceso, búsqueda, gestión y recuperación de información y conocimiento contenidos en la Web. Para ello se muestra el uso de técnicas avanzadas de Inteligencia Artificial, en particular aquellas más cercanas a la manipulación del lenguaje natural y al comportamiento humano.

## CAPÍTULO 1

# Sistemas de recuperación de información

Habitualmente, un sistema de recuperación de información es definido como el proceso que trata la representación, almacenamiento, organización y acceso de elementos de información (Salton, 83). Es decir, es un sistema capaz de almacenar, recuperar y mantener información (Kowalsky, 97).

Pero podríamos plantearnos qué representa el concepto de información en este contexto. Se entiende por información cualquier elemento susceptible de ser recuperado, lo que incluye principalmente texto (incluidos números y fechas), imágenes, audio, video y otros objetos multimedia (Kowalsky, 97). Pero el tipo principal de objeto recuperable, hasta el momento siempre ha sido el texto, motivado especialmente por su facilidad de manipulación en comparación con los objetos multimedia, especialmente en lo que se refiere a capacidad de cómputo. Actualmente están surgiendo muchos sistemas que tratan de gestionar este tipo de objetos (diversos buscadores comerciales incluyen buscadores de imágenes), aunque de momento simplemente buscan en el texto de las etiquetas de dichos objetos multimedia, sin escudriñar realmente su contenido interno, lo que suele dar frecuentemente origen a engaños o falsos etiquetados.

En los sistemas de recuperación de información no se suele trabajar directamente con los documentos de texto sino con representaciones más estructuradas de los mismos. La forma de representar los documentos determina en gran medida las características del resto de elementos del sistema. Los modelos de representación de documentos clásicos se basan generalmente en el modelo *booleano* o en el modelo *vectorial*. En el primero, cada documento es representado por un vector donde cada posición se corresponde con cada uno de los términos susceptibles de aparecer en el documento y el valor de cada posición será 0 ó 1 según ese término aparezca o no en ese documento. La esencia del modelo *vectorial* es similar, salvo que el contenido de cada componente representa algún valor que tiene que ver con la frecuencia de aparición de ese término en el documento.

Claramente, estos modelos de representación de documentos son adecuados para documentos de texto, que pueden ser por ejemplo páginas Web u otros objetos (como elementos multimedia) que estén descritos de forma textual.

Quizá el concepto más importante en recuperación de información es el de *relevancia*. Tiene que ver con cómo medir la satisfacción de un usuario con los resultados devueltos por el sistema ante una determinada pregunta (*query*). Esta medida es claramente subjetiva, ya que ante una misma *query* y el mismo resultado (documento o lista ordenada de documentos), la relevancia puede ser totalmente distinta para dos usuarios diferentes, e imposible de medir de forma precisa. Esta es una de las razones por las que cada vez se tiene más en cuenta el papel del usuario en los sistemas de recuperación de información: si se conocen los intereses de los usuarios el sistema puede “guiar” la búsqueda de información hacia los mismos.

Un ejemplo: Supongamos que dos usuarios diferentes (U1 y U2) introducen la consulta “monitor barato” en un buscador Web comercial. Si U1 habitualmente hace búsquedas en páginas de gimnasios y deportes y U2 lo hace en páginas de productos informáticos, lo más probable es que U1 esté buscando entrenadores baratos y U2 pantallas de ordenador baratas. Si se hubiesen “almacenado” de alguna forma estos “perfiles de usuario”, estas dos búsquedas podrían haber sido guiadas de formas totalmente diferentes y la relevancia de los resultados para cada usuario hubiera aumentado significativamente. Además, este ejemplo pone de manifiesto uno de los principales problemas de la recuperación de información, que es la propia complejidad del lenguaje natural, eje central de este trabajo. La palabra “monitor” es polisémica, lo que dificulta enormemente la tarea de recuperación de información cuando es usada.

## **Recuperar información no es recuperar datos**

Cuando accedemos a una base de datos, por ejemplo la de una biblioteca, usamos un lenguaje muy estructurado y con una semántica muy precisa. Si buscamos libros de Lope de Vega, lo pondremos en el campo autor de la ficha que nos proporcione el sistema, y éste nunca tendrá en cuenta la acepción que de “vega” que tiene que ver con el paso de un río. El sistema tratará de recuperar aquellos libros de su base de datos cuyo autor es Lope de Vega. Pero si entre sus más de mil obras queremos localizar aquéllas que contengan la palabra “rimas” en su título, el sistema puede transformar nuestra pregunta en una sentencia precisa de un lenguaje que entienda la base de datos



(por ejemplo *SQL: Structured Query Language*), y que represente algo como “busca las obras de nuestra base de datos cuyo autor es Lope de Vega y cuyo título contenga la palabra rimas”. Con esta especificación, el sistema podría recuperar por ejemplo las obras *La Circe con otras rimas y prosas* (1624) y *Rimas* (poesías, 1604). Por el contrario, en un sistema de recuperación de información, el objeto recuperado no tiene porqué adaptarse de forma exacta a las peticiones de búsqueda. La razón fundamental es que la información que gestiona un sistema de recuperación de información está en lenguaje natural, sin estructurar, por lo que puede ser semánticamente ambigua.

## **Etapas en el proceso de recuperación de información**

Para un usuario, el proceso de recuperación de información consiste en realizar una pregunta al sistema y obtener como respuesta un conjunto de documentos ordenados. Pero en todo sistema de recuperación de información es necesaria la realización de una serie de pasos previos y diferenciados para poder llegar a sus respuestas, los más relevantes son:

1. *Indexación*: El sistema de recuperación de información crea un índice que contiene los términos que el sistema considera importantes (después de un preprocesado de cada documento) y su ubicación en los documentos.
2. *Consulta*: El usuario formula una pregunta al sistema, en un lenguaje (formalismo) procesable por éste.
3. *Evaluación*: El sistema devuelve los resultados (documentos que satisfacen en cierto grado la demanda de información del usuario), ordenados según su (posible) relevancia con respecto a la consulta formulada.
4. *Retroalimentación del usuario* (opcional): El sistema aprende de las diferentes consultas de un usuario, focalizando la recuperación según este conocimiento adquirido.

Estos procesos suelen ser estándar en todos los sistemas de recuperación de información. Además esta clasificación no es cerrada, sino que dependiendo del sistema de recuperación pueden ser ejecutados otros nuevos métodos diferentes, por ejemplo los sistemas que utilizan estructuras de conocimiento adicionales a los índices de términos clásicos suelen tener procesos adicionales encargados de construir, mantener o actualizar dichas estructuras. Los sistemas de recuperación de información que se basan en el uso de perfiles de usuario pueden realizar una nueva etapa para la construcción y actualización del perfil de usuario (almacenando los términos que

representan las preferencias de los usuarios con la finalidad de mejorar el comportamiento del sistema en futuras consultas).

Hay varias propuestas formales para definir un modelo de sistema de recuperación de información, pero una de las más utilizadas es la de Baeza-Yates (1999), que define un sistema de este tipo como una cuádrupla  $[D, Q, F, R(q_i, d_j)]$  donde:

- D es un conjunto de vistas lógicas (o representaciones) de los documentos que forman la colección.
- Q es un conjunto compuesto por vistas lógicas (o representaciones) de las necesidades de información de los usuarios. Estas vistas se denominan consultas (*queries*).
- F es una forma de modelar la representación de los documentos, consultas y sus relaciones.
- $R(q_i, d_j)$  (*ranking*) es una función de evaluación que asigna un número real al par formado por una consulta  $q_i \in Q$  y la representación de un documento  $d_j \in D$ . Este valor determinará el orden de aparición de los documentos de una consulta  $q_i$ .

Todos estos elementos se verán reflejados en las etapas en el proceso de recuperación de información, que se detallan a continuación.

## Indexación

Para conseguir D (conjunto de vistas lógicas o representaciones de los documentos que forman la colección) es necesaria la construcción de una base documental que contenga la información de los objetos que el sistema es capaz de escudriñar para poder llevar a cabo el proceso de recuperación. Si un objeto no está en esta base de datos no podrá ser recuperado. Pero lo que maneja realmente el sistema no son los propios documentos susceptibles de ser recuperados sino una representación de los mismos. Estos documentos se representan con algún formalismo, habitualmente creando un índice con el conjunto de términos significativos que aparecen en los documentos, que suele ser un subconjunto de todos los términos de los documentos. También es necesario que el sistema de recuperación de información disponga de mecanismos que permitan introducir un nuevo objeto en la base documental, o bien utilizan mecanismos automáticos que se encargan de explorar el espacio de búsqueda (en nuestro caso la Web) añadiendo al índice la información sobre los términos que aparecen en estos nuevos documentos. Esto es necesario entre otras cosas porque los mecanismos de ordenación por relevancia para el usuario (se explicarán con detalle más adelante) suelen seleccionar como

documentos más relevantes, entre otros criterios, aquellos que posean con más frecuencia o en determinada posición los términos que están en la consulta del usuario.

La opción más simple sería almacenar los documentos y buscar en cada uno de ellos la existencia o no de los términos, pero este planteamiento es inviable computacionalmente debido a la enorme cantidad de información que sería necesario manejar, lo que imposibilitaría que el sistema fuese eficiente. Por tanto, es necesaria la utilización de estructuras que almacenen información sobre los documentos y que permitan realizar las búsquedas en tiempos razonablemente cortos. Estas estructuras es lo que hoy en día se denominan índices. Pero habitualmente los documentos suelen ser preprocesados antes de ser indexados para reducir el número de elementos (términos, signos de puntuación, ubicación de los términos...) a tener en cuenta y por tanto mejorar la eficiencia del proceso de recuperación. Está claro que esta reducción de elementos supone una pérdida de información, que puede ser muy importante a la hora de buscar los documentos más relevantes ante una consulta, por lo que el secreto del éxito en esta etapa radica en encontrar el equilibrio justo entre la eliminación de elementos a indexar de los documentos y la eficiencia de los procesos de búsqueda en este índice.

### ***Preprocesado de documentos***

Las tareas que se utilizan habitualmente en el preprocesado de documentos para su indexación son las siguientes:

- *Eliminación de signos de puntuación*: se eliminan los acentos, comas, puntos y demás signos de puntuación con el fin de tratar los términos de forma uniforme. Este proceso tiene los inconvenientes de que se pierde esta información y no se podrán utilizar signos de puntuación en las consultas de los usuarios y que los signos de puntuación poseen información semántica importante. Por ejemplo:

Documento original: “...vivía cerca de León. Furioso con lo que le rodeaba, los hombres de la comarca lo odiaban...”

Documento sin signos de puntuación: “...vivía cerca de león furioso con lo que le rodeaba los hombres de la comarca lo odiaban...”

- *Eliminación de palabras prohibidas (stop words)*: en todos los idiomas hay un conjunto de palabras muy frecuentes que se usan por cuestiones lingüísticas de concordancia sintáctica entre palabras y frases. Si se considera que estas palabras no aportan

ningún significado a un documento (cosa que no es cierta), sino que sólo se utilizan para seguir las reglas del idioma, podrían ser eliminadas. Por ejemplo en español podrían eliminarse artículos, preposiciones, conjunciones, algunos adverbios, etc. Existen listas para los diferentes idiomas (denominadas *stoplists*) con estas palabras, que sirven como referencia para no tener en cuenta estas palabras cuando aparezcan en los documentos a la hora de ser indexados. También es frecuente la construcción dinámica de estas listas cuando se desarrollan sistemas de recuperación de información. En el ejemplo se puede ver claramente cómo cambia la semántica del texto tras eliminar estas palabras:

Documento original: “...vivía cerca de León. Furioso con lo que le rodeaba, los hombres de la comarca lo odiaban...”

Documento sin signos de puntuación y sin palabras prohibidas: “...vivía león furioso rodeaba hombres comarca odiaban...”

- *Stemming o lematización*: Consiste en obtener la raíz léxica (o *stem*) de una palabra. Normalmente se ignoran las diferentes variaciones morfológicas que puede tener a la hora de indexar. En la mayoría de los casos la raíz es una palabra sin significado, como por ejemplo en el caso de “vivía” y “vivencia”, cuya raíz común sería “viv”. Normalmente este mecanismo se suele aplicar para eliminar el sufijo de una palabra, pero no se aplica al prefijo. Esto se debe a la idea de que la raíz contiene la fuerza semántica de la palabra, y que los sufijos introducen ligeras modificaciones del concepto o tienen meramente funciones sintácticas. El objetivo inicial de este proceso de lematización fue mejorar el rendimiento de los sistemas de recuperación de información al reducir el número de palabras que un sistema tenía que almacenar en el índice. Otra de las características de la lematización es que frecuentemente favorece la exhaustividad (*recall*) en la búsqueda, es decir, se recuperan más palabras relacionadas léxicamente al tener la misma raíz y por tanto se obtiene un conjunto más elevado de términos, evitándose así perder términos potencialmente relevantes. Pero esto es a costa de una reducción en la precisión, porque los lenguajes naturales no suelen ser regulares en sus construcciones y además hay muchas palabras con la misma raíz cuyo significado no tiene nada que ver. Por ejemplo, podría suceder que se indexen bajo la raíz “cas” los términos “casa”, “casero” y “casual”. Este fenómeno se denomina *sobrelematización*. También es posible que el mecanismo de lematización falle y obtenga raíces distintas para dos palabras semánticamente similares. A esta situación se la denomina *bajolematización*. Este caso es muy frecuente en los verbos

irregulares, y se podría dar por ejemplo con dos variaciones del verbo “haber”, para las palabras “habido” y “hayamos”, indexándose bajo raíces distintas (“habi” y “hay”). Otro problema es que este método es dependiente del idioma, y por lo tanto sería necesario a la hora de indexar utilizar un mecanismo específico para cada idioma. Esta situación lleva asociada la utilización de una técnica para determinar el idioma. Además, este tipo de métodos funcionan bien con idiomas que tengan una sintaxis no excesivamente complicada, como el inglés, pero en cambio fallan mucho más con otro tipo de idiomas más complejos, como el español. Por tanto, la lematización difiere mucho dependiendo de los distintos idiomas.

Existen muchas técnicas utilizadas en este tipo de métodos, destacando la utilización de reglas y diccionarios. Existen multitud de propuestas de lematización basadas en reglas, la mayoría de ellas para el idioma inglés, de los cuales el clásico el más sencillo, el lematizador S (Hull, 96), que simplemente quita las terminaciones plurales. El método de lematización más famoso es el que se ha implementado en el algoritmo de Porter (1980), que elimina cerca de 60 terminaciones en cinco etapas, en cada una de las cuales se elimina un tipo concreto. También cabe destacar los algoritmos de Lovins (1968) y Paice (1990). Para eliminar los errores más frecuentes descritos, se han desarrollado mecanismos basados en diccionarios, como KSTEM (Krovetz, 93). Hay mucha polémica sobre la efectividad de la lematización y algunos autores afirman que esta técnica mejora la precisión y exhaustividad de las búsquedas en tanto las consultas (y también los documentos) sean más cortas.

Finalmente se puede hablar del uso de los n-gramas (subsecuencia de n elementos de una secuencia dada). Al trabajar con n-gramas se ignora el aspecto semántico de las palabras y la hipótesis de los mecanismos basados en n-gramas es que dos palabras relacionadas semánticamente suelen contener los mismos caracteres.

Siguiendo con el ejemplo:

Documento original: “...vivía cerca de León. Furioso con lo que le rodeaba, los hombres de la comarca lo odiaban...”

Documento sin signos de puntuación, sin palabras prohibidas y tras un proceso de lematización: “...viv león furios rodea hombr comarc odiaba...”

- *Eliminación de documentos duplicados*: Muchos contenidos de la páginas Web están multiplicados (como mínimo duplicados) en diferentes sitios. La eliminación de estos documentos

multiplicados permite mejorar el rendimiento de los programas encargados de la indexación y reducir el espacio de almacenamiento que ocupan los índices generados. Pero la tarea de identificar documentos iguales o similares no es trivial, ya que pueden darse diferentes situaciones que compliquen esta labor, como por ejemplo, variaciones en el formato del documento. Dos documentos pueden ser idénticos en contenido pero estar en diferentes formatos (html, pdf, Word...). Una de las formas de detectar la similitud entre documentos consiste en convertirlos a un mismo formato, normalmente texto plano, utilizando alguna herramienta de conversión estándar. Posteriormente cada documento se divide en una colección de partes o trozos formados por pequeñas unidades de texto (por ejemplo líneas o sentencias). Después, a cada trozo se le aplica una función *hash* para obtener un identificador único. Si dos documentos comparten un número de trozos con igual identificador por encima de un umbral T, entonces se consideran documentos similares.

### ***Estructuras de indexación clásicas***

En los primeros sistemas, los índices se limitaban a contener un conjunto de palabras clave representativas del documento, pero actualmente el número de términos ha crecido demasiado. Como se ha dicho, en la indexación no se utilizan todos los términos (aunque hay excepciones), sino que se suele usar un subconjunto de términos y el documento completo se almacena aparte en repositorios o caches si es posible, pero lo más habitual es que sólo se almacene su ubicación (normalmente su URL: *Universal Resource Locator*). La estructura más utilizada en la indexación de documentos es el archivo invertido, formada por dos componentes: el vocabulario y las ocurrencias (véase la Figura 1). El vocabulario es el conjunto de todas las palabras diferentes del texto. Para cada una de las palabras del vocabulario se crea una lista donde se almacenan las apariciones de cada palabra en un documento. El conjunto de todas estas listas se llama ocurrencias (Baeza Yates, 99). Este mecanismo no es el único, sino que existen otros muchos como los ficheros de firmas, basados en técnicas *hash*, árboles PAT y grafos.

<b>Documento</b>	<b>Texto</b>
1	“...vivía cerca de León. Furioso con lo que le rodeaba, los hombres de la comarca lo odiaban...”
2	“... no tiene tanta furia el león como ...”
3	“...León pertenece a Castilla...”
4	“...los hombres y las comarcas castellanas...”

Tabla A: cada línea es un documento diferente.

<b>Número de índice</b>	<b>Término</b>	<b>Documento</b>
1	<i>vivía</i>	1
2	<i>león</i>	1, 2, 3
3	<i>furi (furioso, furia)</i>	1, 2
4	<i>rodeaba</i>	1
5	<i>hombres</i>	1, 4
6	<i>comarca (comarca, comarcas)</i>	1, 4
7	<i>odiaban</i>	1
8	<i>cast (castilla, castellanas)</i>	3, 4

Tabla B: fichero invertido para los documentos de la tabla a, entre paréntesis se presentan las palabras antes de la lematización.

Figura 1: Ejemplo de fichero invertido, después de que los documentos iniciales hayan sido preprocesados

## Consulta

El inicio de un proceso de búsqueda lo origina un problema que requiere información para poder resolverse. La carencia de esta información depende de la amplitud de conocimiento de cada usuario. Un usuario avezado en un tema concreto tendrá más claro que información solucionar su problema y seguramente lo encontraría en un plazo de tiempo más corto. La aparición de un problema conlleva la demanda de información en el usuario para solucionarlo, y esta carencia de información origina lo que se denomina una “necesidad de información”. Las personas buscan información basándose en su conocimiento previo, que es muy diferente de unas a otras. La necesidad de información puede ser definida como la representación implícita de un problema en la mente de los usuarios (Mizzaro, 96). Se diferencia del problema, ya que cada usuario percibe las cosas de diferente forma, y ante un mismo problema varios usuarios pueden construir necesidades de información distintas. Las necesidades de información se pueden clasificar en necesidades verificativas, sobre temas conscientes, e imprecisas o mal definidas (Ingwersen, 95). La primera categoría se refiere a la situación en la que se buscan documentos con propiedades conocidas, por ejemplo se conoce el

nombre del autor, el título, etc. En el segundo tipo se conoce el tema y es definible, pero menos exacto que en la primera categoría. En esta categoría una persona que busca información tiene algún nivel de comprensión de lo que busca. La tercera categoría son los casos en los que una persona desea encontrar nuevo conocimiento en dominios que no le resultan familiares. Una necesidad de información se puede satisfacer de distintas formas. Es decir, el concepto de necesidad de información tiene una naturaleza ambigua. Debido a esta característica, se han comentado distintos problemas cuyo motivo es la inexactitud de la necesidad de información, como el problema ASK (*Anomalous State of Knowledge*) (Belkin, 82), ISK (*Incomplete State of Knowledge*) y USK (*Uncertain State of Knowledge*) (Ingwersen, 92).

Cuando se aborda el desarrollo de un sistema de recuperación de información se asume la idea de que las necesidades de información pueden describirse. La persona que quiere recuperar la información tiene que ser capaz de expresar la necesidad de información que demanda en forma de una petición o consulta (*query*). La petición es una representación de la necesidad de información del usuario en un lenguaje humano, casi siempre en lenguaje natural (no estructurado, como se ha comentado anteriormente en la recuperación de datos). Pero esta consulta debe ser también comprensible y procesable para el sistema de recuperación de información. Evidentemente, la representación mental de la información que el usuario necesita para resolver su problema difiere enormemente de la información que recibe el SRI del usuario. Este proceso implica una adaptación de lo que el usuario cree que resolverá su problema a una expresión que represente lo que el usuario necesita encontrar (Figura 2).

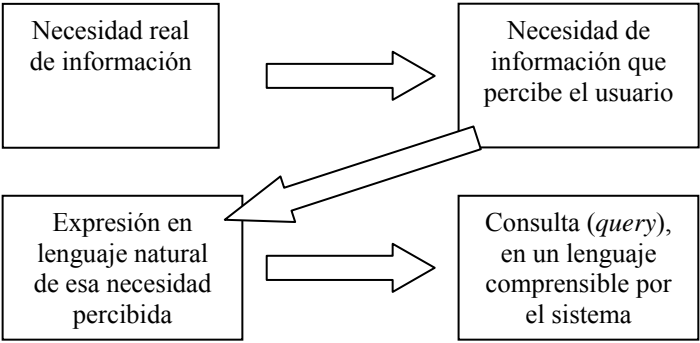


Figura 2: Proceso de construcción de las consultas

Pero no basta con seguir este proceso para obtener la información que resuelva el problema. Si los resultados no satisfacen al usuario puede



ser necesario repetir este proceso de forma cíclica. Durante cada ciclo el sistema recibe realimentación del usuario con nueva información, formalizada en forma de nuevas consultas. En este proceso, según Hofstede (96) se pueden distinguir a grandes rasgos 4 fases:

1. Fase explorativa. El usuario reúne la información que pueda serle útil en el proceso de búsqueda.
2. Fase constructiva. Se aprovecha la información adquirida en la fase anterior para reformular una nueva consulta.
3. Fase de realimentación. Si los resultados de la consulta formulada en la fase 2 no son satisfactorios es necesario volver a realizar las fases 1 y 2 para refinar el resultado.
4. Fase de presentación. Se limita a la forma de representar los resultados.

Se pueden encontrar otras muchas descripciones del proceso de búsqueda en la literatura. Por ejemplo, el modelo propuesto por Kuhlthau (88) que divide el proceso de búsqueda en siete etapas: comienzo, selección, exploración, formulación de la consulta, análisis de los resultados, recopilación de documentos y presentación de los resultados.

## **Evaluación**

Los algoritmos de evaluación que utilizan muchos de los buscadores actuales se basan en la estructura de la web para determinar la relevancia de las páginas que deben ordenar. Estos algoritmos de evaluación se denominan “algoritmos basados en enlaces” y los más usados son estos tres: *PageRank* (utilizado en el buscador Google), HITS (Hypertext Induced Topic Selection) y SALSA. Los algoritmos basados en enlaces consideran la Web como un grafo dirigido de páginas y enlaces: una página con muchos enlaces a ella se supone que es una página de alta calidad, especialmente si (circularmente) los enlaces vienen de páginas que son a su vez de alta calidad. Por tanto, se puede considerar a la Web como un grafo dimitido  $G = (P, E)$  donde P son los nodos o páginas web y E los enlaces entre las páginas.

Este tipo de mecanismos sufren el “efecto de la contribución circular” descrito por Wang (2004). Este efecto se basa en el hecho de que las páginas Web se pueden enlazar unas a otras, de forma que se produzca un camino circular entre ellas. Por tanto, cada página estimula la evaluación de las que enlaza, y si existe un camino circular, entonces estimula su propia evaluación indirectamente. Para tratar de evitar este problema, Wang propone la aplicación del concepto de “distancia en la Web”, de forma que se asignen pesos a los enlaces en función de la

importancia de la página enlazada. Estos métodos presentan el inconveniente de que son potencialmente vulnerables a ataques del tipo *link spamming*, como por ejemplo (en este caso se suele denominar *Google bombing*) cuando se introduce en Google el término “ladrones” y en la lista de resultados obtenida el primer puesto lo ocupa (u ocupaba antes de tomarse las medidas oportunas) la página de la SGAE (Sociedad General de Autores Españoles) debido a que mucha gente se ha puesto de acuerdo para enlazarla en sus páginas utilizando el término “ladrones”, (por la baja popularidad de la SGAE por el cobro de cánones en la adquisición de material informático).

El algoritmo de ordenación (*ranking*) *Pagerank* (por Larry Page, fundador de Google) define un camino aleatorio con saltos aleatorios sobre la Web (completa). Así, la puntuación *PageRank* de una página se puede interpretar como *global*, evaluando la importancia de cada página independiente del tema (Lempel, 04). En cambio, HITS y SALSA son específicos a un tema y se pueden considerar como algoritmos de evaluación *locales*. Estos dos algoritmos funcionan utilizando una pequeña porción de la Web donde los recursos correspondientes de un tema específico es probable que existan, analizando la estructura de enlaces de ese subgrafo Web y asignando a sus páginas puntuaciones *hub* y autoridad. Una página es una autoridad en un tema si contiene información valiosa y de alta calidad sobre ese tema. Una página es un “hub” sobre un tema si enlaza a buenas autoridades sobre el tema, si es por ejemplo una lista de recursos de calidad sobre ese tema.

Además de la estructura de los enlaces, también se suelen tener en cuenta otras características a la hora de evaluar una página. Por ejemplo, Google tiene en cuenta el texto que acompaña a cada enlace, ya que se supone que da una descripción general o el nombre de la página a la que enlaza. Esto tiene varias ventajas ya que permite obtener una descripción bastante exacta de la página, además permite recuperar documentos que no estén basados en texto como por ejemplo imágenes, programas o bases de datos. Otros aspectos que se suelen tener en cuenta son el título de la página, el tamaño de la fuente empleada, etc.

## ***PageRank***

La puntuación *PageRank* de una página A, denotada como  $PR(A)$ , es la probabilidad de visitar A en un camino aleatorio que implique a toda la Web, donde el conjunto de estados del camino aleatorio es el conjunto de páginas, y cada paso aleatorio es de uno de estos tipos: Elegir una página Web aleatoriamente, y saltar a ella o desde un

estado s dado, elegir aleatoriamente un enlace saliente de s y seguir ese enlace hasta la página de destino. Larry Page y Sergey Brin (Brin, 98), creadores de Google, describen el cálculo del algoritmo PageRank de la siguiente forma: Se asume que la página A tiene las páginas T1...Tn que apuntan a ella. El parámetro d es un factor que puede tomar valores comprendidos entre 0 y 1. Normalmente se establece d con el valor 0.85. Además C(A) se define como el número de enlaces que salen de la página A. El valor PageRank de una página A se determina como sigue:

$$PR(A) = (1 - d) + d(PR(T1)/C(T1) + \dots + PR(Tn)/C(Tn))$$

Hay que destacar que PageRank establece una distribución de probabilidad sobre las páginas Web, de tal modo que la suma de todos los valores PageRank de las páginas Web serán uno. El valor PageRank o PR(A) se puede calcular utilizando un algoritmo iterativo. La idea sobre la que se basa PageRank es bastante intuitiva. Asume que si una página recibe bastantes enlaces provenientes de otras páginas, entonces se supone que esa página merece ser visitada. No obstante, también tiene en cuenta el hecho de que páginas muy importantes enlacen a otra página, lo que implica que es probable que esa página sea digna de ser visitada al ser enlazada por una página de calidad. A grandes rasgos, se podría decir que el algoritmo PageRank mide la probabilidad de que un usuario visite una página Web. El factor d es la probabilidad de que un visitante que navega en una página se aburra de ella y solicite otra.

### ***HITS (Hypertext Induced Topic Selection)***

HITS se basa en un modelo de la Web que distingue *hubs* y autoridades. Cada página tiene asignado un valor “hub” y un valor “autoridad”. El valor *hub* de la página H esta en función de los valores de autoridad de las páginas que enlaza H, el valor autoridad de la página A está en función de los valores *hub* de las páginas que enlazan a A. Por tanto, según HITS cada página tiene un par de puntuaciones: una puntuación *hub* (h) y una puntuación *autoridad* (a), basadas en los siguientes principios: La calidad de un *hub* se determina mediante la calidad de las autoridades que le enlazan. La calidad de una autoridad se determina mediante la calidad de los *hubs* a los que enlaza. Por tanto, el algoritmo HITS establece que una página tiene un alto peso de “autoridad” si recibe enlaces de muchas páginas con un alto peso

de “hub”. Una página tiene un alto peso “hub” si enlaza con muchas páginas autoritativas.

### ***SALSA (Stochastic Approach for Link Structure Analysis)***

SALSA también asigna dos puntuaciones a cada página: la puntuación *hub* y *autoridad*. Estas puntuaciones se basan en dos caminos aleatorios realizados en  $G$ , el camino *autoridad* y el camino *hub*. Intuitivamente, el camino *autoridad* sugiere que las páginas autoritarias deberían ser visibles (enlazadas) desde muchas páginas. Así, un camino aleatorio de este subgrafo visita aquellas páginas con alta probabilidad. Formalmente, el estado del camino *autoridad* son los nodos de  $G$  con al menos un enlace de entrada. Sea  $v$  un nodo, y  $q_1, \dots, q_k$  los nodos que enlazan con  $v$ . Una transición desde  $v$  implica elegir un índice aleatorio  $i$  uniformemente sobre  $\{1, 2, \dots, k\}$ , y seleccionar un nuevo estado desde los enlaces salientes de  $q_i$  (de nuevo, aleatoriamente y uniformemente). Así, la transición implica atravesar dos enlaces Web, el primero de ellos se atraviesa al revés (desde el destino al origen) y el segundo se atraviesa hacia delante. Si  $\pi$  denota la distribución estacionaria del camino aleatorio descrito anteriormente, cuando la distribución inicial es uniforme sobre todos los estados. La puntuación de cada página (=estado)  $v$  es  $\pi_v$  (las páginas que no tienen enlaces de entrada alcanzarán una puntuación 0).

Cabe destacar el efecto TKC (*Tightly-Knit Community*) (Lempel, 00) que pone de manifiesto importantes diferencias entre los algoritmos HITS y SALSA. HITS favorece a los grupos de páginas que tienen muchas cocitaciones “internas”, mientras SALSA prefiere las páginas con muchos enlaces de entrada. Una comunidad estrechamente tejida (*tightly-knit community*) es un conjunto de páginas pequeño pero sumamente interconectado. El efecto TKC se da cuando dichas colecciones de páginas (comunidades estrechamente tejidas) obtienen evaluaciones altas en los algoritmos basados en los enlaces, aunque esas páginas no sean autoridad en el tema, o sólo conciernan a un aspecto de dicho tema.

## CAPÍTULO II

# Herramientas de búsqueda en la Web: Directorios, buscadores y metabuscadores

Hoy en día en Internet existen tres tipos principales de herramientas utilizadas en la búsqueda Web: los directorios, los buscadores y los metabuscadores. Se podría hablar de un cuarto tipo de herramienta, los agentes de búsqueda, cuya implantación está vinculada algunas veces a la Web Semántica.

## Breve historia de las herramientas de búsqueda en la Web

El primer buscador apareció en 1990 (todavía no existía la Web) y su nombre era *Archie*. Se conectaba a los servidores mediante FTP (*File Transfer Protocol*), descargaba el listado de archivos y creaba una base de datos basándose solamente en los nombres de los documentos, sin tener en cuenta el contenido, por lo que indexaba todo tipo de archivos. Si observamos su interfaz de búsqueda (figura 3), podemos encontrar algunas similitudes con los de algunos buscadores actuales.



The image shows a simple web form titled "Archie Query Form". It features a text input field with the label "Search for:" above it. Below the input field are two buttons: "Search" and "Reset". The entire form is enclosed in a rectangular border.

Figura 3: Interfaz de búsqueda de Archie

Poco después apareció el protocolo *Gopher*, diseñado específicamente para el almacenamiento y recuperación de información a través de

Internet, siendo el primer sistema que indexó el contenido de archivos de texto plano. Pronto adquirió gran popularidad (incluso hoy en día siguen existiendo algunos servidores *Gopher*, aunque son pocos los navegadores que soportan el protocolo), sobre todo gracias al desarrollo de dos aplicaciones que permitían realizar búsquedas de documentos sobre este tipo de servidores: *Veronica* y *Jughead*.

En agosto de 1991 Tim Berners-Lee, fundador de la Web, crea el primer sitio web. En 1993 se desarrolló el primer robot web, llamado *World Wide Web Wanderer*. La definición de robot Web<sup>1</sup> es: “Programa que automáticamente navega por la estructura de hipertexto de la Web recuperando un documento y, recursivamente, aquellos documentos hacia los que el primero tiene enlaces”.

El primer buscador web propiamente dicho fue *Aliweb*. Su robot accedía a los servidores y preguntaba por un archivo especial donde el responsable del servidor especificaba el contenido del mismo, y en base a esa meta-información construía su base de datos. Pronto se vio que los buscadores Web podían ser un nicho de mercado a explotar, y con ese fin nació *Excite* en 1993. En 1994 llega el primer directorio Web, *Galaxy*. Pero el directorio más importante de todos ha sido *Yahoo!*. Empezó como un entretenimiento de dos compañeros de clase, construyendo una página Web con enlaces a sus páginas favoritas, de las que se ofrecía una pequeña descripción. Tal fue el éxito que en 1995 se creó la empresa *Yahoo!* con una inversión inicial de dos millones de dólares.

También en 1994 aparecieron *WebCrawler*, el primer buscador que indexaba el contenido entero de los documentos, y *Lycos*, que introdujo las consultas con aproximaciones léxicas y llegó a tener la base de datos más grande de entonces (incluso fue anunciado por televisión). A finales de 1995 ya existían más de una docena de buscadores importantes, fue entonces cuando se hizo público el primer meta-buscador: *MetaCrawler*. Nació con la intención de ofrecer un interfaz común para la consulta de varios motores de búsqueda a la vez, obteniendo una única lista de resultados (ver sección sobre metabuscadores más adelante).

En 1995 *AltaVista* ofrecía la búsqueda de documentos multimedia. A partir de entonces han sido numerosos los motores de búsqueda, como también numerosas han sido las fusiones comerciales, compras y ventas entre ellos: *Inktomi*, *Infoseek*, *Ask.com* (antes Ask Jeeves), *Live Search* (antes MSN Search), *Northern Light*, *Exalead*, *Gigablast* o *WiseNut*.

En 1996, Larry Page y Sergey Brin comenzaron a desarrollar *BackRub*, un buscador que tenía en cuenta los enlaces que apuntaban a un documento a la hora de establecer su posición en la lista de resultados. Mientras *BackRub* iba ganando fama debido a su novedoso

---

<sup>1</sup> <[www.robotstxt.org/](http://www.robotstxt.org/)>

método de *ranking*, sus autores siguieron perfeccionándolo hasta que en el año 2000 presentaron *Google* como “*un prototipo de un motor de búsqueda a gran escala que hace un uso intensivo de la estructura de hipertexto [...] diseñado para recoger e indexar la Web eficientemente y producir muchos más y mejores resultados que los sistemas existentes*”. *Google* ha experimentado un crecimiento increíble, llegando a convertirse en una de las mayores empresas internacionales y ofreciendo multitud de servicios.

## Directorios

Los directorios o índices temáticos son listados de recursos organizados según una jerarquía de temas. La jerarquía sigue una estructura de árbol, de forma que vaya desde las categorías más generales hacia categorías más específicas conforme bajamos en la estructura. Tradicionalmente, los documentos que forman parte de este tipo de sistemas son clasificados por humanos, que pueden ser los propios autores de la página. Recientemente, están comenzando a aparecer algoritmos automáticos de clasificación que realizan esta tarea de forma automática. Por ejemplo, Kim (2003) propone un algoritmo que utiliza la lógica borrosa para obtener a partir de una colección de documentos una jerarquía. Otro mecanismo de categorización automática es TAPER (*a Taxonomy And Path Enhanced Retrieval system*) desarrollado por IBM. Este algoritmo construye una taxonomía en forma de árbol formada por términos que sean buenos discriminantes de la temática del documento, agrupados por clases. Posteriormente, se evalúa cada documento para obtener los términos discriminantes y a continuación se someten a un proceso de evaluación para determinar a que clase corresponde el documento. Una vez determinada la clase, ese documento se asocia con el término que describe la clase. Este algoritmo se mejoró mediante la utilización de la información que proporcionan los enlaces de cada documento.

Los directorios además suelen permitir también búsquedas por palabras clave. La ventaja de este mecanismo reside en que se pueden restringir las búsquedas a categorías particularmente relevantes, pudiéndose de esta forma mejorar la relevancia de los documentos obtenidos. Otro intento de automatizar la clasificación de los documentos dentro de un índice temático es el sistema OpenGrid<sup>2</sup> que utiliza las opiniones y comentarios de cientos de navegantes *Web* sobre las páginas para clasificar los documentos. Para ello los

---

<sup>2</sup> <[www.opengrid.org](http://www.opengrid.org)>

creadores de las páginas *Web* introducen en el enlace información acerca de la categoría y *ranking* del documento. De esta forma, juntando toda la información referente a esa página, se puede obtener de forma distribuida una categorización y ranking. Evidentemente, OpenGrid sólo es una propuesta, ya que necesita que los creadores de las páginas *Web* se pongan de acuerdo para incluir esta información en los enlaces que introducen en sus páginas.

Otros directorios muy conocidos y usados actualmente son Terra<sup>3</sup>, Yahoo<sup>4</sup> y el Open Directory Project DMOZ<sup>5</sup>, aunque por ejemplo muchos periódicos, como El Mundo<sup>6</sup> o El País<sup>7</sup> utilizan sus ediciones digitales prácticamente con formato de directorios.

## Buscadores Web

Los *buscadores* o *motores de búsqueda* son sistemas de recuperación de información que indexan los documentos de la *Web* sin seguir una estructura jerárquica como hacen los directorios. Este tipo de sistemas poseen unos programas especializados en recorrer la *Web* de forma automática denominados *crawlers* (también llamados *robots*, *spiders*, *wanderers*, *walkers* o *knowbots*), que indexan los documentos que no contiene su base documental. Normalmente este tipo de sistemas abarca un mayor número de documentos que los directorios debido al proceso de automatización de la indexación. Además, suelen estar mejor actualizados que los directorios ya que cada cierto tiempo se comprueba si el documento referenciado no ha sufrido modificaciones, tanto en su contenido como en su ubicación. La forma de buscar documentos en este tipo de sistemas consiste en que el usuario introduce una consulta (*query*) con un conjunto de términos relacionados con sus necesidades de información.

## Ventajas e inconvenientes de los buscadores Web

Para medir la bondad y la capacidad de un sistema de recuperación de información se suelen usar una serie de métricas estándar. Las más usadas son la exhaustividad (*recall*) y la *precisión*. Debido a su

---

<sup>3</sup> <www.terra.es/>.

<sup>4</sup> <www.yahoo.com>.

<sup>5</sup> <dmoz.org>.

<sup>6</sup> <www.elmundo.es/>.

<sup>7</sup> <www.elpais.es/>.



tendencia inversa, hoy se usan algunas métricas que tienen en cuenta simultáneamente ambas medidas (como por ejemplo la *f-measure*). *Recall* o *exhaustividad*, es la proporción de material relevante recuperado con respecto al total de los documentos que podrían ser relevantes con respecto a una determinada consulta en la colección de documentos, independientemente de que éstos se recuperen o no.

$$recall = \frac{N^{\circ} \text{ de documentos .relevantes .recuperados}}{N^{\circ} \text{ de documentos .relevantes .de la colección}}$$

Está claro que cualquier sistema de recuperación de información podría conseguir una *Exhaustividad* del 100% simplemente devolviendo todos los elementos de la colección. Por ello se utiliza también otra métrica, llamada *Precisión*, que se define como la proporción de elementos recuperados realmente relevantes para una consulta dada del total de los documentos recuperados.

$$precisión = \frac{N^{\circ} \text{ de documentos .relevantes .recuperados}}{N^{\circ} \text{ total .de documentos .recuperados}}$$

Para comprender mejor estas medidas, en la figura 4 se muestra una representación del espacio de documentos de la Web con los conjuntos de documentos relevantes con respecto a una determinada consulta y aquellos que se recuperan con dicha consulta. La intersección de estos dos subconjuntos la constituyen aquellos documentos relevantes para la consulta que son realmente recuperados.

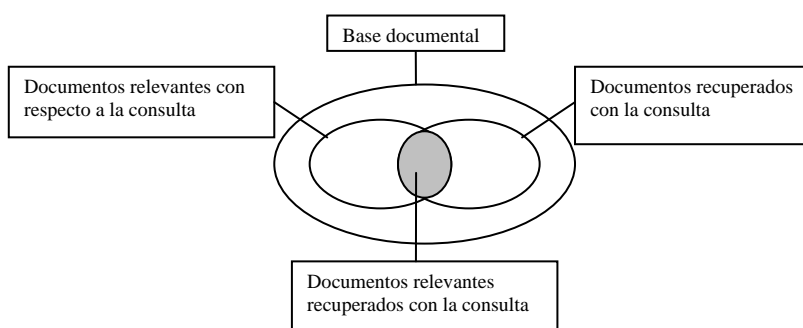


Figura 4: Conjuntos para el cálculo de la *precisión* y el *recall*

Para un mismo sistema y una misma consulta, *recall* y *precisión* son inversamente proporcionales: si se recuperan los *n* documentos de mayor relevancia, tendremos una alta *precisión* y bajo *recall* para

valores pequeños de  $n$ , así como una baja *precisión* y alto *recall* para valores grandes de  $n$ . Como se ha dicho, es debido a esta tendencia inversa que cada vez más se usen métricas que tienen en cuenta simultáneamente ambos valores, como por ejemplo la *f-measure*, que sería una media armónica entre *precisión* y *recall*. Si se les da a ambas el mismo peso,  $F$  quedaría así (esta versión se suele llamar  $F_1$ ):

$$F = \frac{2 \times \textit{precisión} \times \textit{recall}}{\textit{precisión} + \textit{recall}}$$

Los sistemas de recuperación de información comerciales no suelen hacer afirmaciones explícitas sobre la relevancia o no de un documento, sino que ordenan la colección de mayor a menor relevancia respecto a una consulta, según los criterios (algoritmos como *PageRank*) que el sistema considere oportuno. Por esta razón, generalmente no es posible calcular estas métricas para los buscadores Web comerciales. Para evaluar buscadores experimentales o en desarrollo se suelen usar colecciones de prueba que contienen información sobre la relevancia de sus documentos con respecto a consultas predefinidas. Las colecciones de prueba más usadas actualmente son TREC<sup>8</sup> o Reuters<sup>9</sup>.

Los mayores inconvenientes que presentan los buscadores comerciales están estrechamente ligados a la *precisión* y el *recall*. Lawrence y Giles (1999), en su trabajo de evaluación de los buscadores, identificaron cinco problemas principales que con frecuencia presentan:

- *La cobertura de los buscadores decrece con el crecimiento de la Web*: Los sistemas de indexación no son capaces de contemplar el rápido crecimiento de la Web, quedándose una gran parte de ella fuera de los índices de los principales buscadores, y por tanto, no es accesible para los usuarios que utilicen los buscadores.
- *Acceso desigual*: Existen tendencias a la hora de indexar las páginas debido al rastreo de los buscadores en busca de enlaces a otras páginas que indexar. Por ejemplo, será más probable que se indexen páginas que reciben muchos enlaces de otras páginas (sitios populares). Así mismo, es más probable que se indexen sitios comerciales en vez de sitios educacionales.
- *Enlaces rotos*: los *crawlers* verifican cada cierto tiempo si la página que indexan no se ha movido. Pero esta comprobación,

---

<sup>8</sup> <trec.nist.gov/>.

<sup>9</sup> <archive.ics.uci.edu/ml/datasets/Reuters-21578+Text+Categorization+Collection>.

debido al elevado número de documentos que indexan, requiere de un cierto tiempo, durante el cual pueden producirse inconsistencias debido a que una página se cambie de sitio o que desaparezca.

- *Baja utilización de meta datos:* muchos buscadores utilizan los metadatos definidos en la página Web como fuertes indicadores a la hora de indexar. Sin embargo, muy pocas páginas utilizan los meta-tags de HTML que cubren esta función como los tags “Keywords” y “description” (solo un 34% según Lawrence y Giles). Existen otros mecanismos que incorporan un conjunto de meta-etiquetas estándar en las páginas Web que ayudan a determinar distintos parámetros del documento, como por ejemplo Dublin Core<sup>10</sup>, pero su utilización es todavía muy baja.
- *Distribución de la información:* la información en la Web es muy variada y además su distribución es desigual. Por ejemplo, existe un mayor número de sitios de carácter comercial frente a los de carácter educativo o científico.

Pero las características que los usuarios mejor valoran de los buscadores son las siguientes:

- Fácil de utilizar.
- Carga y repuesta rápidas.
- Fiabilidad y precisión de los resultados.
- Información organizada y actualizada.

## Operadores de búsqueda

La facilidad de uso es uno de los principales factores a la hora de utilizar un buscador. Sin embargo, este tipo de sistemas suele incorporar una serie de opciones avanzadas que permiten especificar distintos filtros que se aplicarán a los resultados. Estas características se suelen implementar mediante formularios de búsqueda avanzados donde se indican varias opciones, o mediante operadores de búsqueda. A continuación se presenta una pequeña descripción de los operadores más usados en los buscadores Google<sup>11</sup>, Yahoo!<sup>12</sup> y Bing<sup>13</sup> (Antes LiveSearch).

---

<sup>10</sup> <dublincore.org/>.

<sup>11</sup> <www.google.com/>.

<sup>12</sup> <www.yahoo.com/>.

<sup>13</sup> <www.bing.com/>.

## **Operadores lógicos**

Son los operadores clásicos booleanos. Para simplificar, y debido a la dificultad que implica la utilización de estos operadores para usuarios noveles, se suelen indicar pantallas de opciones avanzadas con las opciones ‘incluir todos los términos’ en vez del operador AND, ‘incluir alguno de los términos’ en lugar del operador OR y ‘excluir el termino’ en vez del operador NOT. Suelen utilizarse los siguientes:

**AND, “+”ó “&”:** recupera aquellos documentos que contengan todos los términos de búsqueda. Cuantos más términos se usen, el número de resultados será menor, pero éstos serán más específicos. En la mayoría de buscadores, el operador AND es el operador por defecto. Google, cuando se usa este operador, indica que no es necesario porque ya es usado por defecto, sin embargo el número de resultados es ligeramente distinto. Se debe a que el operador AND no es lo único usado por defecto, sino que Google también tiene en cuenta el orden de las palabras de manera que concede mayor relevancia a aquellos documentos en los que los términos de búsqueda aparezcan en el orden indicado en la consulta, por lo que dos búsquedas con los mismos términos pero en diferente orden pueden producir resultados distintos.

Ejemplo: Se pueden ver las diferencias en los resultados al ejecutar las consultas:

drink AND drive

drive AND drink

**OR:** recupera aquellos documentos que contengan al menos uno de los términos de búsqueda, pudiendo contener también varios o todos ellos. Este operador hace que el número de resultados se incremente considerablemente pero, usado adecuadamente junto a otros operadores, resulta útil para realizar búsquedas con sinónimos. El operador OR siempre ha de escribirse con mayúsculas. En los tres buscadores analizados puede sustituirse por la barra vertical |. Es útil para incluir sinónimos, variaciones léxicas, voces extranjeras...

Ejemplo: Si quisiéramos encontrar monitores de ordenador baratos podríamos incluir en la búsqueda el sinónimo *pantalla*. Mejoraríamos radicalmente la relevancia de los resultados simplemente poniendo:

monitor OR pantalla barato.

**NOT, “-” ó “!”:** como en la lógica clásica, este operador es unario y descarta de los resultados aquellos documentos que contienen

los términos precedidos por el operador, por lo que restringe el número de resultados. En algunos buscadores su sintaxis es *AND NOT*, (en Bing puede usarse indistintamente una forma u otra). En Google, Yahoo y Bing puede sustituirse por el signo menos -, pero sólo cuando no se especifica ningún otro operador booleano. Este operador es útil cuando se prevén términos no deseados en los resultados, o si se encuentran al examinarlos tras una búsqueda ya realizada.

Ejemplo: Si quisiéramos encontrar los pueblos que se llaman Clinton tendríamos que evitar las miles de páginas que hablan de Bill y Hillary Clinton. Mejoraríamos radicalmente la relevancia de los resultados simplemente poniendo:

clinton -bill -hillary

**Anidamiento de operadores booleanos:** el anidamiento de operadores booleanos, mediante el uso de paréntesis, permite realizar consultas más elaboradas y específicas, pero su construcción es difícil para la gran mayoría de los usuarios, pues dependiendo del orden de los paréntesis, las consultas pueden tomar un significado u otro (actúan según el álgebra de Boole). Además, no todos los buscadores ofrecen esta posibilidad, como es el caso de Google, que siempre da mayor relevancia al operador AND sin tener en cuenta el orden de los paréntesis.

Ejemplo: Si quisiéramos encontrar personajes que se llamen Bill o Hillary pero que no sean los Clinton tendríamos que evitar las miles de páginas que hablan de Bill y Hillary Clinton. Mejoraríamos radicalmente la relevancia de los resultados simplemente poniendo:

(bill OR hillary) -clinton

### ***Otros operadores***

**Inclusión obligatoria de un término.** El signo más + se usa para obligar a la inclusión de un término en el resultado, pero dado que el operador AND es usado por defecto sólo es útil para forzar la aparición de “stop words” (términos de uso tan común que son obviados) en los resultados y evitar que se busquen posibles variaciones gramaticales del término (en aquellos casos en los que se haga automáticamente). Como los documentos se ven obligados a contener otra palabra, el buscador restringe el número de resultados.

Ejemplo: Si quisiéramos encontrar información sobre la película *Wild at heart* (David Lynch), tratando de evitar aquellos documentos que contiene las palabras *wild* y *heart* por separado, mejoraríamos radicalmente la relevancia de los resultados simplemente poniendo:

wild AND +at AND heart

En el este caso se está forzando la aparición del término *at*. En la mayoría de buscadores este operador solamente funciona con términos en inglés.

**Búsqueda de frases literales.** La búsqueda de frases literales (que han de ir entrecorridas) devolverá aquellos documentos en los que las palabras aparezcan en el orden en el cual se especifica en la consulta). En Google también pueden realizarse búsquedas de frases literales uniendo los términos con puntos (sin espacios y sin comillas). En Google pueden usarse palabras comodín mediante uno o varios asteriscos en cualquier posición dentro de una frase entrecorrida.

Ejemplo: para encontrar la frase de *Cien años de soledad* “el Coronel Aureliano Buendía había de recordar” la consulta podría ser “el Coronel \* \* había de recordar”. Sin embargo, Google hace que a veces un asterisco pueda ser usado para representar a más de una palabra. En Yahoo las palabras comodín no se representan mediante asteriscos, sino con alguna *stopword*, una *a* por ejemplo. Tanto comodines se deseen usar, tantas *stopwords* hay que incluir.

**Truncamiento.** Ninguno de los tres buscadores analizados dispone de un operador específico para ello. Google realiza búsquedas automáticas de ligeras variaciones de términos que pueden ser deshabilitadas, como se ha dicho anteriormente, mediante el uso del operador +. Yahoo ni siquiera realiza búsquedas sobre variaciones del término automáticamente. Dispone del operador *stem*: para lanzar un algoritmo de *stemming* para términos en inglés. Por ejemplo, *stem:big* buscará documentos que contengan *big* y *bigger*, entre otras.

**Sinónimos** (Google). El operador ~ usado justo antes de un término sirve para realizar la búsqueda también sobre sinónimos de dicho término. Este operador de sinónimos también incluye, a veces, búsquedas sobre plural/singular y otras variaciones gramaticales.

Ejemplo: Si quisiéramos encontrar información sobre pantallas, mejoraríamos radicalmente la relevancia de los resultados simplemente poniendo:

~pantalla

Se puede observar que se recuperan páginas que hablan de *monitor, ventana, escritorio, display...*

**Operadores posicionales o de proximidad:** afectan a la posición de los términos en el documento y las relaciones de las palabras de la consulta atendiendo normalmente a criterios de proximidad u orden. Los más usuales son:

- **NEAR:** Se sitúa entre dos términos de la consulta para indicar que recupere los documentos que contengan ambos términos, pero que no estén separados por un número determinado de palabras. Este número oscila entre 25 palabras o 100 caracteres, aunque a veces este número es configurable.
- **FAR:** es el operador contrario a NEAR y recupera documentos en las que debe haber una distancia mínima entre los términos.
- **ADJ:** Se utiliza aplicado a dos términos y recupera solo los documentos que poseen los dos términos y además están juntos en el documento. El orden no se tiene en cuenta.
- **FOLLOWED BY:** Es un operador parecido a NEAR pero define muy claramente el cual debe ser el orden de los términos.
- **BEFORE:** Funcionamiento parecido al operador AND, pero teniendo en cuenta el orden de aparición en el documento.
- Existen distintas variantes o modificaciones que se les pueden aplicar a estos operadores dependiendo de las características propias del lenguaje de consulta. Por ejemplo, existen modificadores de orden para los operadores ADJ, NEAR y FAR, que consiste en añadir delante del operador la letra O de orden (OADJ, ONEAR y OFAR), de esta forma si se utiliza por ejemplo *coches OADJ carreras* sólo recuperará los documentos referidos a coches de carreras y no a carreras de coches. Otro modificar que afecta a NEAR y FAR es el de la distancia entre palabras, que se puede indicar mediante el parámetro “/”, por ejemplo NEAR/3 quiere decir que la máxima diferencia en palabras entre los términos es 3 palabras. También se puede utilizar esta característica con ADJ, para indicar el número exacto de palabras que debe haber entre los dos términos. No obstante, existen algunos otros operadores dependiendo del buscador, como por ejemplo A WITHIN 10 BEFORE B que indica que entre los términos A y B no debe haber más de 10 caracteres. Otro operador sería WITH

o SENT que indica que dos términos deben aparecer en la misma sentencia, etc.

**Incrementar importancia de un término.** En Google se puede incrementar la importancia de un término repitiéndolo. Cuanto más se repita, más importancia adquiere. A continuación se muestra un ejemplo donde

Ejemplo: Si se busca información sobre los términos coches y carreras, haciendo énfasis en el término carreras, mejoraríamos radicalmente la relevancia de los resultados simplemente poniendo:

coches AND carreras AND carreras (Google)

### **Búsqueda por campos.**

#### **Título:**

- **intitle:** Recupera aquellos documentos que contengan en el título el término indicado por el operador (tantos términos se deseen, tantas veces que se ha de usar el operador). Puede ser usado junto con otros términos de búsqueda y operadores.

intitle:motor intitle:diesel audi

Según el ejemplo, se obtendrán documentos con los términos motores y dieselen en el título y audi en cualquier otra parte del documento (que también puede ser, o no, el título). Es una manera eficaz de restringir el número de resultados y aportar más especificidad a la búsqueda. Otro de sus usos, en Google, es encontrar archivos de audio, video. pdfs,... Al realizar una búsqueda como la del siguiente ejemplo, junto a algunas páginas web, en los resultados se obtiene acceso directamente al árbol de directorios de los servidores web que se encuentran con la configuración por defecto, desde donde podremos descargar directamente los archivos deseados. Por ejemplo, para buscar archivos en determinado formato habría que realizar una búsqueda de la siguiente manera:

intitle:index.of formato\_archivo terminos\_búsqueda

Por ejemplo:

intitle:index.of mp3 pink floyd    ó    intitle:index.of pdf cervantes

- **allintitle:** operador específico de Google. Recupera aquellos documentos que incluyen todos los términos indicados en el título. Equivale a usar *intitle* para todos los términos de búsqueda. No puede combinarse con restricciones a otros campos.



### **Cuerpo del documento:**

- **inbody:** operador específico de Bing. Recupera aquellos documentos que contengan los términos indicados en el cuerpo del documento, pudiendo aparecer también en otros sitios. Dado que por defecto los términos siempre se buscan en el cuerpo del documento, además de en otros sitios, parece un operador inútil (de hecho, es el único buscador que lo incorpora), pero su uso reduce el número de resultados.

### **URL:**

- **inurl:** Recupera aquellos documentos que contengan en cualquier parte de la URL (nombre del *host*, ruta o nombre del archivo) el término indicado por el operador (tantos términos se deseen, tantas veces que se ha de usar el operador). Puede ser usado junto con otros términos de búsqueda y operadores. Este operador resulta útil para encontrar páginas de búsqueda y ayuda, porque tienden a tener una composición regular.
- **allinurl:** operador específico de Google. Recupera aquellos documentos que incluyen todos los términos indicados en la URL. Equivale a usar *inurl:* para todos los términos de búsqueda. No puede combinarse con restricciones a otros campos.

### **Enlaces (hipervínculos):**

- **inanchor:** operador específico de Bing. Recupera aquellos documentos que contengan en los enlaces incluidos en el documento (elemento HTML `<a href=...>`) el término indicado por el operador (tantos términos se deseen, tantas veces se ha de usar el operador). Puede ser usado junto con otros términos de búsqueda y operadores.
- **allinanchor:** operador específico de Google. Recupera aquellos documentos que incluyen todos los términos indicados en los hipervínculos a otros documentos (el operador *inanchor:* no está disponible en Google). No puede combinarse con restricciones a otros campos.

### **Fecha de publicación.**

- **daterange:** operador específico de Google. Sirve para recuperar documentos publicados dentro de un rango de tiempo. De uso muy complicado, se basa en el día juliano (usado para fechar fenómenos astronómicos), que se obtiene contando los días habidos desde el 1 de enero del año 4713 a.C en adelante (hasta el límite del año 7980). Por ejemplo, el día 3 de mayo de 2002 equivaldría al día

juliano de 2452397. Indicar que Google “refresca” algunos documentos más frecuentemente que otros, en concreto aquellos que detecta que son actualizados más asiduamente.

Ejemplo:

guerra AND Afghanistan daterange:2452392-2452395

Según el ejemplo, se recuperarán documentos publicados entre el 28 de abril y el 1 de mayo del año 2002.

**Tipo de documento:**

- **filetype:** operador específico de Google. Sirve para obtener documentos del tipo deseado, entre los cuales se pueden encontrar .pdf, .doc (Microsoft Word), .ps (Adobe PostScript), .xls (Excel), .ppt (PowerPoint) y .rtf (Rich Text Format).txt, .wpd (word perfect), .swf (Shock Wave Flash) y otros más.
- **originurlextension:** operador específico de Yahoo, de uso idéntico al de anterior.
- **contains:** operador específico de Bing. Este operador es un tanto distinto a los dos anteriores, pues no devuelve documentos de un tipo, sino páginas Web que contienen enlaces hacia archivos del formato especificado.
- **feature:** operador específico de Yahoo, el cual puede ir seguido de varias claves para realizar búsquedas muy específicas. Las claves para que se comporte como el operador *contains*: de Bing son:
  - **feature:acrobat** páginas que contienen enlaces a archivos PDF.
  - **feature:applet** páginas que contienen *applets* embebidos.
  - **feature:activex** páginas que contienen controles ActiveX.
  - **feature:audio** páginas que contienen enlaces a archivos de audio en varios formatos.
  - **feature:flash** páginas que contienen archivos Flash o enlaces hacia ellos.
  - **feature:image** páginas que incluyen imágenes en varios formatos.
  - **feature:video** páginas que contienen videos o enlaces a ellos.
  - **feature:vrml** páginas que contienen enlaces a archivos VRML.
  - **feature:shockwave** páginas que contienen archivos ShockWave o enlaces a ellos.

### **Enlaces a un documento:**

- **link:** recupera documentos que contienen enlaces a la URL especificada. En Bing se puede incluir el prefijo `http://`, y los resultados suelen ser los mismos poniendo o no `www`.

Ejemplo:

`linux link:barrapunto.com`

La consulta del ejemplo recuperará documentos que contienen el término `linux` y un enlace a `www.barrapunto.com`. Nótese que no es necesario poner `www`.

### **Documentos pertenecientes a un dominio.**

- **site:** solamente recupera documentos contenidos en un dominio Web, pudiendo también especificarse rutas y tipos de dominio (es, com, org, gov, ...). Si no se especifican términos de búsqueda, se recuperaran todos los documentos indexados pertenecientes al sitio.

Ejemplo:

`site:barrapunto.com amarok`

La consulta anterior devolverá todos aquellos documentos pertenecientes al dominio `barrapunto.com` en los que aparece el nombre del reproductor musical *amarok*.

- **ip:** operador específico de Bing. Similar al operador *site:* pero indicando la dirección IP.
- **domain:** operador específico de Yahoo. Realiza la misma función que *site:*.
- **hostname:** operador específico de Yahoo. Realiza la misma función que *site:* y *domain:* excepto que solamente acepta URLs y no tipos de dominio.

**Búsqueda por idioma.** Por defecto, Google realiza sus búsquedas en todos los idiomas que tiene disponibles, pero es posible filtrar los resultados para una búsqueda indicando uno de ellos, lo cual se hace desde la página de búsqueda avanzada. También se puede configurar para que por defecto solamente muestre los resultados en ocho idiomas seleccionados, lo cual se hace desde la página de Preferencias.

Yahoo dispone de 36 idiomas desde la página de búsqueda avanzada, pudiendo restringir desde ahí a varios de ellos.

- **language:** operador específico de Google que se usa seguido de un código de dos letras que sirve para determinar el idioma deseado. Desde la búsqueda avanzada sólo se puede restringir la búsqueda a un idioma, pero desde la página de *settings* se pueden seleccionar varios a la vez.

**Filtro familiar.** Usado para excluir contenidos para adultos de los resultados. Se configura desde la página de preferencias. Por defecto se encuentra en “Filtro moderado”, el cual sólo filtra imágenes con contenidos explícitos. También puede ponerse en “Filtro estricto”, el cual también filtrará texto explícito, o dejar sin filtrar los resultados en absoluto. Sin embargo, ninguno de los filtros, incluido el estricto, logra bloquear todo el contenido explícito. En Bing es similar a Google y se encuentra en la página de *settings*. En Yahoo el filtro solamente tiene dos estados, activado o desactivado, y se encuentra disponible en la página de preferencias.

**Búsqueda por país.** En Google la búsqueda de documentos por país se realiza accediendo a la dirección de Google para ese país (por ejemplo, para España es [www.google.es](http://www.google.es) y en Argentina [www.google.com.ar](http://www.google.com.ar)) y marcando la opción que restringe los resultados a ese país. En Yahoo el país puede elegirse desde la página de búsqueda avanzada. Además permite buscar por regiones usando el operador *region*: junto con el código de la región, los cuales pueden ser: africa-asia-centralamerica-downunder-europe-mediterranean-mideast-northamerica-southamerica-southeastasia.

Bing dispone del operador *loc*: que es usado junto a un código de dos letras que determina el país del cual queremos recuperar los documentos. Bing automáticamente intenta detectar la localización del usuario, la cual puede usarse para efectuar una búsqueda sobre documentos “cerca” al usuario pulsando sobre el botón “Near me”.

### **Rango numérico.**

- **numrange:** operador específico de Google. Recupera documentos que contienen números incluidos en un rango. Puede usarse como `numrange:5-11` o `5..11`. También tiene en cuenta los números decimales contenidos en el rango. Hay que asegurarse de poner el número más pequeño al principio del rango o no funcionará. También es posible buscar números “en solitario”, es decir, sin especificar un rango, pero no pueden buscarse números negativos (aunque luego se encuentren en los resultados) porque el signo – se interpreta como el operador NOT. Los números y rangos de números pueden incluirse en las frases entrecomilladas. Los números en solitario pueden incluir comas en los resultados, es decir, una búsqueda sobre 2001 puede dar como resultado un documento conteniendo 2,001, pero no si usamos 2001..2001.

Ejemplo:

bmw caballos numrange:100-200

Que devolverá documentos con información sobre coches bmw con un rango de potencias de entre 100 y 200 caballos.

- **pricerange:** operador específico de Google. Encuentra páginas que contienen números incluidos en un rango y precedidos por el signo \$.

Ejemplo:

korando pricerange:10000-30000

Que devolverá documentos con información sobre coches del modelo Korando con un rango de precios de entre 10.000 y 30.000 unidades monetarias (incluye euros).

### Otro tipo de consultas.

- **feed:** encuentra suministros RSS y *Atom* que contienen el término de búsqueda.  
feed:noticias motor
- **hasfeed:** encuentra documentos Web que contienen enlaces a suministros RSS y *Atom*.  
hasfeed:noticias motor
- **url:** disponible sólo en Bing. Consulta si la URL se encuentra indexada en la base de datos del buscador. No puede combinarse con otro tipo de búsquedas. Puede, o no, incluirse http://.
- **related:** operador específico de Google. Encuentra sitios Web de contenido similar al indicado.  
related:barrapunto.com (186 resultados)  
www.barrapunto.com es un dominio Web español con noticias sobre Linux, software libre, ciencia y tecnología. Con el ejemplo anterior se recuperan otros sitios Web de todo el mundo que tratan los temas mencionados.
- **linkdomain:** operador específico de Bing. Limita la búsqueda a aquellas páginas que contienen enlaces a alguna página perteneciente al dominio especificado. Se puede usar también con direcciones IP y no se ha de incluir http://.
- **linkfromdomain:** operador específico de Bing. Ofrece una lista de enlaces hacia “el exterior” del dominio, los cuales están contenidos en las páginas que forman el dominio.
- Las siguientes claves para el operador *feature:* de Yahoo sirven para encontrar:
  - **feature:form**, páginas que usan formularios para la entrada de datos.
  - **feature:frame**, páginas que usan *frames* (marcos).

- **feature:homepage**, páginas personales que usan el símbolo ~ antes del nombre del directorio.
- **feature:index**, páginas de inicio (*home*) solamente.
- **feature:javascript**, páginas que usan JavaScript
- **feature:meta**, páginas que incluyen meta-etiquetas HTML.
- **feature:script**, páginas que contienen scripts embebidos.
- **feature:table**, páginas que usan tablas.

## Arquitectura de los buscadores

Se pueden distinguir dos tipos de buscadores: de propósito general y de propósito especial. Los buscadores de propósito general abarcan páginas de toda la Web sin centrarse en ningún dominio específico. Los buscadores de este tipo más conocidos son *Google*, *Altavista*, *Excite*, *Lycos* y *Bing*. Por el contrario, los buscadores de propósito especial se centran en documentos pertenecientes a un dominio concreto como puede ser una temática determinada o los documentos de una organización. Algunos ejemplos de este tipo de buscadores son *Citeseer*<sup>14</sup> que se centra en artículos de investigación y *Medical World Search*<sup>15</sup> que se centra en información sobre medicina.

Las distintas herramientas utilizadas para buscar en la Web presentan diferentes arquitecturas aunque suele existir un conjunto de componentes básicos común. Las implementaciones de la mayoría de buscadores comerciales no suelen ser públicas, a no ser que sean de carácter experimental. Afortunadamente, existen algunas excepciones que se describen a continuación.

Los cinco componentes principales de motor de búsqueda Web son: el *crawler* (o robot), el generador del índice, el índice (base documental), el motor de búsqueda propiamente dicho y el interfaz con el usuario (figura 5).

---

<sup>14</sup> <citeseer.ist.psu.edu>.

<sup>15</sup> <www.mwsearch.com>.

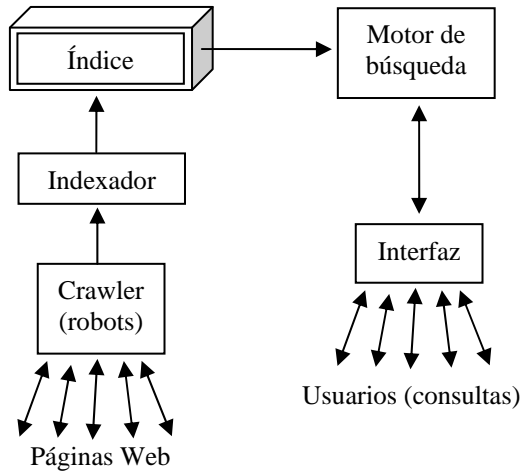


Figura 5: Arquitectura básica de un motor de búsqueda

*Crawler.* Puede definirse como: “Programa que automáticamente navega por la estructura de hipertexto de la Web recuperando un documento y, recursivamente, aquellos documentos hacia los que el primero tiene enlaces”. Es decir, va creando un catálogo de documentos que luego pasarán a ser procesados para su indexación en la base documental.

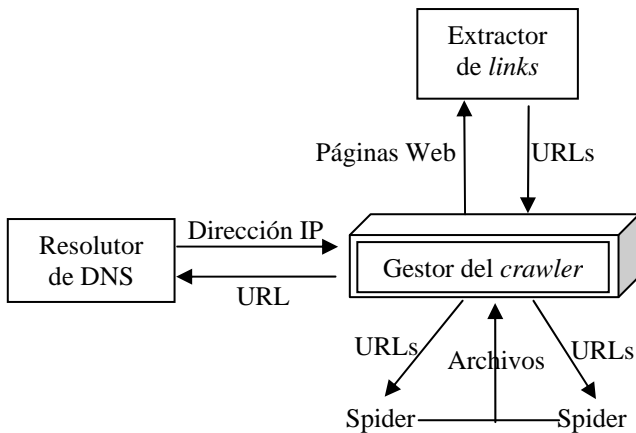


Figura 6: Arquitectura de un crawler sencillo

En ocasiones la visita de un *crawler* puede generar inconvenientes técnicos en el servidor, y puesto que no decide por sí mismo qué documentos debe o no recuperar, puede acceder a zonas del servidor

cuyo responsable desea que no lo sean. Para evitar estos problemas los *crawlers* pueden hacer uso del “*Protocolo de exclusión de robots*”. Según la especificación: “*No es un estándar oficial [...] ni pertenece a ninguna compañía comercial. No obliga a nadie y no garantiza que todos los robots lo usen. Ha de considerarse como una facilidad que los autores de robots ofrecen a la comunidad para proteger los servidores Web de accesos indeseados*”. El método es sencillo, sólo hay que colocar en el servidor un archivo llamado *robots.txt*, donde el responsable del servidor especifica una serie de directivas indicando los documentos o directorios que el *crawler* no debe tener en cuenta, así como otras de índole técnico. Si el archivo no existe, el *crawler* asume por defecto que puede acceder a todo el contenido.

*Generador del índice.* Su función es realizar el procesado de los documentos e incluirlos en la base de datos (índice). El procesado consiste en establecer unos parámetros de frecuencia de aparición de cada uno de los términos encontrados en cada uno de los documentos (como se ha visto, tales parámetros varían según el tipo de modelo utilizado: vectorial...) para generar unos índices sobre los cuales realizar las búsquedas.

*Base de datos.* En la base de datos, o índice, el buscador almacena la información de los documentos recuperados por el *crawler* mediante un algoritmo de indexado. Cuando se realiza una búsqueda realmente no se está consultando la Web sino el índice, que consiste en una representación parcial y desactualizada de la Web. Hay varios tipos de índices, siendo el más común, como se ha visto, el *archivo invertido*.

*Motor de búsqueda.* Recoge las consultas introducidas por los usuarios, las preprocesa si es necesario y las contrasta con los datos contenidos en el índice, obteniendo de esta manera la lista de resultados ordenados según la relevancia de cada documento con respecto a la consulta realizada.

*Interfaz de búsqueda.* El usuario expresa su consulta mediante términos y operadores sobre dichos términos. Cuando la consulta es enviada al motor de búsqueda, ésta se examina y manipula para poder ser contrastada con los datos contenidos en la base de datos. Tras obtener los documentos relevantes a la consulta éstos son ordenados por un algoritmo de *ranking* acorde con el modelo del índice, de forma que los documentos más relevantes para la consulta realizada aparezcan en primer lugar.

Un buen representante de la arquitectura típica de un buscador en la Web es la arquitectura de *Altavista* (véase la figura 5). Baeza-Yates (1999) describe la arquitectura de *Altavista* como una arquitectura centralizada. Se le aplica este calificativo debido a que utiliza un proceso de rastreo e indexación de la Web centralizado. Este rastreo de la Web en busca de páginas Web lo realiza el componente denominado *crawler*, el cual se encarga de enviar peticiones a los servidores Web remotos para obtener



las páginas Web. Una vez que las tiene las examina en busca de nuevos enlaces, que posteriormente recorre. El resultado del proceso de rastreo es un listado de las direcciones de las páginas Web que el *crawler* ha encontrado. Este listado se le pasa al indexador que es el encargado de almacenar en el índice los términos relevantes del documento. Altavista se considera formado por dos partes: Una encargada de interactuar con el usuario. Su función es recibir las peticiones de los usuarios y de resolverlas. Esta formada por el Interfaz de Usuario y el motor de búsqueda y otra parte encargada del proceso de rastreo e indexación de los documentos Web. Esta parte la componen el indexador y el *crawler*. Esta arquitectura presenta un par de problemas significativos. El primero es consecuencia de la utilización de un único componente encargado de recoger las peticiones de los usuarios, lo que provoca la saturación de las líneas de comunicación, así como una sobrecarga en los servidores. El segundo problema radica la centralización del proceso de rastreo e indexación. Este planteamiento tiene problemas para manejar el crecimiento de la Web, debido al enorme volumen de datos que tienen que manejar el indexador y el *crawler*.

La arquitectura *Harvest* (Bowman, 94) utiliza una arquitectura distribuida para recoger los datos y almacenarlos, que es más eficiente que la arquitectura de *Altavista*. El principal inconveniente es que *Harvest* requiere la coordinación de varios servidores Web. Para tratar de resolver los inconvenientes de *Altavista*, debido a la saturación provocada por los *crawlers*, esta arquitectura introduce dos elementos principales: recogedores (*gatherers*) e intermediarios (*brokers*). El recogedor se encarga de recopilar y extraer información de indexado de uno o más servidores Web de forma periódica. Los intermediarios aportan el mecanismo de indexado y el interfaz de consulta a los datos recopilados. Los recogedores e intermediarios se comunican entre sí para intercambiar información de indexado, así como para balancear la carga en el tráfico de la red (Baeza-Yates, 99).

El nombre de Google viene de la palabra *googol*, que significa  $10^{100}$ . Las principales razones del éxito actual de Google son su enorme base de datos (la actual hoy en día) que indexa millones de documentos de la Web, su algoritmo de evaluación basado en la estructura de la Web (*PageRank*) y su eficiencia al estar principalmente implementado en C/C++ sobre plataformas *Solaris* o *Linux*. Google utiliza varios *crawlers* distribuidos para descargar las páginas Web. La lista de páginas que hay que recuperar se la proporciona el servidor URL. Las páginas Web recuperadas se pasan al Servidor de almacenamiento, que se encarga de comprimir y almacenar las páginas recibidas de los *Crawlers* en un repositorio. Una vez almacenadas las páginas en el repositorio, tienen que ser indexadas. Esta labor la realiza el indexador, que obtiene los

documentos del repositorio descomprimiéndolos y analizándolos, extrae de cada documento las ocurrencias de cada palabra y construye un registro de *hits*, que en definitiva son las ocurrencias de cada palabra. Cada registro de *hits* contiene la palabra, la posición en el documento, una aproximación del tamaño de la fuente y la capitalización. A continuación el indexador distribuye el registro entre un conjunto de “barriles” y crea un índice parcial ordenado. Además, el indexador extrae los enlaces de cada página junto con el texto asociado y los almacena en el *fichero de anclas*. A continuación el componente *URL Resolver* lee este fichero de anclas y extrae los enlaces, convirtiendo los relativos a enlaces absolutos. El texto que describe el enlace (*snippet*, cada uno de los fragmentos de la lista que nos devuelve Google tras una búsqueda) es asociado con el documento y almacenado en los barriles. Además, genera una base de datos de enlaces compuesta por dos identificadores de dos documentos (origen y destino del enlace). Esta base de datos es la fuente utilizada por el algoritmo *PageRank*. Por último los clasificadores (*sorters*) generan un índice invertido a partir de los documentos de los barriles, así como también se construye un nuevo *lexicón* que contemple los nuevos documentos. La resolución de las búsquedas utiliza el componente *searcher*, que se ejecuta en un servidor Web. Para la evaluación y obtención de los documentos, se utilizan el índice invertido, los barriles, el lexicón y el resultado proporcionado por el algoritmo *PageRank*. La figura 7 muestra la arquitectura de Google según sus creadores (Brin, 98).

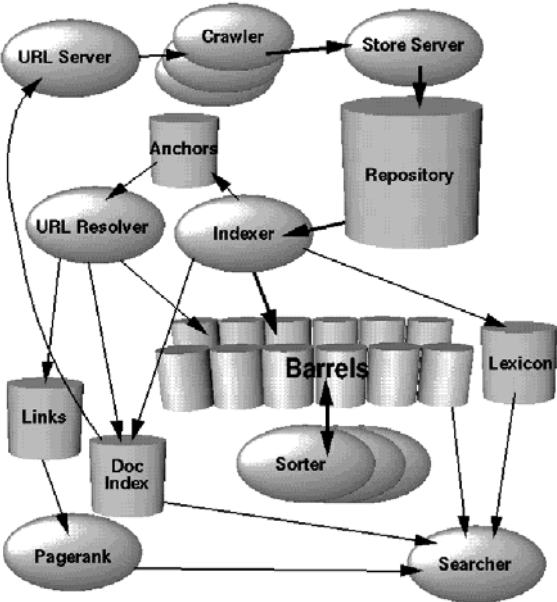


Figura 7: Arquitectura de Google (Brin, 98)

En Google los resultados son ordenados según la relevancia otorgada por el algoritmo *PageRank*, el cual, además de tener en cuenta los términos de la consulta examina los enlaces de unas páginas a otras, dando mayor importancia a un documento cuanto mayor es el número de enlaces que apuntan a él. Aparte, las páginas son agrupadas por dominio. Sólo se muestran dos páginas por sitio, con la segunda “colgando” de la primera (el resto de páginas del sitio puede verse pulsando sobre “Más resultados de...”).

En Yahoo los resultados son ordenados teniendo solamente en cuenta los términos de búsqueda y el contenido de los documentos. También se agrupan páginas de un mismo sitio Web y sólo se muestra una de ellas, las demás están disponibles pulsando en “Más páginas de este sitio” que se encuentra junto al enlace de “Caché” en el *snippet*.

Si la búsqueda devuelve menos de mil resultados, teniendo ya en cuenta el agrupamiento de dos páginas por sitio, en la última página de resultados, tras el último de ellos, aparece el mensaje *Para mostrarle los resultados más pertinentes, omitimos ciertas entradas muy similares a los que ya hemos mostrado. Si lo prefiere, puede repetir la búsqueda e incluir los resultados omitidos*. Si pincha sobre “repetir la búsqueda” se devolverán más páginas, algunas de ellas son muy parecidas, o incluso copias de las mostradas anteriormente mientras que otras son páginas que fueron agrupadas por un dominio pero no fueron mostradas (aunque no tiene por qué mostrar todas las páginas agrupadas por un dominio).

Los *snippets* son la forma en que se muestran los resultados. Incluyen el título, la URL, un extracto mostrando texto con los términos de búsqueda, el tamaño del archivo y, para algunos casos, un enlace a la copia en caché. Por defecto se muestran 10 *snippets* por página, pero en la página de Preferencias este valor puede cambiarse a 20, 30, 50 ó 100. Hay autores que han llegado a la conclusión de que la información contenida en un *snippet* no es mucho menos relevante con respecto a la consulta que la que contiene el documento completo. Google fue el primer motor de búsqueda en ofrecer acceso a los documentos tal como estaban cuando fueron indexados. La copia en caché resalta los términos de búsqueda y, si hay más de uno, utiliza un color distinto para cada uno.

## Metabuscadores

Un Metabuscador es un motor de búsqueda, que a diferencia de los buscadores conocidos, no tiene una base de datos propia donde indexar documentos. Otra desventaja con respecto a los motores de

búsqueda es que son más lentos debido a que, además de no tener base de datos propia, tienen que realizar un proceso de selección y elaboración de la lista de resultados bastante complejo. Sin embargo, disponen de un interfaz que permite al usuario consultar a la vez en diferentes motores de búsqueda, es decir, el Metabuscaador se encarga de recibir la petición del usuario, enviarla a diferentes buscadores y mostrarle después los resultados.

El principal problema que tienen los Metabuscaadores consiste en combinar las listas de resultados devueltos por los motores de búsqueda que utiliza, ya que en este proceso se deben clasificar y ordenar los documentos según su relevancia. Sin embargo, este tipo de sistemas tienen ciertas mejoras con respecto a los buscadores tradicionales, solucionan algunos de los problemas que estos tienen, como el del *recall*, aunque hay otro tipo de inconvenientes que no han conseguido resolver, como por ejemplo el de la mejora de la *precisión*. De acuerdo con Kerschberg (2001), para llegar a conseguir mayor precisión en la búsqueda, se pueden utilizar cualquiera de estos cuatro mecanismos: basados en el contenido, colaborativos, de conocimiento del dominio y basados en el uso de ontologías.

Los métodos basados en el contenido tratan de obtener una representación tan precisa como sea posible de las preferencias del usuario para después hacer una mejor evaluación y *ranking* de las páginas devueltas. Dentro de esta categoría podrían estar *WebWatcher* (Armstrong, 95), *WAWA* (Shavlik, 98) y *WebSail* (Chen, 00).

El método colaborativo se basa en establecer la similitud que puede haber entre usuarios para determinar la relevancia de la información. *Phoaks* (Terveen, 97) y *Siteseer* (Bollacker, 00) son los ejemplos más claros de este tipo de metodología.

El método basado en el conocimiento del dominio se caracteriza por utilizar dos fuentes de información para proporcionar una mayor relevancia en los resultados: la ayuda del usuario y el conocimiento del dominio de búsqueda.

Finalmente, el método basado en ontologías establece una jerarquía entre conceptos que permite concretar y mejorar la búsqueda. De este tipo son *WebSifter II* (Kerschberg, 01), que usa un árbol de representación llamado *WSTT* (*Weighted Semantic Taxonomy-Tree*) para representar las intenciones de búsqueda de los usuarios, *OntoSeek* (Guarino, 99), *On2Broker* (Fensel, 99) y *WebKB* (Martin, 00).

Las principales ventajas que encontramos en los Metabuscaadores (Meng, 02) son las siguientes:

- *Facilita la consulta simultánea en múltiples buscadores.* Es decir, permite, por medio de una única consulta, obtener la lista ordenada de los documentos más relevantes que han devuelto los

diferentes buscadores, evitándole al usuario la tarea de tener que realizar la misma consulta en cada uno de ellos.

- *Mejora la eficiencia de recuperación.* Dada la existencia de buscadores especializados en ciertos dominios, el Metabusador puede utilizarlos para mejorar los resultados finales, evitando así la información irrelevante que se pueda obtener de otros buscadores más generales.
- *Resuelve el problema de la escalabilidad de la búsqueda en la Web.*
- *Incrementa la cobertura de búsqueda en la Web.* Debido a la gran cantidad de documentos que hay en Internet, es imposible que un solo motor de búsqueda indexe la totalidad de *links* de la Web (Lawrence, 99). Por lo tanto, con la combinación de diferentes buscadores, es posible cubrir un mayor número de documentos en las búsquedas.

Así mismo, de acuerdo con Aslam (2001), un Metabusador también presenta las siguientes ventajas potenciales:

- *Arquitectura modular:* las tecnologías utilizadas en un Metabusador se pueden dividir en módulos más pequeños y especializados que puedan ser paralelizados y ejecutados colaborativamente.
- *Consistencia:* los buscadores actuales a menudo responden de manera muy diferente a la misma consulta según pasa el tiempo (Selberg, 00). Sin embargo, el Metabusador, al utilizar diferentes fuentes, tendrá menos variabilidad en los resultados ya que se verá favorecido por aquellos buscadores que proporcionen resultados más estables.
- *Mejora el factor recall:* habiendo obtenido los resultados de múltiples buscadores, se puede mejorar el número de documentos relevantes recuperados (el factor *recall*) con respecto al número total de documentos existentes.
- *Mejora la precisión:* diferentes algoritmos recuperan más documentos relevantes iguales, pero diferentes documentos irrelevantes (Lee, 97). Basándose en este fenómeno, en caso de ser cierta esta teoría, cualquier algoritmo que de prioridad a los documentos que aparecen en las primeras posiciones en los resultados de diferentes buscadores obtendrán una mejora en la recuperación. Este fenómeno se conoce como “efecto coro”.

A pesar de contar con toda esta serie de ventajas, los Metabusadores también cuentan con una serie de inconvenientes:

- *La selección de la base de datos:* este problema se asocia con la selección del buscador más adecuado para recibir la consulta introducida, seleccionar el o los buscadores que devolverán

buenos resultados para una consulta concreta no es nada sencillo. Por ejemplo, no tiene sentido la consulta “guitarra eléctrica” en un buscador especializado de literatura científica. Para intentar resolver esta desventaja, Meng (2002) propone el uso de medidas que indiquen la utilidad de cada base de datos con respecto a una consulta dada y clasifica estos mecanismos en tres categorías: métodos de representación amplia, métodos de representación estadística y métodos basados en aprendizaje.

- *La selección de documentos:* una vez seleccionado el origen de los documentos, el problema consiste en determinar el número apropiado de documentos que es necesario devolver. Si se consideran demasiados, el coste computacional para determinar los mejores documentos y el coste de comunicación para obtenerlos puede ser excesivo. Meng (2002) también estableció una serie de mecanismos para tratar de resolver este problema, y los clasificó en cuatro categorías: decisión del usuario, pesos (se obtienen mayor número de documentos del buscador que se considere el mejor), métodos basados en el aprendizaje (se basa en experiencias pasadas para determinar el número de documentos de cada buscador) y la garantía de recuperación (trata de garantizar la recuperación de todos los documentos potencialmente útiles).
- *Combinación de los resultados:* El problema consiste en combinar los resultados de diferentes buscadores, teniendo en cuenta sus características y formas de evaluación, en una lista ordenada por relevancia. Además, existe la posibilidad de encontrar documentos devueltos que estén repetidos en diferentes buscadores. Las técnicas utilizadas para resolver este problema están basadas en un ajuste de semejanza local (se basa en las características del buscador o la semejanza devuelta) y la estimación de una semejanza global (se evalúa o estima la semejanza de cada documento recuperado con la consulta original).

La traducción de una consulta a cada uno de los lenguajes específicos de los buscadores puede ser un factor importante en un Metabusador dado que cada motor de búsqueda tiene su propio lenguaje de consulta. Así pues, adaptar cada consulta al lenguaje de cada buscador parece necesario.

Hay muchas arquitecturas de Metabusadores, como entre otras las propuestas por Li (2001), Kerschberg (2001), y Glover (1999). Habitualmente, esta estructura se descompone en una serie de módulos más o menos específicos. Meng describe una arquitectura de referencia con 5 componentes:

- *Interfaz de usuario:* es el encargado de recoger la consulta del usuario y posteriormente mostrar los resultados de la búsqueda.

En algunos casos, el interfaz también contiene un sistema de refinamiento de la consulta, basada en el uso de algunas estructuras de conocimiento.

- *Selector de la base de datos*: trata de seleccionar los buscadores que darán mejores respuestas a la consulta introducida por el usuario. Trata de evitar el envío masivo de consultas a los buscadores más lentos y con un elevado coste computacional.
- *Selector de documentos*: el objetivo es obtener el mayor número de documentos relevantes, evitando recuperar los no relevantes. Si se recupera un número excesivo de documentos no relevantes, la eficiencia de la búsqueda se verá afectada negativamente.
- *Emisor de Consultas*: Es el encargado de establecer la conexión con el buscador y enviarle la consulta (o consultas), así como de obtener los resultados. Se utiliza habitualmente *http (HyperText Transfer Protocol)* por el hecho de utilizar los métodos *GET* y *POST*. Sin embargo, existen buscadores que facilitan un interfaz de programación (*API*) para enviar consultas y que utilizan diferentes protocolos (Google usa el protocolo *SOAP* en su *API*).
- *Agrupador de resultados*: Su función principal es combinar los resultados de los diferentes buscadores en una lista. Es esencial el uso de algún criterio de evaluación para establecer un orden en la lista que finalmente se muestre al usuario.

Hoy día es normal encontrar referencias bibliográficas acerca de Metabuscaadores de tercera generación (o motores de búsqueda de nivel-3). Estos trabajan de la siguiente forma: a petición del usuario se crea una base de datos de nivel-3 a partir de los resultados obtenidos por los Metabuscaadores de nivel-2. Los Metabuscaadores (que representan el nivel-2 de los motores de búsqueda) utilizan motores de búsqueda estándar (nivel-1) para encontrar los correspondientes resultados. Después se realiza un análisis de relevancia retroalimentada con estos resultados. Con esta colección de direcciones y documentos de texto se desarrolla una base de datos (a la que se denomina de nivel-3) que contiene sólo documentos que sean relevantes en ese dominio de búsqueda para el usuario. En otras palabras, se crea una base de datos centrada específicamente en los campos de interés del usuario sobre la que podrá conseguir la información que necesita obteniendo unos resultados de gran calidad. Así pues, se plantea la necesidad de incluir perfiles de usuario y personalizaciones en futuros Metabuscaadores.

Los “verdaderos” Metabuscaadores buscan simultáneamente en los mejores motores de búsqueda, agregan los resultados, eliminan los duplicados y devuelven las coincidencias más relevantes, de acuerdo

con el algoritmo del motor de búsqueda. Algunos de los Metabuscadores más populares de la Web son:

- **Vivísimo/Yippy/Clusty** (ahora se llama *yippy* y anteriormente fue *clusty* [<http://search.yippy.com/>]) utiliza algoritmos de *clustering*, organizando las coincidencias semánticas encontradas en directorios. Fue creado por investigadores de la Universidad de Carnegie-Mellon. Dispone de algunas opciones de búsqueda avanzada: búsqueda de frases exactas, operadores booleanos, campos de búsqueda (dominio, host, título, URL, etcétera).
- **Mamma** [<http://mamma.com>] muestra los resultados de una manera uniforme de acuerdo con la relevancia.
- **MetaCrawler**. [<http://www.metacrawler.com/>].
- **Dogpile** [<http://www.dogpile.com/>].
- **Ixquick** [<http://www.ixquick.com/>] Multilingüaje.
- **Ez2Find** [<http://Ez2Find.com/>] utiliza varios motores de búsqueda y directorios. También busca en una pequeña parte de la Web invisible a través de su función de “Búsqueda Avanzada”.
- Otros Metabuscadores interesantes son los siguientes: **InfoGrid** [<http://www.infogrid.com/>]. **IBoogie** [<http://iboogie.com/>] tiene un diseño minimalista y usa algoritmos de *clustering*. **Fazzle** [<http://www.fazzle.com/>] que usa un interesante algoritmo de ranking. **Query Server** [<http://www.queryserver.com/web.htm>] busca en una lista de 11 motores de búsqueda (exceptuando Google). Es otro ejemplo de uso de algoritmos de *clustering*. **Meta Bear** [<http://www.metabear.com/>] que ofrece los resultados más relevantes para sitios internacionales y rusos. **Experts Avenue** [<http://www.expertsavenue.com/>] que activa la traducción de lenguajes online para páginas Web. **EmailPinoy** [<http://www.emailpinoy.com/>] de las Filipinas. **Search 66** [<http://www.search66.com/>] agrupa páginas del mismo dominio. **Internav** [<http://www.internav.com/>]. **Metengine** [<http://www.metengine.com/>](Antigua). **One2Seek** [<http://www.one2seek.com/>]. **Ithaki** [<http://www.ithaki.net/>]. **meta EUREKA** [<http://www.metaeureka.com/>] (Holanda). **7 Meta Search** [<http://www.7metasearch.com/>]. **Bytedog** [<http://www.bytedog.com/>] (Canada). **il motore** [<http://www.ilmotore.com/>] (Italia). **ApocalX** [<http://www.search.apocalx.com/>] (Francia)...

También hay que tener en cuenta los motores que podrían considerarse *pseudo* Metabuscadores: Son aquellos que únicamente agrupan los resultados según el motor de búsqueda utilizado en una larga lista (tales como **qb Search** [<http://www.qbsearch.com/>], **Better Brain** [<http://www.betterbrain.com/>], **My Net Crawler**



[<http://www.mynetcrawler.com/>] y **Search Wiz** [<http://www.searchwiz.com/>]), y aquellos que abren un explorador diferente para cada motor de búsqueda (como por ejemplo **The Info** [<http://www.theinfo.com/>], **Net Depot** [<http://www.netdepot.org/>], **Alpha Seek** [<http://www.alfaseek.com/>] o **Express Find** [<http://www.expressfind.com/>]).

Este trabajo se centra en Metabuscadores en Internet debido a que las páginas Web son el principal punto de interés en la mayoría de los motores de búsqueda general, además de que gran parte de los que han sido descritos en este documento y que se han desarrollado también se centran en ellas. El diseño e implementación de estos tipos de sistemas se basa habitualmente en un proceso de indexación que consiste en la creación de un índice de términos y otras características que contienen las páginas Web a las que el buscador puede o quiere acceder. El proceso de indexación es implementado generalmente como una tarea off-line (sin conexión), y el índice se almacena en una gran base de datos (normalmente distribuida) a la que se accede cada vez que el usuario realiza una consulta.

Los algoritmos de evaluación de los Metabuscadores se tienen que enfrentar a problemas diferentes que los buscadores tradicionales. El principal problema se debe a la fusión de las listas de documentos evaluados que cada buscador devuelve. Las valoraciones producidas por diferentes buscadores normalmente no son comparables ya que se calculan frecuentemente utilizando alguna métrica en función de la distancia.

## **Agentes inteligentes**

Un agente es una entidad software que recoge, filtra y procesa información contenida en la Web, realiza inferencias sobre dicha información e interactúa con el entorno sin necesidad de supervisión o control constante por parte del usuario. Estas tareas son realizadas en representación del usuario o de otro agente. Hay que distinguir los agentes inteligentes de los buscadores inteligentes. Estos últimos, incorpora información semántica en el proceso de búsqueda para mejorar los resultados de la búsqueda, normalmente la precisión, utilizando en la búsqueda los recursos previamente indexados. Sin embargo, un agente inteligente recorre la Web a través de los enlaces entre recursos (hiperdocumentos, ontologías,...) en busca de aquella información que le sea solicitada, pudiendo además interactuar con el entorno para el cumplimiento de tareas encomendadas. Los agentes pueden realizar funciones de búsqueda, discriminación y selección.

Así es habitual en la literatura distinguir los siguientes tipos de agentes:

- *Agentes vigilantes*: de forma autónoma buscan información específica y pueden utilizarse para elaborar versiones personalizadas de los periódicos según los intereses del lector. Ejemplos: Personal Journal (Dow Jones), JobCenter, Personal View (Ziff Davies).
- *Agentes ayudantes*: actúan sin intervención humana. Se suelen emplear para la gestión de red y para las funciones normales de mantenimiento. Ejemplo: LANAlert.
- *Agentes aprendices*: aprenden a ajustar sus prestaciones al modo de actuar de su usuario. Ejemplos: Firefly.
- *Agentes compradores*, capaces de comparar precios y determinar qué producto ofrece las mejores condiciones. Ejemplo: BargainFinder.
- *Agentes de recuperación de información*: buscan formas inteligentes de recoger información y son capaces de reducir la sobrecarga de información en la localización de documentos comprimiéndolos o resumiéndolos. Ejemplos: Architext, ConText, Autonomy.

Algunas de las características deseables de los agentes son las siguientes:

- Poseer un nivel de inteligencia suficiente para aprender.
- Autonomía: La autonomía dependerá del grado de interactividad que se precise entre el usuario y el servidor.
- Movilidad: han de poder navegar por las redes y acceder a servidores.
- Modularidad: Permite reutilizar el agente y reducir la complejidad de los problemas.
- Comunicación: Tienen que comunicarse con otros agentes para poder trabajar en entornos distribuidos.
- Fiabilidad: Los usuarios sólo aceptarán a los agentes si éstos son de confianza.

Con objeto de facilitar la comunicación entre agentes, se han ideado herramientas para construir bases de conocimiento a gran escala que sean compartibles y reutilizables:

- KIF (*Knowledge Interchange Format*). Es un lenguaje formal para el intercambio de conocimiento entre programas dispares (escritos por diferentes programadores, en diferentes momentos, en diferentes lenguajes, etc.).
- KQLM (*Knowledge Query and Manipulation Language*). Es un formato de mensaje y un protocolo para manejar mensajes con el

objetivo de soportar el conocimiento compartido entre agentes. Es además un interfaz de comunicación entre agentes; está enfocado a las operaciones que los agentes usan para comunicarse.

Muchos servicios de búsqueda de Internet están ya ofreciendo información personalizada. Por ejemplo My Yahoo!, que pregunta al usuario cuando se da de alta por sus temas de interés, aficiones, edad, sexo, en qué país y región vive, empresas cuya cotización en bolsa le interesa, y con toda esta información confecciona unas páginas que incluyen enlaces a noticias, otros servidores, y recursos disponibles en Internet que tratan de aquello que interesa al usuario.

Una buena clasificación establecida por Casasola (1997) divide los agentes inteligentes según 4 dominios de aplicación:

- Asistentes en la navegación (*Browsing Assistants*): Este tipo de agentes monitorizan la actividad del usuario y recomiendan documentos facilitándole la búsqueda.
- Asistentes de búsqueda (*Search Assistants*): En esta categoría entran aquellos agentes que aportan nuevas características al proceso de búsqueda orientadas a la comunicación entre agentes.
- Agentes de emparejamiento y comparación (*Matchmaking and Comparison Agents*): Se encargan de monitorizar sitios que cambian con frecuencia para resumir su contenido y extraer relaciones entre sus componentes que permitan mantener información actualizada.
- Agentes de filtrado (*Filtering Agents*): Su función consiste en seleccionar de entre grandes volúmenes de información aquella más útil para el usuario.

## **Posicionamiento Web - SEO (Search Engine Optimization)**

La amplia utilización de los motores de búsqueda hace indispensable para el éxito o fracaso de un negocio poder aparecer entre las primeras páginas de los buscadores para conseguir un mayor número potencial de clientes. Las empresas que aparecen en las primeras posiciones de una consulta relacionada con su dominio de negocio tienen una mayor posibilidad de adquirir nuevos clientes. Este importante fenómeno ha favorecido la aparición de técnicas y sistemas que intentan mejorar la posición de una página Web en el *ranking* de los buscadores. SEO se define como el proceso de intentar maximizar la exposición de un sitio en varios motores de búsqueda y directorios, mediante la utilización de palabras claves y frases específicas. Este proceso también se denomina *Posicionamiento Web*. La mecánica consiste principalmente

en realizar cambios al sitio (título de la página, desarrollo de un contenido rico en palabras clave importantes, utilización de *metadatos*) con la finalidad de hacerlo más atractivo para los motores de búsqueda.

En la construcción de la página optimizada juegan factores como la localización de las palabras clave, su frecuencia, las *metaetiquetas*, el tamaño del texto, etc. Algunas de las principales técnicas utilizadas son la repetición de palabras clave al comienzo de la página. Esto, unido a la especial atención que suelen prestar algunos buscadores al comienzo de la página a la hora de indexar, permite asociar fuertemente esa página con los términos repetidos. Pero esta técnica es poco atractiva para los usuarios, por lo que suelen incorporarse en etiquetas ocultas o metaetiquetas.

Algunos de los factores económicos que favorecen la utilización de las técnicas de SEO son, según la empresa Vertexera (2011):

- Los usuarios utilizan los buscadores y la búsqueda de productos más frecuentemente que cualquier otra actividad de Internet, excepto el correo electrónico.
- Una buena posición en el *ranking* de un buscador es 2 ó 3 veces más efectivo a la hora de generar ventas que la utilización de *banners*.
- El 42% de la gente que realizó una compra *on-line* llegó al sitio a través de un buscador.
- El método promocional de sitios Web mejor calificado por los Webmasters es el posicionamiento en los motores de búsqueda.

Aunque la técnica de SEO no se puede considerar un problema en sí misma, si que es una forma de modificar el cálculo de la relevancia de una página en un buscador. Para poder mejorar la posición es necesario un proceso de prueba y error durante el que se comprueban las modificaciones introducidas. Evidentemente, para la modificación de la posición de una página en un buscador es necesario conocer el algoritmo que utiliza para evaluar cada página. Por ejemplo, Google se basa en la popularidad de una página y en los términos que se utilizan a la hora de enlazarla, entre otros factores. Los buscadores también proporcionan servicios de pago que permiten a los sitios aparecer en posiciones relevantes. Este tipo de servicios se denomina enlaces esponsorizados. Algunos ejemplos de sistemas de SEO son Vertexera<sup>16</sup> o Spider Food<sup>17</sup>.

---

<sup>16</sup> <[www.vertexera.com/](http://www.vertexera.com/)>.

<sup>17</sup> <[www.spider-food.net/](http://www.spider-food.net/)>.

## CAPÍTULO III

### **El lenguaje y los usuarios en los buscadores Web**

Un sistema de recuperación de información se sustenta en el hecho de que la información puede organizarse y representarse para su recuperación, y que las necesidades de información tienen alguna característica que se repite. Se asume que los documentos tienen alguna parte expresada de forma textual. Pero no sólo el contenido de los documentos determina la calidad en el proceso de recuperación de información, sino que el comportamiento del usuario y la propia naturaleza de Internet, junto con una adecuada utilización del lenguaje natural, influirán drásticamente en la calidad del proceso de búsqueda. El usuario muchas veces no tiene la suficiente información para expresar lo que quiere. Además, en algunas situaciones la información relevante se reconoce sólo cuando se encuentra y examina, no antes. Con bastante frecuencia, el proceso de búsqueda de los usuarios para cubrir sus necesidades de información se puede describir acertadamente mediante la frase de Bruza (1993): “No sé lo que estoy buscando, pero lo sabré cuando lo encuentre”.

La concepción y utilización particular del lenguaje natural del que hacen uso los distintos usuarios es uno de los principales problemas a la hora de buscar información. Los documentos contienen texto, que es una representación escrita del lenguaje natural. Además, el usuario también debe expresar lo que busca de forma textual para que sea comprensible para el sistema de recuperación de información. Por tanto, el problema derivado de la utilización particular del lenguaje natural se traslada también a su representación escrita, es decir, al texto. Un ejemplo de los problemas derivados de la variabilidad del lenguaje se produce cuando dos documentos tratan del mismo concepto pero utilizando dos palabras diferentes a la hora de nombrarlo, siendo por tanto indexados utilizando cada palabra concreta. Si un usuario busca información sobre este concepto, si no tiene presente que se puede describir el mismo concepto mediante dos palabras diferentes, será más complicado para él obtener sendos documentos. Esta situación ilustra que el usuario y sus

conocimientos también juegan un papel importante en el proceso de búsqueda de información en la Web.

## Semántica

La semántica estudia el significado de los signos y de sus combinaciones. Se puede distinguir entre semántica léxica, que se ocupa del sentido de las unidades del vocabulario, la semántica gramatical, cuyo cometido es estudiar las relaciones entre los elementos gramaticales que forman una oración y la semántica lingüística, que analiza el significado de los signos lingüísticos. Para los sistemas de recuperación de información, la semántica lingüística es a la que se presta mayor atención, ya que las otras semánticas sólo se fijan en la forma de las palabras y no en su significado.

Es evidente que cualquier signo lingüístico empleado hace referencia a una porción de la realidad, ya sea un objeto concreto o abstracto. Tal y como muestra el clásico triángulo semántico (figura 8) o triángulo de Odgen-Richards (Odgen, 72), se puede apreciar que existe una relación entre las palabras y la realidad a través del *sentido*. Existen diferentes representaciones de este triángulo que utilizan en sus vértices diferentes palabras para representar la misma idea.

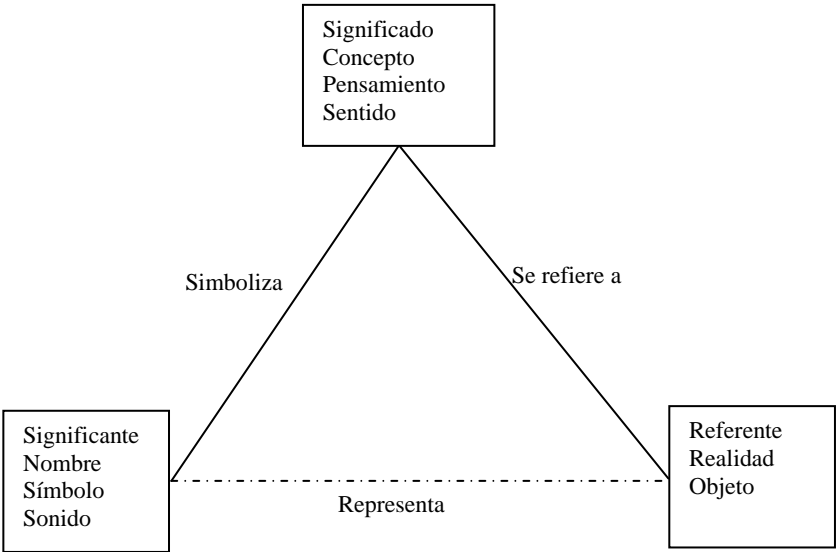


Figura 8: Triángulo semántico

En esta representación se puede observar que el significante de una palabra “simboliza” un significado, el cual “se refiere a” un elemento de la realidad o referente. La línea discontinua de la base del triángulo indica que no existe una relación directa entre el significante o palabra y el referente o elemento de la realidad: si no se conoce el significado de una palabra, no es posible establecer una relación que permita representar el referente perteneciente a la realidad mediante un símbolo o significante. Se puede definir por tanto el significante como la entidad física, típicamente en forma de imagen acústica, perceptible por los sentidos. El significado es un concepto, la idea que se representa en la mente al escuchar el significante. Por último, el referente es la realidad efectiva a la que se remite el signo. Por tanto, un aspecto relevante del lenguaje a tener en cuenta es el tipo de relación existente entre los significantes y el significado. Atendiendo al tipo de relación se pueden distinguir los siguientes casos:

- *Monosemia*: Se produce cuando la relación entre el significante y el significado es estrictamente de uno a uno y viceversa. Cuando se da esta situación se dice que el significante es monosémico. Este es el caso más favorable para los sistemas de recuperación de información, ya que una vez conocido el significante se puede conocer sin lugar a dudas su significado. Pero esta situación dista mucho de ser la habitual en la lengua común, produciéndose en su lugar el siguiente tipo de relación.
- *Polivalencia semántica o significación múltiple*: Este fenómeno se presenta cuando la relación entre significante y significado es de uno a varios. En este caso, un mismo significante puede simbolizar distintos conceptos. La significación múltiple se puede presentar de dos formas distintas:
  - *Homonimia*: cuando dos significantes originalmente distintos en su forma fonética y que simbolizan distintos conceptos llegan a coincidir a través del tiempo en un mismo significante (con igual forma fonética). Por ejemplo /llama/ cuando significa “masa gaseosa en combustión” viene de *flamma*, si se refiere a un rumiante sudamericano la palabra se toma del *quechua*, y cuando es forma del verbo llamar, proviene del verbo latino *clamare*.
  - *Polisemia*: cuando un significante adquiere un nuevo significado a lo largo del tiempo. Por ejemplo *Java* “era” una isla de Indonesia pero también el nombre de un lenguaje de programación.
- *Sinonimia*: Es el caso inverso a la significación múltiple. Se presenta cuando la relación entre significante y significado es de

varios a uno. Es decir, se produce cuando existen varios significantes diferentes en su forma pero que hacen referencia a un “mismo” significado. Sin embargo, hoy se acepta normalmente que la sinonimia concebida como relación precisa, esto es, como relación de equivalencia entre dos expresiones, o como identidad de significado, no existe en la práctica (Fernández, 00). Es decir, normalmente entre dos palabras consideradas como sinónimas existen ligeros matices que acentúan alguna característica del mismo concepto.

La existencia de significantes idénticos en la forma y diferente en el significado es un fenómeno común, al parecer ventajoso para la memoria, que se ve liberada de tener que retener una palabra diferente para cada concepto nuevo que se produzca. La existencia de sinonimia aporta variedad a la utilización del lenguaje, evitando la repetición excesiva de las palabras e incorporando diferentes matices en función de la expresividad, énfasis o intención de una comunicación. Pero es necesario disponer de una serie de mecanismos que permitan desambiguar los casos de significación múltiple para obtener una comunicación efectiva. Este proceso de desambiguación se consigue teniendo en cuenta el resto de significantes que intervienen en una comunicación, y que influirán en cierta medida en la decisión de seleccionar alguno de los posibles significados. Es decir, el significado de una misma palabra depende del “contorno lingüístico” que la envuelve y que determina su significación. Debido a este fenómeno, se pueden distinguir dos tipos de significado: referencial o contextual. Una palabra tiene un significado referencial cuando se “refiere” a su relación convencional con la realidad. Los significados referenciales son aquellos que se pueden encontrar en un diccionario. En cambio, el significado contextual es el que adquiere la palabra dentro de un contexto, cuando amplía, restringe y aún transforma el significado referencial.

Por tanto, el contexto es un elemento determinante en el acto comunicativo, y viene determinado por los actos comunicativos anteriores y posteriores. Además, es el criterio utilizado para determinar uno de los posibles significados y permitir descartar el resto. El contexto se suele definir en base a todas las propiedades de una situación social que son necesarias para la producción, interpretación y funciones del texto y la conversación. Es decir, que no sólo intervienen factores lingüísticos, sino también sociales y culturales. Manejar el contexto en un sistema de recuperación de información es una labor excesivamente compleja y que requiere de información previa que actualmente sólo poseen las personas, adquirida a través de su propia experiencia. Sin embargo, si que existen buenos diccionarios o tesauros



que permiten conocer las distintas acepciones de una misma palabra, conocer los significados referenciales de las palabras. El inconveniente de estos diccionarios es que la mayoría están diseñados para su utilización por personas, lo que implica que atienden a un único criterio de ordenación (típicamente el alfabético) y que carecen de una estructura que establezca un criterio que permita conocer las relaciones entre varias palabras sin atender a su forma (sustantivo, verbo, etc.), sino a su semántica. Afortunadamente, existen algunas excepciones como son por ejemplo WordNet<sup>18</sup>. Una persona se decanta por un significado u otro de una palabra con significación múltiple en función de las posibles relaciones semánticas que se puedan establecer entre los significantes participantes de una comunicación. Por ejemplo, ante la expresión: “Necesito un monitor barato”, en función del contexto o la situación “monitor” se referirá a “pantalla” o bien a “entrenador”. Si se modifica la expresión anterior aportando más información obtenemos: “Se me ha estropeado el ordenador. Necesito un monitor barato”, donde se puede apreciar que el significante “monitor” se refiere al significado “pantalla”. Esta afirmación se puede realizar debido a que todo el mundo sabe que un monitor es la pantalla de un ordenador. Por tanto, basándose en este conocimiento previo y en el contexto, se puede determinar el sentido correcto de “monitor”. Se puede apreciar que existen tres factores básicos necesarios para poder realizar esta desambiguación correctamente: un contexto, algún tipo de relación semántica y la necesidad de un conocimiento previo que nos permita conocer esa relación semántica.

Para que exista alguna relación semántica entre varios significantes, es necesario que pertenezcan a algún campo semántico que contemple esa relación. Pero es muy difícil construir alguna estructura que contemple la totalidad de las posibles relaciones existentes en la realidad. Existen algunos esquemas de representación formalizados que intentan representar el conocimiento para posteriormente poder obtener ese tipo de relaciones. Estos esquemas, cuyo primer impulsor fue Quillian (1968) con su memoria semántica, se denominan *redes semánticas*, y han sido muy utilizados en Inteligencia Artificial para representar las relaciones entre conceptos de un determinado ámbito. El buen funcionamiento de muchos sistemas de recuperación de información depende de que los usuarios introduzcan las palabras correctas. Los usuarios nuevos o esporádicos frecuentemente utilizan palabras inadecuadas. Los usuarios utilizan habitualmente una sorprendente cantidad de términos para referirse a conceptos similares

---

<sup>18</sup> <wordnet.princeton.edu>.

o relativos al mismo tema. Por ejemplo, en los directorios donde los documentos se incluyen en categorías de forma manual, puede suceder que categorías similares pertenecientes a una misma rama confundan al usuario y no encuentre en ella lo que esperaba. El problema es que la persona que asigna un documento a una categoría concreta puede dar más importancia a unas palabras determinadas o tener una concepción diferente a la del usuario que realiza la búsqueda, es decir, cada usuario tiene una preferencia personal a la hora de utilizar unas u otras palabras. Según Furnas (1987), en la elección espontánea de una palabra para objetos de cinco dominios, la probabilidad de que dos personas escojan el mismo término está por debajo de un 20%.

Cuando un autor escribe un documento, utiliza un vocabulario específico y personal, que en muchos casos no coincidirá con el utilizado por otras personas. Puede darse el caso de que otra persona utilice otras palabras para describir lo mismo, con un sinónimo, un alias o una frase explicativa. Pues bien, los sistemas de recuperación de información cuando indexan un documento, lo hacen atendiendo exclusivamente a los términos que forman el documento, salvo excepciones como por ejemplo el modelo FIS-CRM (Olivas, 03), que se explica con detalle en el capítulo 5. Cuando un usuario realiza una consulta elegirá un conjunto de términos que para él son representativos del concepto que busca. Si los términos utilizados en la consulta son distintos a los que forman el documento, es muy probable que ese documento no sea recuperado o que lo haga con un grado de relevancia muy bajo.

Otro problema con el vocabulario es la distribución geográfica de los usuarios, lo que se denominan *variedades diatópicas*, ya que los conceptos o ideas y sus vocabularios asociados pueden evolucionar o cambiar a lo largo del tiempo. Es decir, este problema se acentúa entre comunidades que hablan la misma lengua, pero dispersas geográficamente. Por ejemplo, para un argentino la palabra “frutillas” se refiere a la palabra “fresas” para un español. Además, este problema también se produce cuando se traducen palabras de un idioma a otro común, dando como resultado traducciones distintas. Por ejemplo, “fuzzy logic” se puede encontrar traducido como “lógica difusa” o “lógica borrosa”.

Existen distintas propuestas para tratar de solucionar este problema. Casi todas ellas se basan en la construcción de estructuras de conocimiento que contemplen las relaciones entre los distintos términos para ser tenidos posteriormente en cuenta. Otras técnicas que intentan paliar este problema son los mecanismos de expansión de consulta y más recientemente, el modelo FIS-CRM (olivas, 03), que

contempla los términos relacionados conceptualmente, aunque no presentes en el documento, a la hora de indexar.

Otro aspecto muy importante que afecta a la recuperación de información es el comportamiento del usuario. Cada usuario es distinto de los demás en sus motivaciones, suposiciones, conocimientos y experiencia. Este cúmulo de circunstancias afecta a la forma en la que cada usuario utiliza un sistema de recuperación de información. Los principales problemas que afectan al usuario en su interacción con el sistema son la forma en la que especifica su consulta y la forma en la que interpreta la respuesta proporcionada por el sistema. Como se ha visto a lo largo de este trabajo, los usuarios no suelen aprovechar al máximo las posibilidades que ofrecen las herramientas de búsqueda. La principal deficiencia es el bajo número de términos que los usuarios utilizan en sus cadenas de búsqueda. La media de utilización, según múltiples trabajos está en torno a los 2 términos por consulta. La utilización de pocos términos repercute en una menor precisión, debido a la desviación que puede causar la polisemia y a la inexistencia de otros términos que permitan concretar la búsqueda.

Otro factor que afecta a la obtención de buenos resultados es la poca realimentación que los sistemas de búsqueda reciben del usuario. Para estudiar este fenómeno, hay que tener en cuenta el concepto de sesión, que es el periodo de tiempo durante el cual el usuario realiza consultas al sistema de forma continuada. No suele durar más de 30 minutos y el resultado es que el número medio de consultas por sesión es muy bajo, no suele pasar de 4. Además, el número medio de pantallas de resultados del buscador que los usuarios visitan es muy bajo, no se suele pasar de la tercera. Si se considera que la mayoría de los usuarios solamente comprueban una o dos pantallas del resultado, se puede determinar que los usuarios buscan de una forma muy limitada: sencillamente abriendo los documentos con el título y descripción que mejor se ajusta a sus necesidades. La forma en la que se proporciona la retroalimentación suele ser modificando alguno de los términos de la consulta, aunque también se suele hacer cambiando totalmente la consulta, añadiendo términos, eliminando términos o modificando los operadores de búsqueda. La utilización de los operadores de búsqueda suele ser muy baja. Los operadores más usados suelen ser las *comillas* (frase exacta), y algún operador *booleano*, pero en no más de un 5% del total de consultas que envían usuarios “normales” (por ejemplo en entornos científicos este porcentaje aumenta considerablemente).

Es de esperar que si un usuario no utiliza de la mejor forma posible las posibilidades de los buscadores, estos no puedan proporcionar los mejores resultados. No obstante, es posible encontrar diversos sistemas o herramientas que tratan de paliar este inconveniente utilizando distintas técnicas, por ejemplo la expansión de consulta

interactiva en metabuscadores, que ayudan al usuario en el proceso de búsqueda. Se podría concluir que la interacción de los usuarios con los motores de búsqueda en la Web es corta y limitada para mejorar estos factores y el comportamiento humano parece necesaria una nueva generación de herramientas de búsqueda Web que trabajen más cercanos a los usuarios para ayudarles a conseguir mayor eficacia en la búsqueda para resolver sus problemas de información.

Los sistemas de recuperación de información generalmente poseen un interfaz de búsqueda que permite indicar al sistema lo que el usuario quiere buscar. Normalmente el usuario plasmará sus necesidades de búsqueda utilizando un conjunto de palabras, que se llamará cadena de búsqueda, utilizando a veces operadores de búsqueda. Pero la cadena de búsqueda introducida por el usuario es sometida a transformaciones por el buscador en función de sus características particulares. El resultado de estas transformaciones es una expresión denominada *términos de búsqueda*, y suele consistir en un subconjunto de la cadena de búsqueda. Las transformaciones que sufre la cadena de búsqueda a menudo propician cambios en los resultados esperados por el usuario. Por tanto, un adecuado conocimiento de las características particulares del sistema de recuperación de información hace consciente al usuario de estas transformaciones. Esta situación provoca una desviación en las estrategias de búsqueda del usuario para intentar adecuar a sus intenciones de búsqueda los resultados. Algunas de las transformaciones de la cadena de búsqueda más comunes en las herramientas de búsqueda Web actuales son las siguientes:

- *Los buscadores suelen aplicar un operador booleano de búsqueda por defecto*: aunque los buscadores actuales no suelen basarse en el modelo booleano, si que permiten la utilización de operadores booleanos en la cadena de búsqueda. La utilización de los mismos suele determinar de forma importante los resultados, como se ha visto en los ejemplos de uso de operadores. Por tanto, la utilización de alguno de estos operadores por defecto cuando el usuario no indica ninguno puede afectar seriamente al resultado de la búsqueda, como por ejemplo Google, que por defecto usa el operador AND.
- *Eliminación de stopwords*: a la hora de indexar los documentos la mayoría de los buscadores ignoran las palabras prohibidas, por tanto, tampoco tendrá mucho sentido permitir la utilización de este tipo de palabras en la consulta del usuario, ya que no será posible recuperarlas. Por el contrario, actualmente existen formas de evitar este comportamiento utilizando el operador *comillas* de Google, que permite indicar exactamente lo que se busca.

- *Expansión de los términos con sufijos*: algunos buscadores consideran como términos relevantes a la hora de recuperar documentos los que compartan la raíz con alguno de los términos de búsqueda, como por ejemplo Yahoo.
- *Orden de los términos en el resultado de la búsqueda*: Algunos algoritmos de *ranking* prestan especial atención a la proximidad y orden de los términos de búsqueda. Evidentemente, si este aspecto se tiene en cuenta, el resultado será diferente cuando los mismos términos se utilizan en diferente orden. Esta situación se da en Google, donde por ejemplo, como se ha visto, se obtendrán diferentes resultados para la consulta “drive drunk” y “drunk drive”.

La solución de este problema causado por el desconocimiento del usuario de las transformaciones a las que se somete la cadena de búsqueda es complicada. Normalmente, los usuarios suelen utilizar un único buscador, que hace que con el uso se conozcan mejor sus características y por tanto el usuario adapte su forma de buscar. Se pueden aplicar diferentes tácticas para mejorar la eficacia de la búsqueda (se toman como base las que ya describió Marcia Bates (1979), pero actualizándolas a las condiciones actuales):

- *Términos de la consulta*: tácticas para ayudar en la selección y revisión de los términos que conforman la consulta. Por ejemplo, probar con variaciones léxicas de los términos, o examinar los documentos ya revisados para ver si se pueden encontrar otros términos útiles para la búsqueda, elegir los términos más específicos posibles o incluir sinónimos en la consulta.

Para esto puede ser de mucha ayuda el uso de las relaciones lingüísticas contenidas en *Wordnet*<sup>19</sup>: sinónimos, hipónimos, hiperónimos, palabras relacionadas, definiciones... Por ejemplo, si pedimos información a Wordnet sobre la palabra “house”, obtenemos lo que muestra la figura 9.

#### WordNet Search - 3.1

Key: "S:" = Show Synset (semantic relations, "W:" = Show Word (lexical relations

Display options for sense: (gloss) "an example sentence"

#### Noun

- **S:** (n) **house** (a dwelling that serves as living quarters for one or more families) "he has a house on Cape Cod"; "she felt she had to get out of the house"
  - [direct hyponym](#) / [full hyponym](#)
  - [part meronym](#)
  - [direct hypernym](#) / [inherited hypernym](#) / [sister term](#)
  - [derivationally related form](#)
- **S:** (n) **firm, house, business firm** (the members of a business organization that owns or operates one or more establishments) "he worked for a brokerage house"
- **S:** (n) **house** (the members of a religious community living together)

<sup>19</sup> <wordnet.princeton.edu>.

- **S:** (n) **house** (the audience gathered together in a theatre or cinema) *"the house applauded"; "he counted the house"*
  - **S:** (n) **house** (an official assembly having legislative powers) *"a bicameral legislature has two houses"*
  - **S:** (n) **house** (aristocratic family line) *"the House of York"*
  - **S:** (n) **house** (play in which children take the roles of father or mother or children and pretend to interact like adults) *"the children were playing house"*
  - **S:** (n) **sign of the zodiac**, **star sign**, **sign**, **mansion**, **house**, **planetary house** ((astrology) one of 12 equal areas into which the zodiac is divided)
  - **S:** (n) **house** (the management of a gambling house or casino) *"the house gets a percentage of every bet"*
  - **S:** (n) **family**, **household**, **house**, **home**, **menage** (a social unit living together) *"he moved his family to Virginia"; "It was a good Christian household"; "I waited until the whole house was asleep"; "the teacher asked how many people made up his home"; "the family refused to accept his will"*
  - **S:** (n) **theater**, **theatre**, **house** (a building where theatrical performances or motion-picture shows can be presented) *"the house was full"*
  - **S:** (n) **house** (a building in which something is sheltered or located) *"they had a large carriage house"*
- Verb**
- **S:** (v) **house** (contain or cover) *"This box houses the gears"*
  - **S:** (v) **house**, **put up**, **domiciliate** (provide housing for) *"The immigrants were housed in a new development outside the town"*

Figura 9: Búsqueda en Wordnet para la palabra “house”

También puede ser muy útil encontrar los términos más adecuados (y los que más usan los creadores de documentos y los usuarios al tratar de localizarlos), teniendo en cuenta, por ejemplo, las variedades diatópicas (mismo idioma en diferentes lugares), o la evolución de palabras/conceptos a través del tiempo. Para ello se pueden usar herramientas como *Google Trends*<sup>20</sup>, donde por ejemplo si buscamos la palabra “Alakrana” (nombre de un pesquero español secuestrado en aguas somalíes, palabra que probablemente nunca había sido usada antes del secuestro en búsquedas de Google), podemos ver la evolución de sus búsquedas en diferentes idiomas y lugares (figura 10).

<sup>20</sup> <[www.google.com/trends](http://www.google.com/trends)>.

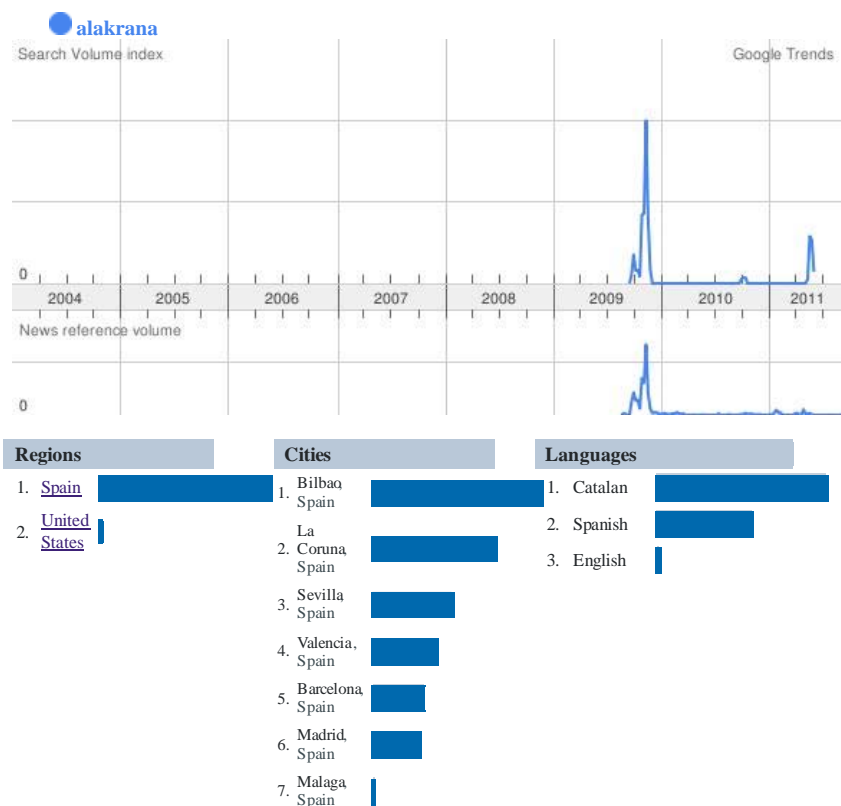
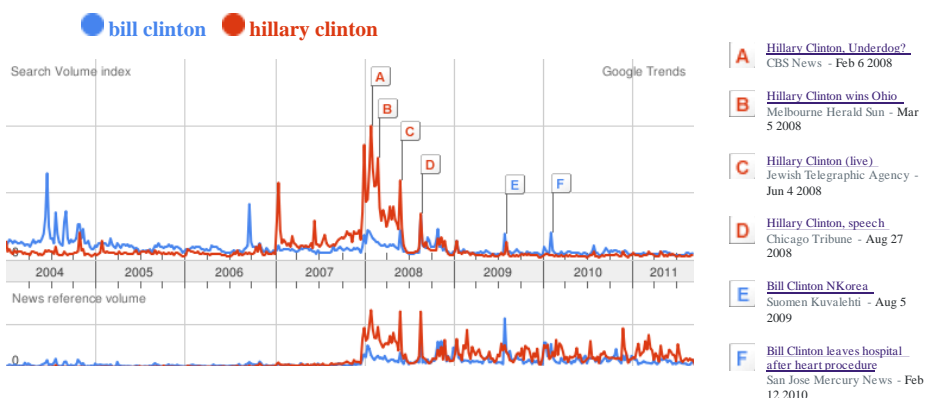


Figura 10: Búsqueda en Google Trends para la palabra “Alakrana”

Si quisiéramos comparar como han cambiado las búsquedas con respecto a Bill y Hillary Clinton en paralelo a sus respectivas proyecciones personales, en Google Trends encontraríamos lo siguiente (figura 11):



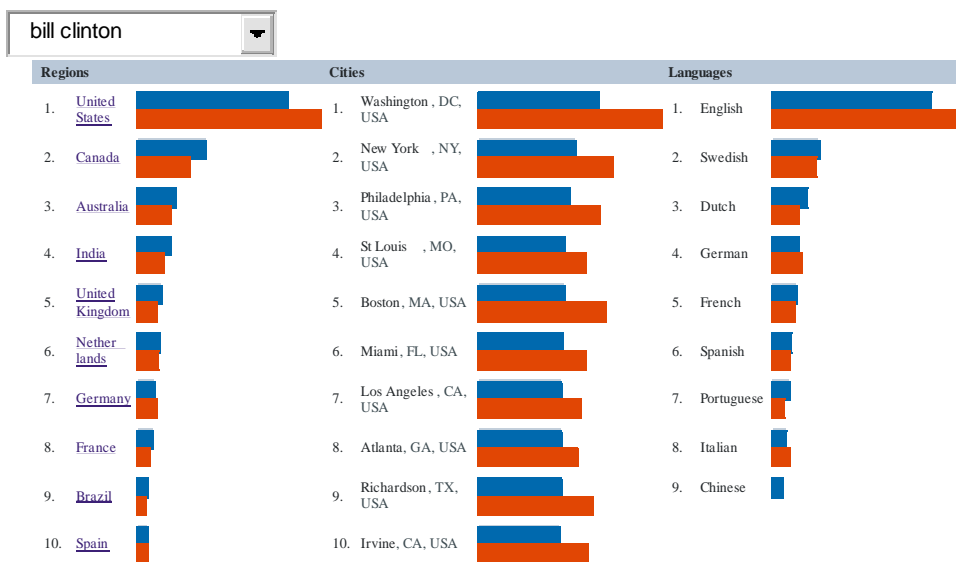


Figura 11: Búsqueda en Google Trends para “Bill Clinton” y “Hillary Clinton”

- *Monitorización de la búsqueda:* tácticas para no perder de vista el motivo de la búsqueda, manteniendo así una sesión de búsqueda más eficaz. Por ejemplo, revisar periódicamente la consulta original y compararla con la actual para ver si se sigue buscando lo mismo, o llevar la cuenta de las consultas que no han dado resultados satisfactorios para no repetirlos.
- *Estructura de archivo:* tácticas para navegar correctamente por el conjunto de documentos hasta el documento, fuente o información deseada. Por ejemplo, terminar de examinar un documento completamente antes de lanzarse a otras búsquedas que su contenido pueda sugerir, apuntando si es necesario esas nuevas “pistas” para luego tenerlas en cuenta o no, de forma que la consulta va sufriendo una evolución.

El usuario, según va adquiriendo más experiencia con los buscadores desarrolla unas conductas y técnicas que hacen que sus sesiones de búsqueda vayan siendo cada vez más eficaces. Por el contrario, los novatos tienden a volver a formular la consulta, mediante cambios poco significativos y con un mínimo o casi nulo uso de operadores, en un intento de conseguir una lista de resultados en la que a simple vista se obtenga la respuesta. Con los análisis de los *logs* de *Excite*<sup>21</sup>,

<sup>21</sup> <www.excite.com/>.



Amanda Spink (1999) intentó descubrir qué buscan los usuarios y obtuvo los siguientes grupos: Entretenimiento y recreo, Sexo y pornografía, Comercio, viajes, empleo y economía, Informática e Internet, Salud y ciencias, Gente, lugares y cosas, Sociedad, cultura, etnias y religiones, Educación y humanidades, Bellas artes, Gobierno, Otras. Además observó que según pasaba el tiempo, el uso de Internet para encontrar sexo y ocio disminuía en favor de un uso más comercial y menos frívolo. Es clásica la taxonomía que presentó Andrei Broder (2002) en la que determinó que mientras el uso de los sistemas de información de recuperación clásicos respondía a una necesidad de información, en el caso de la Web no ocurre así, constituyendo este tipo de consultas menos del 50%. De forma que estableció una clasificación según la cual las consultas pueden ser: de navegación, informativas y transaccionales. Posteriormente, Daniel E. Rose y Danny Levinson (2004) estudiaron y desarrollaron la taxonomía de Broder con el siguiente resultado:

- *De navegación*: el deseo es llegar a un sitio Web que el usuario ya tiene en mente, ya sea porque lo visitó en el pasado o porque asume que existe. El resultado esperado es una única página Web, aunque también pueden ser deseable otro conjunto de páginas sobre sitios Web similares. Por ejemplo, si se está buscando la página Web del periódico El Mundo, es asumible que también puedan ser de interés las páginas de otros periódicos de tirada nacional en España.
- *Informativas*: el deseo es obtener información mediante la lectura de documentos. En este caso quizá sea mejor un conjunto de documentos en vez de un único documento, para poder así contrastar la información.
  - *Dirigidas*: se quiere conocer algo en particular sobre un tema.
    - *Cerradas*: se quiere obtener la respuesta a una pregunta muy concreta.
    - *Abiertas*: se quiere obtener respuesta a una pregunta que puede tener varias respuestas.
  - *No dirigidas*: se quiere aprender “un poco” sobre un tema.
  - *Consejos*: se quiere obtener consejos, ideas, sugerencias o instrucciones.
  - *Locales*: el objetivo es encontrar algún servicio o producto en el mundo real.
  - *Lista*: se pretende encontrar una lista de sitios web similares.

- *De recursos* (sustituye y amplía la categoría de consultas transaccionales definida por Broder): el objetivo es obtener recursos (no información) disponibles en la Web.
  - *Descargas*: se quiere descargar un recurso para ser instalado o usado off-line.
  - *Entretenimiento*: se quiere obtener entretenimiento simplemente viendo el contenido de los documentos.
  - *Interacción*: el objetivo es interactuar con un recurso.
  - *Obtención*: se quiere obtener recursos para los cuales no sea necesario el uso del ordenador, por ejemplo, para ser impresos.

Una relación entre la taxonomía de Broder y los once tipos de consultas obtenidos por Amanda Spink fue descrita por Dirk Lewandowski (2006): las consultas informativas suelen ser sobre salud, ciencias, sociedad, cultura y religión; que las consultas de navegación suelen ser sobre sitios Web de gente, lugares, entretenimiento, comercio y economía; y que las consultas de recursos suelen ser sobre pornografía, entretenimiento e informática.

Al igual que con cualquier otro sistema de información, la realimentación proporcionada por el usuario puede llegar a ser muy útil para la mejora del sistema. Si bien la información más valiosa generalmente se obtiene preguntando directamente al usuario, es difícil de conseguir porque éste no quiere perder tiempo contestando encuestas o escribiendo sus opiniones. Si se quiere realimentación y no se desea depender de la generosidad del usuario, lo mejor es obtenerla implícitamente sin que éste lo note. Al utilizar un buscador el usuario realiza consultas, abre algunos de los resultados e invierte más o menos tiempo en ellos, vuelve a formular las consultas y lleva a cabo otras acciones. Cómo aprovechar todas estas acciones en el diseño del sistema ha supuesto un nuevo campo de investigación, pero no resulta tarea fácil dado el elevado nivel de ruido que estos conjuntos de datos suelen tener. Se ha usado realimentación implícita para mejorar el algoritmo de *ranking* (Agichtein, 06), (Joachims, 05), (Xue, 04) o expansión de consultas (Kelly, 03). Otros autores relacionaron las acciones de los usuarios con los objetivos pretendidos al realizar una consulta e investigadores de Microsoft (Fox, 05) estudiaron cuáles indican mejor su satisfacción.

Hoy en día la mayoría de los buscadores tienen la Web como espacio de búsqueda. Este motivo influye en los buscadores, ya que Internet posee una naturaleza particular y poco apropiada para su tratamiento por parte de los sistemas de recuperación de información tradicionales. Algunos de los principales problemas que presentan los

documentos de la Web a la hora de buscar información son los siguientes (Baeza-Yates, 00):

- *Distribuidos*: la naturaleza distribuida de la Web influye en la confiabilidad de las conexiones y el ancho de banda disponible.
- *Volátiles*: el volumen de información crece, así como su contenido, estimándose que el 40% de la Web cambia cada mes.
- *Dinámicos*: muchos documentos se suelen construir utilizando el contenido de bases de datos.
- *Sin estructura*: La Web está formada por datos semi-estructurados sin una estructura constante.
- *Redundantes*: Hay muchas páginas duplicadas y se estima en un 30% el número de sitios replicados.
- *Tipos heterogéneos*: Cada vez hay más formatos de representación de los documentos, y por tanto, es necesario conocerlo para poder estudiar su contenido e indexarlos correctamente. Además, hay documentos en distintos lenguajes y que utilizan distintos alfabetos.
- *Calidad heterogénea*: Al no haber una autoridad encargada de velar por la corrección de los documentos de Internet, es posible que existan documentos con información falsa, con errores ortográficos, mal redactados, etcétera.

Estas características hacen imposible para un buscador indexar la totalidad de la información que contiene la Web. Este tipo de sistemas serán incapaces de indexar los documentos generados de forma dinámica, como por ejemplo las páginas *asp* ó *php*. Los buscadores sólo son capaces de encontrar las páginas que previamente han indexado, por tanto, los buscadores no serán capaces de encontrar todas las páginas posibles. Los buscadores Web, mediante sus *crawlers*, comienzan indexando un conjunto de páginas (de las que tienen su URL), y van introduciendo en colas los enlaces que encuentran en las páginas que analizan para posteriormente indexarlas. Pero si una página no es enlazada por ninguna otra (o por una página no encontrada por el buscador) no será posible para el buscador encontrarla, y por tanto no podrá indexarla. También puede darse el caso de que existan islas de páginas enlazadas las unas con las otras, pero que no reciben enlaces fuera de su isla de páginas. Si un buscador no conoce ninguna página que pertenezca a esa isla, no será capaz de encontrar el resto.



## CAPÍTULO IV

### **Las técnicas de *Soft Computing* en la recuperación de información**

Desde que en el año 1965 el profesor Lotfi A. Zadeh introdujo la Lógica Borrosa o Difusa (*Fuzzy Logic*), muchas han sido sus aplicaciones, las más importantes en el campo del control industrial (lavadoras Bosch con sistema ECO-Fuzzy, ABS de Nissan, Aire Acondicionado Mitsubishi...). Pero, ya desde sus inicios, la intención del profesor Zadeh era introducir un formalismo capaz de representar y manipular la incertidumbre e imprecisión inherentes al lenguaje natural.

Por otro lado, la mayor parte de la inmensa cantidad de información contenida en Internet, está almacenada en documentos textuales en lenguaje natural en multitud de idiomas.

El profesor Zadeh describe qué es *Soft-computing* en estos términos<sup>21</sup>:  
“*What is soft computing? Soft computing differs from conventional (hard) computing in that, unlike hard computing, it is tolerant of imprecision, uncertainty and partial truth. In effect, the role model for soft computing is the human mind. The guiding principle of soft computing is: Exploit the tolerance for imprecision, uncertainty and partial truth to achieve tractability, robustness and low solution cost. The basic ideas underlying soft computing in its current incarnation have links to many earlier influences, among them my 1965 paper on fuzzy sets; the 1973 paper on the analysis of complex systems and decision processes; and the 1979 report (1981 paper) on possibility theory and soft data analysis. The inclusion of neural network theory in soft computing came at a later point. At this juncture, the principal constituents of soft computing (SC) are fuzzy logic (FL), neural network theory (NN) and probabilistic reasoning (PR), with the latter subsuming belief networks, genetic algorithms, chaos theory and parts of learning theory. What is important to note is that SC is not a*

---

<sup>21</sup> <[www-bisc.cs.berkeley.edu](http://www-bisc.cs.berkeley.edu)>.

*melange of FL, NN and PR. Rather, it is a partnership in which each of the partners contributes a distinct methodology for addressing problems in its domain. In this perspective, the principal contributions of FL, NN and PR are complementary rather than competitive.*

*Implications of soft computing. The complementarity of FL, NN and PR has an important consequence: in many cases a problem can be solved most effectively by using FL, NN and PR in combination rather than exclusively. A striking example of a particularly effective combination is what has come to be known as neurofuzzy systems. Such systems are becoming increasingly visible as consumer products ranging from air conditioners and washing machines to photocopiers and camcorders. Less visible but perhaps even more important are neurofuzzy systems in industrial applications. What is particularly significant is that in both consumer products and industrial systems, the employment of soft computing techniques leads to systems which have high MIQ (Machine Intelligence Quotient). In large measure, it is the high MIQ of SC-based systems that accounts for the rapid growth in the number and variety of applications of soft computing - and especially fuzzy logic. The conceptual structure of soft computing suggests that students should be trained not just in neural network theory or fuzzy logic or probabilistic reasoning but in all of the associated methodologies, though not necessarily to the same degree. This is the principle which guides the BISC Seminar on Soft Computing and the course Fuzzy Logic, Neural Networks and Soft Computing which I teach at present. The same applies to journals, books and conferences. We are beginning to see the appearance of journals and books with soft computing in their title. A similar trend is visible in the titles of conferences.”*

A la hora de clasificar las diferentes líneas de investigación relacionadas con la recuperación de información, y más en concreto, con las posibilidades de las técnicas de *Soft-Computing* en la recuperación de Información en Internet, se pueden utilizar diferentes criterios. Una posibilidad es realizar una clasificación en función de la parte del proceso de recuperación de información en la que están centradas. En este caso podemos distinguir los siguientes grupos de enfoques:

- Modelos de representación lógica de documentos.
- Lenguajes de especificación de consultas.
- Sistemas de evaluación de consultas.
- Sistemas de presentación y clasificación de resultados.
- Sistemas de retroalimentación de consultas.

Otra posibilidad es distinguir los enfoques según las técnicas utilizadas:

- Utilización de ontologías.
- Estudio de asociaciones y relaciones entre términos.
- Construcción y utilización de perfiles de usuarios.
- Utilización de algoritmos de *clustering* y clasificación.

Atendiendo al objetivo y el ámbito de aplicación de los sistemas de recuperación de información, podemos distinguir diferentes tipos de sistemas:

- Sistemas de búsquedas basados en consultas.
- Sistemas basados en directorios dinámicos.
- Sistemas de preguntas-respuestas (*Question-Answering systems*).
- Búsquedas conceptuales.

Todas estas categorías, podrían ser a su vez subdivididas en función de si en su realización se han considerado tecnologías típicas de *Soft-computing* como Lógica Borrosa y técnicas de Inteligencia Artificial. Las posibilidades de estas técnicas en el campo de la recuperación de información están claramente constatadas en numerosos estudios, como por ejemplo:

- F. Crestani, G. Pasi (2000). *Soft computing in information retrieval: Techniques and applications*, Physica Verlag, Series studies in fuzziness.
- M. Kobayashi, K. Takeda (2000). "Information retrieval on the web". *ACM Computing Surveys* 32(2), 144-173.
- G. Pasi (2002). "Flexible information retrieval: some research trends". *Mathware and Soft Computing* 9, 107-121.
- M. Nikraves (2002). "Fuzzy conceptual-based search engine using conceptual semantic indexing". *Proc. of the 2002 NAFIPS annual meeting*, 146-151.
- E. Herrera-Viedma, G. Pasi (2003). "Fuzzy approaches to access information on the Web: recent developments and research trends". *Proc. of the Third Conference of the EUSFLAT*, 25-31.

Por último señalar que el propio profesor Zadeh, en el seminario de BISC<sup>22</sup> (*Berkeley Initiative in Soft-Computing*) resalta la necesidad y actualidad de la línea propuesta:

*"Existing search engines—with Google at the top—have many remarkable capabilities; but what is not among them is deduction capability—the capability to synthesize an answer to a query from bodies of information which reside in various parts of the knowledge base. In recent years, impressive progress has been made in enhancing performance of search engines through the use of methods*

---

<sup>22</sup> Seminario: Web Intelligence, World Knowledge and Fuzzy Logic. Lotfi A. Zadeh, September 14; 2004, University of California, Berkeley.

*based on bivalent logic and bivalent-logic-based probability theory. But can such methods be used to add nontrivial deduction capability to search engines, that is, to upgrade search engines to question-answering systems? A view which is articulated in this note is that the answer is 'No'. The problem is rooted in the nature of world knowledge, the kind of knowledge that humans acquire through experience and education. It is widely recognized that world knowledge plays an essential role in assessment of relevance, summarization, search and deduction. But a basic issue which is not addressed is that much of world knowledge is perception-based, e.g., "it is hard to find parking in Paris," "most professors are not rich," and "it is unlikely to rain in midsummer in San Francisco." The problem is that (a) perception-based information is intrinsically fuzzy; and (b) bivalent logic is intrinsically unsuited to deal with fuzziness and partial truth. To come to grips with the fuzziness of world knowledge, new tools are needed. The principal new tool—a tool which is briefly described in their note—is Precisiated Natural Language (PNL). PNL is based on fuzzy logic and has the capability to deal with partiality of certainty, partiality of possibility and partiality of truth. These are the capabilities that are needed to be able to draw on world knowledge for assessment of relevance, and for summarization, search and deduction."*

A continuación se pretende describir cuál es el papel que puede jugar la *lógica borrosa* como técnica de *Soft-Computing* para mejorar la búsqueda en este tipo de herramientas. La *lógica borrosa* puede proporcionar herramientas para la extracción y uso de conocimiento procedente de tesauros y ontologías, permite formalizar sentencias e implementar capacidades de deducción en sistemas de tipo Pregunta - Respuesta, combinar valores borrosos y diferentes lógicas, mejorar algoritmos de *clustering* y manejar las diferentes arquitecturas de un Metabusador.

Los principales buscadores de la red basan sus criterios de búsqueda en aspectos léxicos y en algunos casos semánticos (Ricarte, 01) con respecto a los términos de la consulta. Debido a esto, son muchas las formas por las que se han tratado de mejorar los resultados de las búsquedas Web, y es aquí donde las técnicas de *Soft-Computing* tienen un papel importante (Herrera-Viedma, 03), (Pasi, 02). Prueba de ello es que en los últimos años se han propuesto múltiples soluciones basándose en este tipo de técnicas: soluciones dirigidas a la construcción de sitios flexibles y adaptables (basados en patrones Web, perfiles de usuario, patrones de acceso, patrones de comportamiento de usuarios...) que utilizan técnicas de *Data Mining* (Tang, 01), (Cooley, 97), (Martin-Bautista, 01); otras centradas en la



organización por grupos de documentos recuperados (destacando la importancia del uso de los algoritmos de *clustering* (Zamir, 01) en lugar de los algoritmos que se basan en los grupos temáticos predefinidos); o en otro tipo de aproximaciones que incluyen sistemas basados en lenguajes de consulta flexibles, o sistemas basados en reglas de asociación borrosa que ayudan al usuario a encontrar nuevos términos que utilizar en su consulta (Delgado, 03).

Por otro lado, existen distintos tipos de aproximaciones que se centran en la representación de documentos, para lo cual utilizan, en la mayoría de los casos, extensiones del *modelo espacio vectorial* estándar (Salton, 83). También es frecuente encontrar sistemas que utilizan las interrelaciones entre términos almacenadas en tesauros y ontologías como WordNet, para construir redes semánticas de grupos de palabras (Miller, 90). Los Metabuscaadores surgen como una herramienta muy prometedora cuyo objetivo es el de mejorar los resultados de la búsqueda Web, para ello utilizan varios de los mejores motores de búsqueda como Google o Yahoo para después hacer una selección de los mejores resultados dados por dichas fuentes. Todos estos tipos de sistemas son muy diferentes al que propuso Zadeh (2003), el cual se centraba en el desarrollo de sistemas Pregunta - Respuesta, un punto de vista muy interesante en problemas de recuperación de información.

Actualmente no hay muchos buscadores comerciales con propiedades borrosas. Tanto las técnicas de Soft-Computing en general como la de la Lógica Borrosa en particular pueden jugar un papel importante en los problemas relacionados con la búsqueda basada en tecnologías Web. A continuación se relatan varios aspectos que podrían ser mejorados a través de las técnicas de Soft-Computing.

Si hacemos la consulta “buscador borroso” en Google, serán muchos los resultados que aparezcan, pero seleccionando aquellos que consideremos más relevantes, podemos observar que la idea de borrosidad que algunos buscadores comerciales implementan radica única y exclusivamente en el uso de funcionalidades de *matching* sintáctico con propiedades borrosas, es decir, tratan de corregir las palabras posiblemente mal escritas por el usuario mediante el uso de algún diccionario específico que contenga el buscador o al cual puede tener acceso, y en el cual aparezcan las palabras que busca el usuario escritas de forma correcta. Como resultado, el buscador mandará una señal de aviso (texto escrito) que diría algo de la forma: ¿Quiso usted decir? Aunque realmente esta funcionalidad es borrosa, es demasiado pretencioso hablar de buscador borroso simplemente por este detalle. Más concretamente, en buscadores famosos, el operador borroso de búsqueda permite al usuario escribir un trozo de la palabra que busca

cuando no sabe a ciencia cierta como se deletrea la palabra completa. Por ejemplo, muchos de los términos médicos o farmacéuticos son difíciles de deletrear, quizás sabes como suena la palabra pero realmente no sabes como se escribe. Tanto la búsqueda borrosa como los operadores *wild card* permiten encarar este problema. Por ejemplo, el buscador *Netscape Search* implementa funcionalidades borrosas, así si un usuario buscara el famoso antibiótico “Amoxicilina” podría hacer uso del operador borroso de búsqueda. Escribiría el trozo de palabra que supiera deletrear con certeza acompañado del símbolo ~ y el resto final de la palabra que ya no sabe si lo está deletreando bien o mal. Así, si hiciéramos la consulta: “amoxi~lilina”, el buscador devolvería satisfactoriamente aquellos documentos que contuvieran coincidencias del término buscado. Es similar a la búsqueda basada en caracteres *wild card*. Es posible combinar búsquedas en las que intervengan diferentes variantes de una palabra o deletreos incorrectos. Por ejemplo, muchos buscadores permiten tres caracteres *wild card*: el símbolo del dólar (\$), la interrogación (?) y el asterisco (\*).

## **El papel de la lógica borrosa en los Metabuscadore**

La idea es que un buscador sea borroso en el momento que implemente búsquedas mediante aproximaciones semánticas, es decir, cuando incluya criterios de aproximación semántica a las consultas, no sólo sintácticas. A continuación se presentan algunos aspectos a considerar.

### **El uso de diccionarios de sinónimos y tesauros (ontologías): búsqueda conceptual**

Cuando un usuario realiza una búsqueda usando una única palabra, la búsqueda puede ser completada haciendo uso de un diccionario de sinónimos. El diccionario permite realizar búsquedas no sólo teniendo en cuenta la palabra original sino además sus sinónimos y pudiendo establecer grados de sinonimia que después serán tenidos en cuenta para calcular el grado de relevancia de los documentos recuperados como respuesta a la consulta realizada por el usuario.

El proceso de búsqueda puede además ser mejorado usando tesauros y ontologías. En la actualidad, existen muchas ontologías referentes a diferentes dominios, que pueden mejorar diversos aspectos en ciertas aplicaciones. Todas ellas han sido hechas a mano siguiendo diferentes

metodologías, tales como Methontology (Fernandez Lopez, 99) o las de Gruninger (1995). En la otra cara de la moneda está la construcción automática de ontologías que representa uno de los principales focos de investigación en la actualidad, donde hasta el momento los resultados obtenidos no son demasiados satisfactorios.

Posiblemente el tesoro más usado en la actualidad es WordNet (Miller, 95), el cual está basado en las relaciones semánticas entre diferentes palabras. Las principales relaciones que gestiona WordNet son las de sinonimia, hiponimia, hiperonimia, etc. Así, un conjunto de sinónimos de una palabra puede ser agrupado en grupos llamados *synsets*, y como consecuencia, una palabra polisémica podrá pertenecer a diferentes *synsets*. La relación de hiponimia (y la de hiperonimia) se da entre diferentes términos entre los que se puede establecer una jerarquía. Así por ejemplo, si la palabra *perro* “es un tipo de” *canino*, entonces el término *canino* es un hiperónimo del término *perro*. Este tipo de relaciones aportan importante información que permite expandir la consulta inicial del usuario incorporando información con carácter semántico al proceso de búsqueda, así como un mecanismo para la identificación del significado adecuado de los términos involucrados en la consulta. Este tipo de sistemas normalmente requiere un mecanismo especial de *matching*, como el algoritmo de *ontomatching* que proponen Kiryakov y Simov (1999) que compara los conceptos asociados a las palabras. Whaley (1999) propone otro sistema de búsqueda que permiten desambiguar significados analizando la probabilidad de que coocuran ciertos conceptos. La idea de Leacock y Chodorow (1998) trata el problema de la desambiguación estudiando el contexto local de las palabras y comparándolas con el contexto habitual de los distintos significados de las palabras. Este sistema requiere tener almacenado en un repositorio el contexto habitual de las palabras. Loupy y El-Bèze (2002) usan Wordnet y además proponen un sistema de desambiguación basado en el entrenamiento del sistema con corpus de documentos. Ramakrishnan (2004) estudia la desambiguación desde el punto de vista del Soft-Computing. En este caso, las palabras no son desambiguadas en un único sentido, más bien en un conjunto de sentidos con sus correspondientes grados de relevancia (obtenidos mediante *Redes de Creencia Bayesianas*). Lafourcade (2001) introduce el concepto de sinonimia relativa para definir un modelo de vectores basados en conceptos. En este modelo, un término puede ser representado por un vector conceptual, el cual es obtenido por combinación lineal de las definiciones de los conjuntos de conceptos. Este sistema requiere un repositorio de conceptos.

Un ejemplo, FIS-CRM (Garcés, 06) es un modelo para representar los conceptos contenidos en cualquier clase de documentos. Puede ser considerado como una extensión del modelo vectorial (VSM) (Salton, 83). Su principal característica es que se alimenta de información proveniente de un diccionario sinonímico borroso y varias ontologías temáticas. El diccionario almacena el grado de sinonimia entre cada pareja de sinónimos y las ontologías guardan el grado de generalidad entre una palabra y otras más generales que ésta. La forma de calcular el grado de generalidad entre términos es usando el método propuesto por Widyantoro y Yen (2001). La clave del éxito del modelo FIS-CRM (en su aplicación a varios metabuscadores) radica primeramente en la construcción de los vectores base de los documentos teniendo en cuenta el número de ocurrencias de los términos (lo que se denominan vectores VSM) y el posterior reajuste de los pesos de los vectores con el propósito de representar el peso de las ocurrencias de conceptos, haciendo uso de la información disponible en el diccionario de sinónimos y las ontologías temáticas. El proceso de reajuste conlleva el hecho de repartir las ocurrencias de un concepto entre los distintos sinónimos que convergen al concepto y que dan un peso a las palabras que representan un concepto más general que el que ellas mismas representan.

## **Sentencias de búsqueda y capacidades deductivas**

Si la búsqueda incluye frases, además del diccionario de sinónimos, los tesauros y ontologías, será necesario el uso de conectivas borrosas adecuadas, para poder discernir por ejemplo en una consulta de la forma “A y B” donde A y B tienen información común o cuando son totalmente independientes. Algo similar ocurre con la relación “A o B”. Otro aspecto deseable es que se mantenga el significado de las palabras que se tienen en mente mediante la relación de sinonimia, para elegir la mejor función de similitud.

Pero el problema puede ser aún mayor si trabajamos con relaciones causales. Primero, es muy difícil detectar una relación causal escrita (en un texto o una consulta). Por ejemplo, el texto podría ser: “Lluvioso y oscuro”, que podría ser entendido por una persona como: “Si el tiempo está lluvioso, el cielo se oscurece”. ¿Cómo puede un buscador distinguir la conectiva “y” y la causal? Ahora mismo es prácticamente imposible incluso si hay información relativa al contexto. Segundo, es muy difícil encontrar la función de implicación más adecuada para representar la sentencia (hay una gran variedad de implicaciones borrosas). La detección y gestión de relaciones causales

podrían ser muy importantes para desarrollar sistemas Pregunta-Respuesta.

Para detectar las relaciones causales que existen en una colección de documentos, un punto de comienzo podría ser la detección de frases condicionales, pero esto no es tarea fácil. Descartes nunca pensó que su frase “Pienso luego existo”, daría lugar a tal cantidad de conjeturas e interpretaciones años después. En realidad, lo que el quiso decir, “Primero pienso y luego soy persona”, o “Como tengo la capacidad de pensar, soy una persona”. Incluso en esta ocasión, donde la intención de Descartes parece clara cuando expresó su máxima, no es tan fácil de interpretar y representar la información expresada en lenguaje natural, especialmente cuando conlleva frases complejas y giros complicados.

Con el fin de encontrar frases condicionales, se han desarrollado en el marco de nuestro grupo de investigación algunos sistemas para la detectar estructuras y clasificar frases causales (Puente, 10), lo cual permite localizar, en términos de componente básicos (verbos, adverbios, giros lingüísticos, etcétera), algunas formas causales. Para realizar el análisis gramatical, tenemos por un lado que es posible separar ciertas relaciones causales considerando su forma verbal, mientras que por otro lado vemos que es posible separar las relaciones atendiendo a los adverbios que aparecen en las frases. Ambos análisis dan lugar a algunas reglas causales que pueden ser usadas para extraer conocimiento de forma automática. De igual manera, cualquier estructura puede ser dividida en dos subestructuras las cuales corresponden al antecedente y al consecuente de la relación causal, y a un parámetro que mide el grado de certeza, conjetura o conformidad de dicha relación causal. En otras palabras, no es lo mismo una frase de la forma: “Si gano la lotería, me compraré un coche”, en la cual no hay duda de que si el antecedente es verdadero el consecuente también lo será, que tener una frase que diga “Si hubiéramos comprado un boleto en Sacramento, podríamos haber ganado la lotería”, la cual deja muchas más dudas y conjeturas, y en la cual no se puede asegurar que el cumplimiento del antecedente garantice el cumplimiento del consecuente.

Pero esto sigue siendo un problema de *Procesamiento de Lenguaje Natural* muy complejo. Hay otras aproximaciones muy interesantes como por ejemplo la de Trillas (2004) para la representación de frases condicionales mediante implicaciones borrosas. Por otro lado también hay una idea basada en *Procesamiento de Lenguaje Natural y protoformas* que podría ser una prometedora línea de investigación, tal

y como propone el Profesor Zadeh<sup>23</sup> (texto traducido del que aparece en su versión original al principio de este capítulo):

“Los motores de búsqueda -con Google a la cabeza- tienen muchas capacidades, pero la que no está entre ellas es la capacidad de deducción, la capacidad de dar una respuesta a una consulta a partir de diversas fuentes de información que residen en distintos puntos de una base de conocimiento. En los últimos años, un progreso impresionante se ha logrado en el desarrollo de motores de búsqueda basados en la lógica bivalente y la teoría de la probabilidad basada en la lógica bivalente. Pero estos métodos, ¿pueden ser añadir capacidades deductivas no triviales a los buscadores, es decir, evolucionar los motores de búsqueda a sistemas Pregunta- Respuesta? A lo largo de estas líneas se demuestra que la respuesta es: ‘No’. El problema es adentrarse en la naturaleza del conocimiento del mundo, la clase de conocimiento que los humanos adquieren a través de la experiencia y la educación. Está ampliamente demostrado que el conocimiento del mundo juega un papel esencial en la afirmación de la relevancia, síntesis, búsqueda y deducción. Pero una idea básica que no es controlada, es que gran parte del conocimiento que tenemos del mundo está basado en la percepción, por ejemplo, “es duro encontrar aparcamiento en París”, “la mayoría de los profesores no son ricos” y “es poco probable que llueva a mediados de verano en San Francisco”. El problema es que (a) la información basada en percepción es intrínsecamente borrosa; y (b) la lógica bivalente es inadecuada para tratar la borrosidad y las verdades parciales. Para trabajar con la borrosidad del mundo real son necesarias nuevas herramientas. La principal herramienta -una herramienta brevemente descrita en su nota- es el Lenguaje Natural Preciso (LNP). LNP está basado en lógica borrosa y tiene la capacidad de trabajar con certeza parcial, posibilidades parciales y verdades parciales. Estas son las capacidades necesarias para moldear el conocimiento del mundo para asegurar la relevancia, y la síntesis, búsqueda y deducción”.

En este sentido también se han presentado propuestas desde nuestro grupo de investigación como las de Soto (2007).

## **Combinación de valores borrosos**

Un Metabusador tiene que llevar a cabo una combinación de lógicas (los algoritmos que cada buscador usa) con la intención de combinar

---

<sup>23</sup> Seminario: Web Intelligence, World Knowledge and Fuzzy Logic. Lotfi A. Zadeh, September 14; 2004, University of California, Berkeley.

las similitudes locales en una similitud global o un orden final. Pero las similitudes locales no están basadas en criterios borrosos. Así, el orden de las páginas relevantes no es aproximado. Normalmente, los Metabuscadores realizan las búsquedas de acuerdo a la importancia que ellos conceden a la pregunta del usuario, y dependiendo de ella, evalúan los resultados para incorporarla a la lista final de resultados devueltos. En este caso, se tienen en cuenta criterios de mercado, y no criterios lingüísticos. Podría resultar interesante aplicar este criterio, primero, como se indicó antes, para conseguir búsquedas semánticas borrosas. Segundo, para lograr que los Metabuscadores creen una lista final de páginas recuperadas conforme a la relevancia proporcionada con los grados de confianza asociados a la lista local obtenida, no sólo con las palabras tomadas de la consulta, y además usando términos relacionados (sinónimos...), las medidas de similitud usadas en el cálculo de las relaciones lingüísticas y los operadores *booleanos* borrosos utilizados en las búsquedas. Cada búsqueda podría responder a diferentes lógicas borrosas realizadas por los buscadores, las cuales el Metabuscador se encargaría de combinar para establecer una lista final de resultados. El uso de los enlaces proporcionados por el Metabuscador para dar el orden de los resultados devueltos, podría ser útil como banco de pruebas para experimentar distintas combinaciones hipotéticas de lógicas borrosas.

Otro problema que puede aparecer es cuando es necesario agregar varios valores borrosos provenientes de distintas fuentes. Dos palabras (*conceptos*) pueden tener más de una relación lingüística (cada una con su valor borroso), tales como la hiperonimia o la sinonimia. Por ejemplo “balompié” y “fútbol” son sinónimos pero el primer término es más general que el último. Una relación causal puede existir entre ambas palabras (*conceptos*). Más aún, una relación borrosa basada en la situación física de los términos (misma frase, párrafo, capítulo) podría ser tenida en cuenta. Entonces es necesario unir todos los valores borrosos en uno sólo para poder ser aplicado en tareas de representación y búsqueda. Cómo agregar estos valores borrosos es un problema que no tiene solución conocida. Actualmente se usan los operadores estándar OWA (Yager, 88) (o derivados de ellos como los LOWA y los WOWA). Otra posible solución podría ser el uso de operadores, tales como definidos por Castro y Trillas (1998).

## **Resultados del proceso de *clustering* borroso**

La clasificación de documentos o la categorización de textos (como se conoce en el mundo de la recuperación de información) es el proceso

por el que se asigna un documento a un conjunto predefinido de categorías basándose en el contenido del documento. Sin embargo, las categorías predefinidas en un repositorio real de documentos no son conocidas. Los métodos de *clustering* de textos pueden ser aplicadas para estructurar los conjuntos de documentos resultantes, así el usuario puede interactuar en el proceso de *clustering* de los documentos. En definitiva, el proceso de *clustering* permite lograr la estructuración de la colección de documentos en un número reducido de grupos, donde los documentos de cada grupo tienen el suficiente grado de similitud entre ellos. En consecuencia, podemos enumerar los principales elementos que influyen a la hora de organizar un repositorio:

- *Dimensionalidad*: El clasificador puede manejar espacios de elementos de miles de dimensiones, por lo que es necesaria la capacidad de gestionar espacios de datos escasos o métodos de reducción de dimensiones.
- *Eficiencia*: Los algoritmos de *clustering* documental deben ser eficientes y escalables. Además el método debería ser preciso en la clasificación de nuevos documentos entrantes.
- *Entendibilidad*: El método debe proveer una descripción comprensible de los *clusters* descubiertos.
- *Actualización*: El clasificador debe ser capaz de actualizarse con cada nuevo documento que es archivado en el repositorio.

Existen muchos algoritmos de *clustering* basados en técnicas de Soft-Computing en la literatura, por ejemplo el fuzzy C-means (Bezdek, 91), los mapas autoorganizados basados en arquitecturas de redes neuronales, como por ejemplo los Mapas de Kohonen (1984), etc. Actualmente los algoritmos de *Soft-Clustering*, como el propuesto por King-Ip (2001) y los interfaces de *clustering* dinámico, Grouper (Zamir, 01), son muy utilizados en las tareas de clasificación de los Metabuscadores.

## Arquitectura de un Metabuscador

Como se ha comentado ampliamente, existen ciertos aspectos que provocan una considerable reducción de la eficacia de los procesos de búsqueda. Además de las conocidas limitaciones provocadas por el uso de palabras clave, se suman la inexperiencia de los usuarios en el uso de los motores de búsqueda y sus herramientas adicionales. Así aparecen los Metabuscadores, como una alternativa para tratar de paliar la baja precisión lograda hasta ahora por los buscadores. Además del uso de Metabuscadores, también se proponen otras



alternativas para tratar de mejorar el grado de relevancia de los resultados obtenidos por los buscadores, tales como las *técnicas de expansión de la consulta* (Efthimiadis, 96) basada en el uso de términos relacionados semánticamente con los términos de la consulta original del usuario. Existen muchas estrategias para llevar a cabo la técnica de expansión de la consulta, cada una con sus características propias. Básicamente, estos mecanismos pueden ser clasificados en: *automáticos, manuales e interactivos*. Las técnicas *automáticas* tratan de añadir nuevos términos de forma automática relacionados semánticamente con aquellos que son introducidos manualmente por el usuario, para así conseguir que el sistema devuelva una colección de documentos más acorde a la idea del usuario. Esta aproximación tiene algunos inconvenientes principalmente léxicos, como por ejemplo la polisemia (distintos significados para una misma palabra). Distintas implementaciones han sido desarrolladas para tratar de identificar el significado adecuado de los términos de la consulta de forma automática. Estos algoritmos reciben el nombre de *Word Sense Disambiguation* (WSD). El método de expansión de la consulta de forma *interactiva* requiere de la colaboración del usuario. El sistema propone una serie de términos al usuario que pueden estar relacionados con su consulta para que éste elija aquellos que más se aproximen a la idea de lo que busca. Generalmente este tipo de sistemas utilizan una estructura en árbol donde ordenan de los términos desde los más generales hasta los más específicos. Este tipo de sistemas suelen ser lentos e incómodos para el usuario porque requieren de un alto nivel de participación del mismo para responder a todas las preguntas a las que le somete el sistema. Por otro lado la expansión de la consulta en muchas ocasiones se realiza utilizando estructuras de conocimiento como por ejemplo Wordnet, aunque pueden ser utilizados muchos otros tesauros y ontologías. Nuestro grupo de investigación también ha presentado nuevas propuestas en este sentido como las de Serrano-Guerrero (2008 y 2009).

Como ya se ha dicho, se han diseñado muchas arquitecturas sobre Metabuscadores (Glover., 99), (Kerschberg, 01), (Li, 01) en las cuales se pueden identificar varios componentes comunes. Un ejemplo pueden ser aquellos componentes encargados de lanzar las consultas a los diferentes motores de búsqueda o bien los componentes encargados de calcular el grado de relevancia de los documentos recuperados. La mayoría de las arquitecturas propuestas están basadas en el uso de agentes específicos, cada uno de ellos con diferentes funciones asignadas e intercomunicados a través de la red. Cada agente tiene designada una función específica, pero trabajan de forma colaborativa con otros agentes para conseguir una reducción de la

complejidad del sistema. Para ello es necesario un lenguaje para la comunicación entre agentes, conocido por todos ellos y que permitirá la colaboración entre los mismo. Este lenguaje común puede ser por ejemplo ACL (Agent Communication Language).

La lógica borrosa podría jugar un papel fundamental en esta arquitectura basada en agentes, principalmente en la tarea de unir la información procedente de diferentes fuentes (agentes) y gestionar los resultados de forma eficiente y satisfactoria.

## CAPÍTULO V

### **Aportaciones y propuestas**

En este capítulo se presentan una serie de desarrollos y aplicaciones llevados a cabo en el grupo de investigación SMILe (*Soft Management of Internet and Learning*), de la Universidad de Castilla-La Mancha, España, que el autor de este libro lidera, que plasman todos los problemas de la recuperación de información presentados anteriormente e intentan, en la medida de lo posible aportar soluciones a los mismos, en la mayoría de los casos con técnicas del ámbito de la Inteligencia Artificial y en particular de la Lógica Borrosa.

### **Representación de documentos mediante el modelo FIS-CRM**

El modelo FIS-CRM (*Fuzzy Interrelations and Synonymy based Concept Representation Model*) (Olivas, 03) es un modelo de representación lógica del contenido de cualquier tipo de documentos, y se puede considerar como una extensión “borrosa” del modelo vectorial (*Vector Space Model, VSM*) de representación de documentos. La diferencia fundamental de este modelo respecto al modelo VSM es que, mientras que en el modelo VSM los pesos de cada vector representan ocurrencias de términos, en el modelo FIS-CRM los pesos representan ocurrencias de conceptos. La primera gran utilidad que representa esta característica es la de abordar la comparación del contenido de diferentes documentos en base a los conceptos contenidos en los mismos. Y esta peculiaridad convierte a este modelo en una interesante herramienta en el campo de la Recuperación de Información.

El modelo FIS-CRM se sustenta en dos tipos de interrelaciones borrosas entre términos: la de sinonimia y la de generalidad, almacenadas en un diccionario de sinónimos y diferentes ontologías temáticas de términos respectivamente. De esta forma, se puede hablar

del grado de sinonimia existente entre dos términos (que puede ser obtenido mediante expresiones del tipo Jaccard) y del grado de generalidad entre dos términos (obtenido a partir de la co-ocurrencia de términos en colecciones temáticas de documentos).

La forma de describir el papel que juegan estos dos tipos de interrelaciones en el modelo FIS-CRM se constata en sus dos premisas fundamentales: Las apariciones u ocurrencias de una palabra en un documento se deben “repartir” entre los diferentes sinónimos de la palabra contenida, realizando el reparto en base al grado de sinonimia entre los sinónimos y la palabra contenida. Si una palabra está contenida en un documento, las palabras que representan conceptos más generales que ésta deben tener un peso en el vector de dicho documento, que será proporcional a su grado de generalidad.

La construcción de los vectores de los documentos se realiza en dos fases: En primer lugar se parte de la representación vectorial estándar de los documentos (basada en ocurrencias de términos), y posteriormente se realiza un proceso de reajuste de los pesos de los elementos de los vectores, de forma que un término pueda tomar un peso en un vector aunque no aparezca en el documento, siempre y cuando el concepto al que represente dicho término subyaga en el mismo. En realidad, el proceso de reajuste se lleva a cabo en dos fases: reajuste por generalidad y reajuste por sinonimia.

El reajuste de pesos realizado en base a la interrelación de generalidad se basa en, partiendo de la aparición de un término ‘t’ en un documento, otorgar un peso a los términos más generales a ‘t’, que sea proporcional al grado de generalidad entre ‘t’ y dichos términos.

El reajuste de pesos realizado en base a la interrelación de sinonimia se realiza mediante las siguientes expresiones:

$$w'_i = N * \frac{1}{\sqrt{\sum_{i=1}^n \mu(t_i, C)^2}} * \mu(t_i, C)$$

$$N = \sum_{i=1}^m w_i * \mu(t_i, C)$$

En estas expresiones  $w_i$  y  $w'_i$  representan el peso del término ‘i’ en el documento antes y después del reajuste respectivamente, C representa el concepto al cual convergen los diferentes sinónimos encontrados en el documento y  $\mu(t_i, C)$  representa el grado de pertenencia de cada uno de estos sinónimos al concepto en sí. A su vez, el grado de pertenencia de un término al concepto es obtenido a partir del grado de sinonimia entre ese término y el conjunto de sinónimos que conforman el concepto.

Es importante resaltar que en la última versión del modelo FIS-CRM (Garcés, 06) las entradas del diccionario de sinónimos se realizan a nivel de término y acepción, por lo que para cada pareja término-acepción se dispone de un conjunto de sinónimos con sus correspondientes grados de sinonimia. Esta característica del diccionario es la que permita identificar la correcta acepción (*word sense disambiguation*) de los términos polisémicos (*débiles*), en contraposición de los términos con una sola acepción (*fuertes*), y representa otra de las novedades que incorpora la última versión del modelo FIS-CRM. Este proceso se basa en el estudio de la co-ocurrencia de sinónimos en el documento para así identificar del conjunto de sinónimos adecuado. Cuando en el documento no co-ocurren al menos dos sinónimos pertenecientes al mismo conjunto de sinónimos es necesario recurrir al contexto local del término polisémico para, utilizando la información almacenada en las ontologías temáticas, identificar el conjunto de sinónimos más adecuado para dicho término.

### Integración del modelo FIS-CRM en sistemas de búsqueda

El modelo FIS-CRM fue inicialmente integrado en el metabuscador FISS (*Fuzzy Interrelations and Synonymy Searcher*) con el fin de representar los *snippets* de las páginas Web recuperadas por un motor de búsqueda tradicional (Google en este caso). En este sistema, el objetivo propuesto fue el de poder comparar conceptualmente los *snippets* recuperados, de manera que los resultados ofrecidos al usuario se pudiesen organizar jerárquicamente en grupos atendiendo a los conceptos contenidos en los documentos.

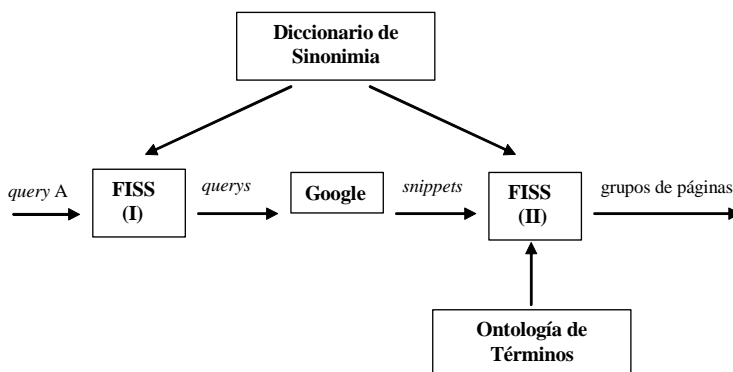


Figura 12: Proceso de búsqueda en FISS

El primer componente del metabuscador se encarga de generar nuevas consultas a partir de los sinónimos de la consulta introducida por el usuario. Las nuevas consultas tienen un grado de compatibilidad con la consulta original obtenido a partir de los correspondientes grados de sinonimia de los términos implicados. Posteriormente los *snnipets* devueltos por el motor se representan mediante el modelo FIS-CRM y finalmente, mediante un algoritmo de *soft-clustering*, se obtiene una jerarquía de grupos de enlaces a páginas web “conceptualmente” relacionadas entre sí.



Figura 13: Interfaz de búsqueda de FISS

Otra interesante aplicación del modelo FIS-CRM al campo de la recuperación de la información es mediante un sistema integral de búsqueda basado en coincidencias de conceptos. La principal diferencia de este sistema respecto del metabuscador FISS es que el modelo FIS-CRM está también integrado en el subsistema de recopilación e indexado, por lo que el conjunto de páginas Web accesibles por el motor están representadas conceptualmente mediante este modelo. Así, el motor de este sistema, utilizando un sistema estándar de *matching* entre consultas y documentos, es capaz de recuperar las páginas que contienen los conceptos solicitados por el usuario. Este proyecto se espera integrarlo de la misma forma en la plataforma BUDI (que se comenta con detalle a continuación).

### Ejemplo de búsqueda con el modelo FIS-CRM

Supongamos las tres páginas Web que se muestran en la figura 14:

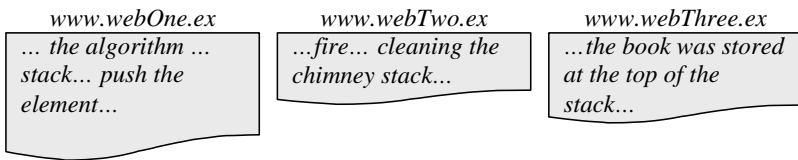


Figura 14: Ejemplo de tres documentos Web

Si se realiza una búsqueda sobre la palabra “stack”, los vectores que representan las tres Webs de la figura 13, según el modelo vectorial clásico, serán los siguientes (figura 15).

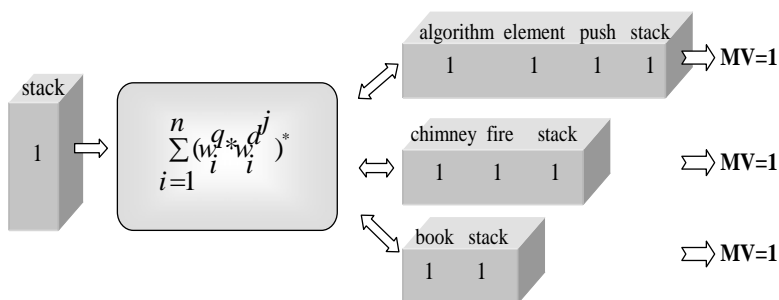


Figura 15: Ajuste de los vectores según el modelo vectorial clásico

Pero supóngase que al indexar estas tres páginas se usa el modelo FIS-CRM en lugar del modelo vectorial clásico. El conjunto de sinónimos de la palabra *stack* podrían ser estos tres:

- *stack1* → (stack): Asumiendo que no hay otro sinónimo para el sentido relacionado con las estructuras de datos.
- *stack2* → (stack, flue): Para el sentido relacionado con chimeneas (*chimneys*).
- *Stack3* → (stack, shelves): Para el sentido relacionado con el almacenamiento de libros.

Supongamos que el usuario desambigua el significado de *stack* cuando lanza la consulta, eligiendo una ontología que tiene que ver con la programación. En este caso, en los vectores el término “data structure” consigue un valor debido a su relación borrosa de generalidad con el término *stack*. Pero el término *stack* no puede compartir sus ocurrencias porque el conjunto de sinónimos para esa acepción sólo incluye ese término. Algo similar sucede con el término “data structure” en el primer vector, pero en este caso, la desambiguación de *stack* fue hecha en el momento de la indexación,

debido al hecho de que la ontología de software es la más adecuada para este documento. En el segundo vector, la palabra *chimney* ha compartido su peso con su sinónimo *fireplace* y el término *stack* se desambigua con su segundo sentido (*stack2*). En este caso, las ocurrencias de *stack* se dan a su sinónimo fuerte *flue*. En el tercer vector, el término *stack* se desambigua a la tercera acepción (*stack3*). Suponiendo que *shelves* es también un término débil, las ocurrencias se comparten entre los dos términos del conjunto. Por lo tanto, los valores de estos vectores con el modelo FIS-CRM serían (figura 16):

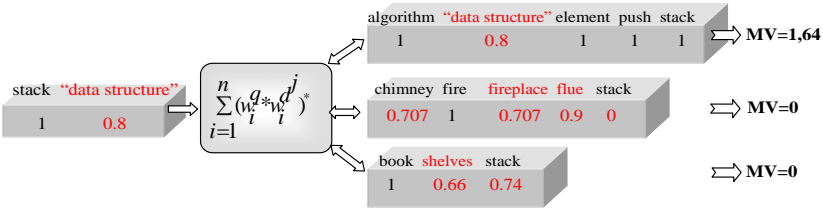


Figura 16: Ajuste de los vectores según el modelo FIS-CRM

## GUMSe: Metabuscador basado en agentes

GUMSe (GUM Search) (De la Mata, 04) es un Metabuscador experimental que se presenta como una alternativa prometedora para tratar de paliar la baja precisión de los buscadores actuales. Como se ha reflejado durante este trabajo, las líneas de investigación actuales para mejorar la relevancia de los documentos devueltos por el proceso de búsqueda se basan en la incorporación de capacidades semánticas y deductivas mediante el uso de diccionarios, conceptos tales como la sinonimia y la antonimia, el uso de ontologías o la teoría de las percepciones. Uno de los principales problemas, a la hora de buscar, es la selección de los términos que formarán parte de la consulta. Existen distintos términos que se pueden elegir a la hora de referirse al mismo concepto, es un caso muy común que el autor de un documento utilice términos distintos a los que el usuario utiliza para tratar de recuperarlo. Este es el conocido, y ya comentado en el capítulo III, problema del vocabulario. Hasta ahora era el usuario el encargado de atenuar este problema mediante el sucesivo refinamiento de la consulta. Este proceso conlleva una serie de iteraciones en las que el usuario debe construir nuevas consultas para obtener los resultados esperados. Para intentar reducir este proceso, y por tanto obtener unos resultados más satisfactorios, GUMSe combina varias técnicas para



reducir el número de iteraciones, con la finalidad última de obtener un conjunto de documentos más relevantes. Las principales técnicas que se utilizan en GUMSe son fundamentalmente tres: la expansión de consulta, la desambiguación del significado y una arquitectura basada en agentes. Dentro de la expansión de consulta se utiliza, dependiendo de la situación, la expansión automática o interactiva.

## GUMSE MetaSearch

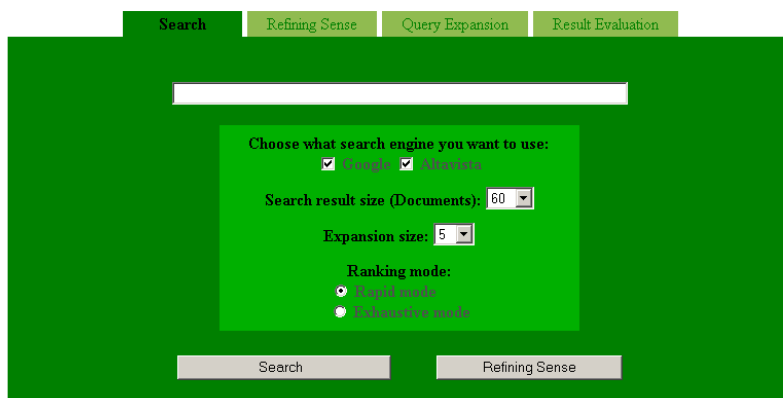


Figura 17: Interfaz de búsqueda de GUMSe

La *expansión de consulta automática* selecciona los términos que formarán parte de la consulta de forma transparente para el usuario. Pero antes de aplicar la expansión de consulta, es necesario conocer el significado adecuado al que el usuario se refiere cuando utiliza términos polisémicos, por lo que es preciso realizar una *desambiguación del significado*. En un principio, se utiliza WordNet como fuente de los términos que formarán parte de la expansión. Como ya se ha comentado reiteradamente a lo largo de este trabajo, WordNet es una herramienta muy utilizada para las aplicaciones del procesamiento del lenguaje natural. Una de sus principales aplicaciones ha sido la recuperación y extracción de información, principalmente para la desambiguación automática de los significados de las palabras.

En la *expansión de consulta interactiva* el objetivo es solicitar del usuario el significado más cercano a sus intenciones de búsqueda en aquellos casos en los que no se halla podido obtener de forma automática (o si el usuario decide utilizarlo). La información obtenida se puede utilizar para expandir la consulta añadiendo términos. Estos términos están relacionados mediante relaciones de sinonimia e hiperonimia con los introducidos por el usuario inicialmente. Además, este mecanismo interactivo permite al sistema aprender las relaciones entre términos y sus

significados o acepciones. Esta característica aporta una valiosa información que se puede utilizar posteriormente para mejorar el sistema. Por ejemplo, es posible aprender nuevos significados, reconocer acrónimos o detectar significados que se pueden agrupar en uno solo. Además, permite mejorar la desambiguación automática basándose en la acumulación de la experiencia de los usuarios, ya que el sistema puede aprender las relaciones existentes entre los términos de una consulta y sus significados. Supongamos que un usuario pregunta al sistema por “house clinton”. El sistema podría preguntarle usando Wordnet a que acepciones de estas palabras se refiere, como se muestra en la figura 18.

## GUMSE MetaSearch



Figura 18: Desambiguación interactiva para la consulta “house clinton”

Habitualmente, las técnicas de expansión de consulta aumentan el factor *recall* sin que lleve asociado un aumento de la precisión. Debido a este inconveniente, no basta con un mecanismo de refinamiento de las consultas para obtener un conjunto de documentos más relevantes para el usuario. Se requiere un posterior procesamiento

de los distintos resultados obtenidos que implica la integración y reajuste de la relevancia a la búsqueda del usuario. Por este motivo, se incorporan *mecanismos de evaluación* que permiten aumentar la precisión de la búsqueda, reduciendo la importancia de los documentos alejados conceptualmente. Una de las formas de abordar esta cuestión consiste en utilizar la información obtenida en la desambiguación del significado. Los términos relacionados con las acepciones no seleccionadas se utilizarán como indicadores negativos a la hora de determinar el grado de relevancia de un documento. De esta forma, conseguimos aprovechar las acepciones no seleccionadas para determinar la afinidad conceptual de los documentos con la consulta del usuario.

Por último, un aspecto determinante en el diseño de GUMSe es su *arquitectura basada en agentes* mostrada en la figura 19. GUMSe está formado por una serie de agentes con distintas funciones que se comunican entre sí a través de una red. Se ha elegido este planteamiento porque permite la utilización de recursos distribuidos. Esta posibilidad aumenta la eficiencia y rendimiento al distribuir la carga de procesamiento entre diferentes computadoras y permitiendo la ejecución de tareas en paralelo.

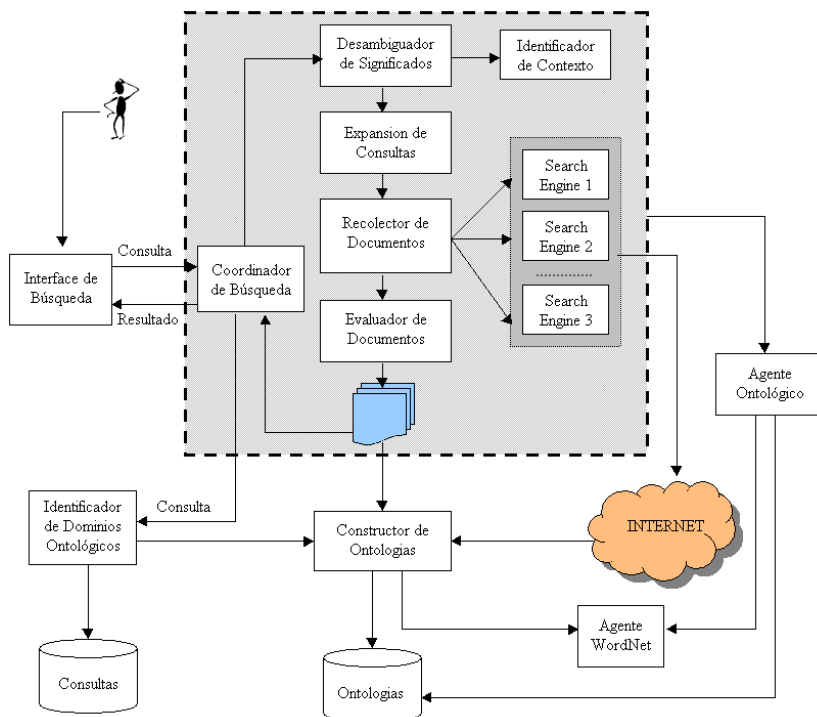


Figura 19: Arquitectura de agentes de GUMSe

Se pueden distinguir dos tipos de agentes dependiendo del tipo de participación en el proceso de recuperación. Si la participación es directa, se denominan “agentes de búsqueda” (con fondo gris en la figura), y son los que se encargan de satisfacer la consulta de los usuarios. El segundo tipo de agentes, denominados “agentes de soporte a la búsqueda”, agrupa a los agentes que aportan una funcionalidad destinada a facilitar el acceso, construcción, gestión, o mejora de las estructuras de conocimiento que GUMSe utiliza para mejorar la búsqueda. La finalidad de estos últimos es identificar aquellos conceptos más demandados por los usuarios con el fin de construir ontologías que capten la semántica que subyace en torno a ellos. La información que poseen estas ontologías se utiliza posteriormente para refinar la consulta introducida por el usuario. El primer paso es identificar los dominios conceptuales más requeridos por los usuarios para mejorar o construir ontologías que traten de captar las relaciones semánticas y jerárquicas de los conceptos pertenecientes a ese dominio. Posteriormente se construyen ontologías utilizando los documentos relacionados con un concepto para obtener un conjunto de términos representativos. Evidentemente, el proceso de construcción de estas estructuras requiere el procesamiento de elevadas cantidades de documentos, lo que implica un proceso de recuperación, indexación y posterior tratamiento. Esta labor la realiza el Agente Constructor de Ontologías ayudado por el Agente Identificador de Dominios Ontológicos. Su labor es identificar los conceptos más buscados por los usuarios. Una vez conocido se procede a construir las ontologías que les den soporte. En este proceso se utilizan las consultas de los usuarios y los mejores documentos obtenidos por dichas consultas.

## **Cálculo de distancias borrosas entre los términos de una ontología**

Una ontología se define habitualmente como la conceptualización consensuada, formal y explícita de una realidad. La representación de una ontología puede ser entendida como un grafo donde los nodos representan los conceptos o términos de la ontología y los arcos el tipo de relación existente entre los nodos que unen. Existen un gran número de relaciones entre términos: holonimia, hiperonimia, meronimia, etc., que pueden ser obtenidas de diversas maneras: procesamiento de lenguaje natural, diccionarios electrónicos, etc., pero generalmente dotan de una gran complejidad a la ontología para poder ser usada en procesos automáticos. Por esta razón se realiza una simplificación del tipo de relaciones existentes entre nodos, distinguiéndose dos grandes tipos: relaciones físicas y relaciones semánticas (Serrano-Guerrero, 08).

Las relaciones físicas engloban todas las relaciones anteriormente citadas: holonimia, meronimia, hiponimia, etcétera, relaciones *fuertes*, *evidentes* e *intuitivas* entre términos. Las relaciones semánticas se basan en la co-ocurrencia de términos y el concepto de distancia física. Así, la aparición reiterada de términos aislados o la co-ocurrencia de términos en el mismo documento fortalecen la idea de que el tema principal del texto versa entorno a ellos. Mientras, la distancia física esta basada en la idea de que los términos que más cercanos se encuentran en un documento tienen mayor probabilidad de estar fuertemente relacionados que aquellos términos que aun estando en el mismo documento se encuentran en párrafos distintos. Basándose en estas dos únicas relaciones se aplican distintos operadores borrosos para obtener distintas ontologías generadas de manera completamente automática.

El proceso de búsqueda se puede descomponer en una serie de fases. El objetivo consiste en refinar sucesivamente la carga semántica de la consulta para permitir expandir los resultados de la misma con documentos relacionados conceptualmente. Las 4 fases son las siguientes:

1. *Refinamiento del significado de la búsqueda*: Se identifican los significados adecuados de los términos polisémicos que intervienen en la consulta.
2. *Expansión de la consulta*: Se utiliza la información adquirida de la fase anterior para aportar una mayor carga semántica al conjunto de consultas. En esta etapa se construyen una serie de consultas adicionales a la original. En GUMSe, las nuevas consultas se pueden construir utilizando cuatro procedimientos distintos. El primero consiste en introducir términos que posean relaciones de sinonimia e hiponimia con los términos de la consulta del usuario. También se utilizan variaciones morfológicas de los términos de la consulta. El aspecto más novedoso consiste en la utilización de operadores avanzados de búsqueda en la consulta. Cada motor de búsqueda existente tiene sus propias características y operadores de búsqueda concretos. Este motivo ha propiciado el diseño de un lenguaje propio basado en XML utilizado en el proceso de expansión de consulta, que posteriormente se traduce al lenguaje de consulta específico de cada buscador. De esta forma se aísla la construcción de las nuevas consultas expandidas de un buscador concreto, para permitir en futuras mejoras del sistema incorporar otros buscadores a GUMSe. El último método de expansión consiste en introducir conceptos relacionados semánticamente. Junto a los métodos anteriores se utiliza un refinamiento negativo de la consulta. El objetivo es descartar documentos que posean términos relacionados con significados descartados, consiguiéndose de esta forma enfocar la búsqueda.

Supongamos que el usuario introduce las consultas “database” (figura 20) o “salt” (figura 21) pero esta última con la intención de buscar por la acepción de *sal común* y nada que tenga que ver con las siglas de *Strategic Arms Limitation Talks* (SALT), GUMSe traduce estas consultas incluyendo operadores de búsqueda propios de los buscadores a los que irán destinadas, tal y como se muestra en la figuras 20 y 21.

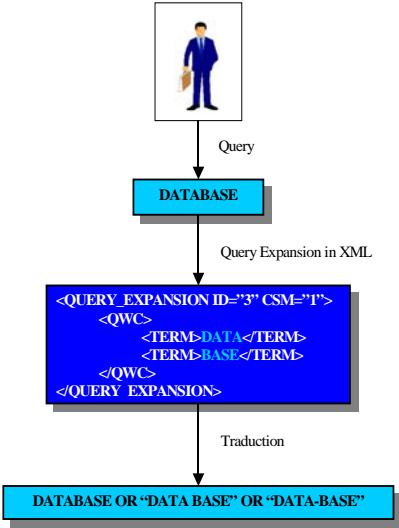


Figura 20: Expansión para la consulta “database”

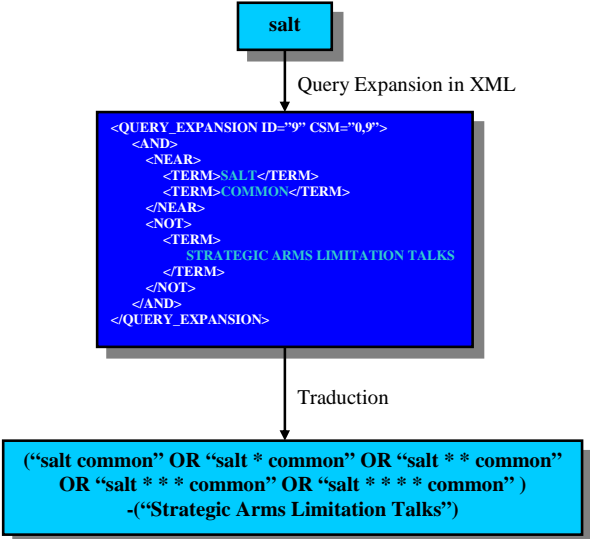


Figura 21: Expansión para la consulta “salt”

3. *Recolección de la web*: El siguiente paso consiste en lanzar las distintas consultas a una serie de motores de búsqueda para obtener una colección finita de páginas. Actualmente, sólo se considera la utilización de Google y Altavista.
4. *Evaluación de los resultados*: Por último se procede a recuperar las páginas obtenidas y mostrarlas al usuario ordenadas bajo un único criterio de relevancia. Se deben obtener los resultados de las consultas realizadas a los diferentes buscadores y agruparlas en una única colección de documentos, eliminando aquellos documentos repetidos y estableciendo un orden de aparición. Por motivos de eficiencia, se han considerado dos sistemas diferentes de evaluación de los documentos denominados evaluación rápida y evaluación exhaustiva. La primera es más eficiente y rápida que la segunda ya que no entra a examinar el contenido de los documentos que evalúa, sin embargo, la segunda modalidad de evaluación ofrece unos cálculos de relevancia más fiables.

## **FzMail: Una herramienta para la gestión inteligente del correo electrónico**

FzMail (Romero, 07) es una herramienta que se pretende desarrollar completamente e integrar en la plataforma BUDI, que utiliza diversas técnicas de *Soft-computing* con el propósito de gestionar de forma eficiente grandes volúmenes de correo electrónico. Los objetivos principales del mecanismo son la obtención de una organización jerárquica y borrosa basada en los conceptos tratados en los mensajes, la clasificación automática de los correos entrantes de forma eficaz y facilitar las diferentes búsquedas que se realicen sobre el buzón de correo electrónico.

Para lograrlo se basa en tres pilares fundamentales: la representación conceptual de los mensajes, el *clustering* jerárquico-borroso y la representación del conocimiento obtenido mediante Prototipos Deformables Borrosos. La organización a construir debe de tener la mayor calidad posible para que permita un aprovechamiento posterior óptimo por parte del usuario. Las características de los resultados son las siguientes:

1. Organización jerárquica de los mensajes en carpetas siguiendo criterios basados tanto en el contenido “conceptual” del mensaje como en los campos estructurados de éste. La representación conceptual de los mensajes permitirá la posterior búsqueda basada en conceptos.

2. Definición de las carpetas mediante términos/conceptos relevantes y Prototipos Deformables Borrosos.
3. Permitir que un mensaje pueda almacenarse según su contenido en más de un grupo de entre los que forman la jerarquía.
4. Cada mensaje debe de tener un grado de afinidad (o pertenencia) a cada grupo en el que esté clasificado. Los mensajes se ordenarán según este grado de afinidad.
5. Sencillo mecanismo de comparación entre documentos y la definición de cada una de las carpetas. Los resultados de este “*matching*” se utilizarán para almacenar el mensaje en una o varias carpetas.

Las fases de construcción de esta organización son las siguientes:

1. Preproceso Lingüístico: Utilización de técnicas clásicas de Procesamiento del Lenguaje Natural (*stop words*, *stemming*, *ranking* de palabras) con el fin de tratar los términos existentes.
2. Representación Conceptual: Utilización de FIS-CRM para abstraer los conceptos inherentes en los mensajes basándose en las relaciones de sinonimia y ontologías borrosas existentes entre los términos.
3. *Clustering* Jerárquico-Borroso: Utilización de algoritmos de *clustering* para agrupar de forma jerárquica los mensajes conceptualmente similares, con la posibilidad de pertenecer un mensaje a varios grupos.
4. Post-Proceso y Representación del Conocimiento: Transformación de las agrupaciones del *clustering* para dar lugar a una estructura de carpetas y la representación de los grupos de documentos mediante Prototipos Deformables Borrosos (Olivas, 00).

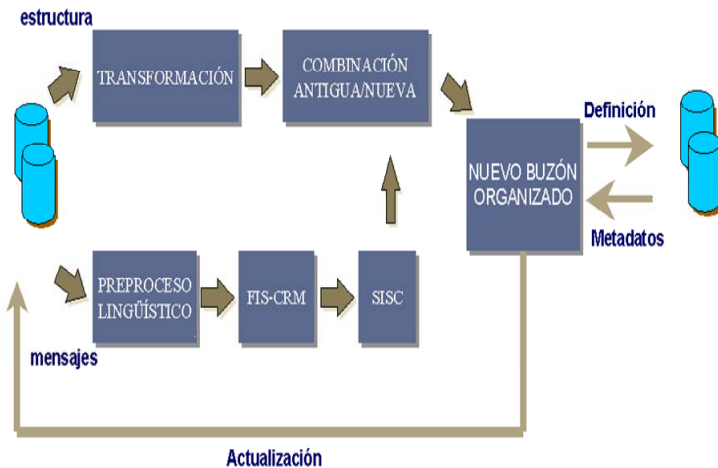


Figura 22: Tratamiento de los mensajes en FzMail



## BUDI: Plataforma de Metabúsqueda para la búsqueda difusa de información

El objetivo principal de BUDI (Serrano-Guerrero, 2009) es dotar a cualquier aplicación de una herramienta que permita realizar búsquedas *conceptuales* y que pueda trabajar no sólo sobre buscadores Web sino también sobre cualquier tipo de base de datos documental o motor de indexación: Terrier (Ounis, 06), SMART (Buckley, 95), etcétera. BUDI es un componente software reutilizable en aplicaciones o sistemas de información “verticales” (referentes al mercado, productos y servicios) en diferentes entornos: jurídicos, sanitarios, parlamentarios, etc. De esta forma BUDI se integra con sistemas empresariales que almacenan información de negocio interactuando, por ejemplo, con sistemas de gestión documental, bases de datos, sistemas Data Warehouse, etc., permitiendo la localización rápida y precisa de información.

BUDI está estructurado en dos componentes principales:

- *El núcleo operativo del sistema.* Está formado por componentes capaces de indexar, realizar búsquedas, ordenar o clasificar respuestas y mostrar resultados. Todas estas tareas las realiza mediante técnicas de lógica borrosa, tratamiento semántico -basado en el significado- y procesamiento de lenguaje natural (subdisciplina de la Inteligencia Artificial).
- *Base de Conocimiento Semántica.* Para realizar las tareas de procesamiento semántico de consultas y resultados se necesita contar con un sistema constituido por diferentes conjuntos de términos, sinonimias, ontologías, etc. que identifiquen distintas disciplinas del saber (información jurídica, sanitaria, etc.). Este sistema o base semántica tiene una gestión única e independiente.

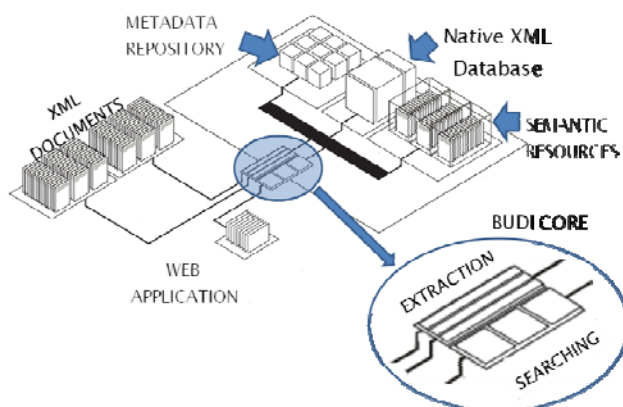


Figura 23: Arquitectura general de BUDI

La funcionalidad del sistema se organiza en módulos diferentes, agrupados por los tipos de tareas que deben realizar. De esta manera, se cuenta con una serie de módulos operativos que se ocuparán de efectuar las tareas clave del sistema como la transformación de las consultas y la evaluación de resultados. Estos módulos interactúan con los elementos clave como el motor de búsqueda y la aplicación cliente que presenta los resultados al usuario. También existen una serie de módulos de gestión o soporte que se ocuparán de las tareas internas propias del sistema como la manipulación de los recursos semánticos y la configuración del sistema.

La filosofía de funcionamiento de BUDI se basa en aplicar tecnologías de *Soft-Computing* para diseñar soluciones de búsqueda basadas en:

- Extracción masiva de información (Data-Mining).
- *Clustering* de los documentos recuperados basados en grupos temáticos, que pueden estar o no predefinidos.
- *Reglas de asociación borrosa* que ayudan al usuario a encontrar nuevos términos utilizables en su consulta.
- Representación de los documentos usando extensiones del modelo espacio vectorial estándar.
- *Metabuscadores*: Fusión de resultados ofrecidos por otros buscadores y/o motores de búsqueda específicos.

Una de las principales funcionalidades de BUDI es el *módulo de procesamiento de consultas*, que se ocupa de transformar la consulta recibida con el objetivo de conseguir una mayor calidad de los resultados explotando toda la información semántica que tiene disponible. Una de las principales tareas que realiza es generar nuevas consultas a partir de las palabras relacionadas semánticamente con las existentes en la consulta introducida por el usuario. Las nuevas consultas tienen un grado de compatibilidad con la consulta original obtenido a partir de los correspondientes grados de relación de los términos implicados. Previamente para cada término y sus posibles conceptos se realiza un proceso de desambiguación de los mismos. Esto intenta reducir al mínimo el número de búsquedas y resultados no útiles y por consiguiente mejorar la calidad de los resultados. Las consultas a realizar se expresarán en un lenguaje genérico basado en XML. Al igual que en GUMSe, dado que cada motor de búsqueda a utilizar tiene sus propias características y operadores de búsqueda concretos será necesario transformar estas consultas genéricas en peticiones específicas a cada motor. También la conexión a cada uno de estos motores será específica y diseñada a partir de conectores los cuales servirán para ejecutar la consulta y devolver los resultados. Dentro de esta fase de recepción se incluye en análisis de los operadores que hay en la consulta, los cuales vendrán definidos por la

aplicación cliente. En la figura 24 se muestra un esquema detallado de cómo se reciben las consultas.

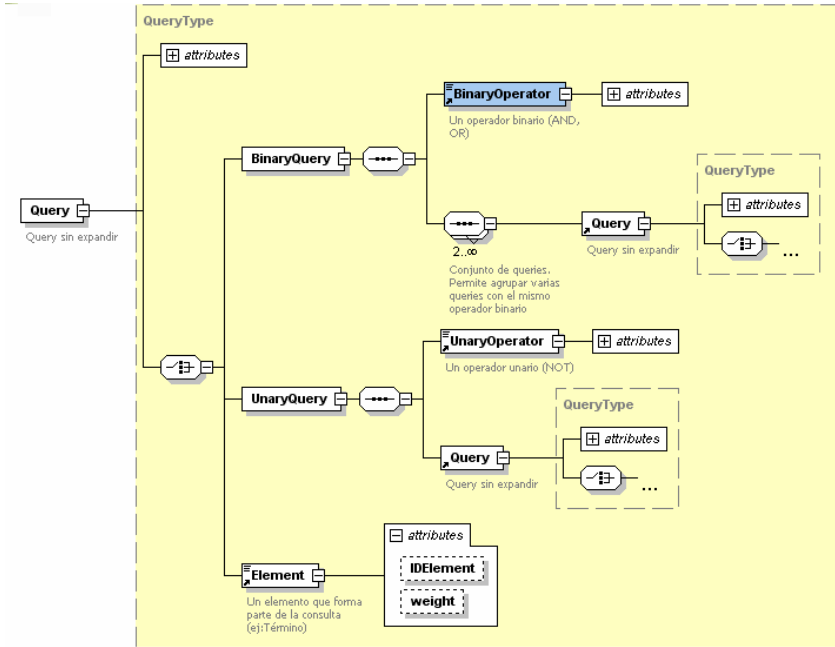


Figura 24: Esquema detallado de recepción de consultas en BUDI

La utilización de BUDI es muy sencilla. A continuación se refleja el código fuente que tendría que incorporarse en las aplicaciones para realizar búsquedas a través de BUDI.

1. Creamos el procesador de consultas:

```
QueryProcessor queryProcessor = new QueryProcessor();
```

2. Inicializamos el procesador de consultas con la configuración deseada: (motores de búsqueda a utilizar, ranking de resultados, etc.):

```
QueryProcessor.init("config.xml");
```

3. Procesamos la consulta que hemos obtenido de la interfaz de usuario para determinar operadores y motores de búsqueda:

```
query_aux = processQuery(query);
```

4. Ejecutamos la consulta:

```
result = queryProcessor.executeXMLQuery(query_aux);
```

5. Obtenemos los resultados en el formato adecuado:

```
documents = parseoDocument(result);
```

6. A partir de aquí deberíamos ofrecer los resultados al usuario en la interfaz de nuestra aplicación concreta (ejemplo en figura 25).



Búsqueda Difusa de la Información

[Guías prácticas](#)  
Guías prácticas

[Calidad ISO 9001 2000](#)  
Calidad ISO 9001 2000

[Objetivos de la metodología de proyectos](#)  
Objetivos de la metodología de proyectos

[Metodología de Proyectos - Construcción](#)  
Metodología de Proyectos - Construcción

Figura 25: Ejemplo de portal de búsqueda con BUDI

## FOG: Generación automática de Ontologías Borrosas

La finalidad de FOG (*Fuzzy Ontology Generator*) es crear automáticamente ontologías que puedan ser utilizadas por un sistema de recuperación de información. La idea de este proyecto surge como elemento complementario para BUDI, desarrollado en colaboración entre la empresa CSD (*Company for Software and Development*, Valencia, España), con el fin de cubrir algunas de sus deficiencias como es la gestión de información externa útil para los distintos procesos que requieren de cierta información con carga semántica. FOG se puede definir como una herramienta abierta a la incorporación de nuevos algoritmos con los que se puedan diseñar distintas estrategias para la creación de ontologías de forma automática. El objetivo es poder realizar variaciones en las funcionalidades e incluso intercambiarlas para poder obtener distintos resultados representados visualmente mediante un grafo o en un lenguaje para la representación de ontologías (como por ejemplo RDF, OWL, Fuzzy OWL...). Así queda patente que es una herramienta en continuo desarrollo al ser posible realizar mejoras de rendimiento, aumentar la funcionalidad de componentes e incorporar nuevas estrategias. FOG debe ser totalmente independiente de cualquier librería externa que pueda ser susceptible de ser utilizada (indexadores, procesadores léxicos o sintácticos, visualizadores...), de tal forma que cualquier componente pueda ser intercambiado sin problemas. Antes de comentar los principales módulos que forman FOG, es necesario comentar un

componente ajeno al flujo de datos. Este componente es el módulo de configuración (CONFIG), en el cual el usuario de la herramienta establece los parámetros que determinan la estrategia a seguir para la creación de la ontología. El módulo de configuración lee los datos de un archivo XML, de forma que mediante reflexión se procede a la carga de los mismos definiendo así el flujo de ejecución del sistema. A lo largo de un proceso intervienen distintos módulos que pueden actuar en tres fases principales: Extracción de datos, Modelado de información y Representación de la ontología. En la figura 26 se presenta la arquitectura general de FOG.

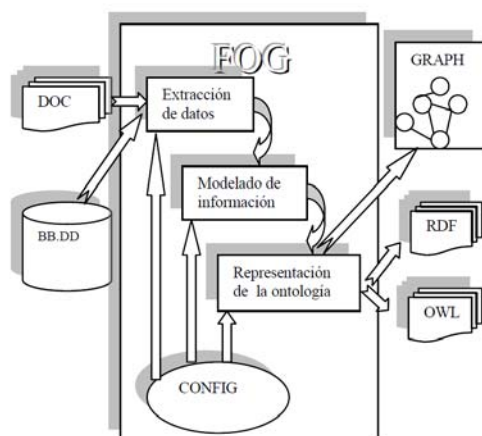


Figura 26: Arquitectura general de FOG

Tras la conclusión de los procesos reflejados en esta arquitectura, se obtiene como resultado una ontología. Pero su utilización en otras aplicaciones depende de la representación final. Para sistemas de recuperación de información como BUDI, la representación de un grafo conceptual de un dominio facilitaría la desambiguación de términos especificados en una consulta y así poder precisar más la respuesta del sistema a una consulta de un usuario basándose en el contexto.

FOG permite la visualización de la ontología en distintos formatos y cada uno pone de manifiesto una serie de características a destacar. Como ejemplo de visualización se presentan las siguientes figuras. Estas muestran las relaciones entre las entidades detectadas en los siguientes documentos que son oraciones de la colección MEDLARS<sup>24</sup>:

<sup>24</sup><[ftp.cs.cornell.edu/pub/smart/](ftp://ftp.cs.cornell.edu/pub/smart/)>.

- Doc1: *...correlation between maternal and fetal plasma levels of glucose and free fatty acids.*
- Doc2: *...free fatty acid concentration in maternal plasma and fetal body fat content.*
- Doc3: *...the concentration of non-etherified fatty acids in maternal and fetal plasma in intact, alloxan-diabetic and x-ray-irradiated rats.*
- Doc4: *...7 patients who had biopsy of the bone marrow, all had abnormal types or numbers of plasma cells.*

En la figura 27, se puede apreciar la expansión de conceptos. El usuario puede seleccionar el nivel de abstracción que desea visualizar. Se puede ver la relación existente entre la entidad *acids* y la entidad *fatty*, y por consiguiente, existe un grado de relación semántica establecido por los mecanismos aplicados en la etapa de detección de relaciones.

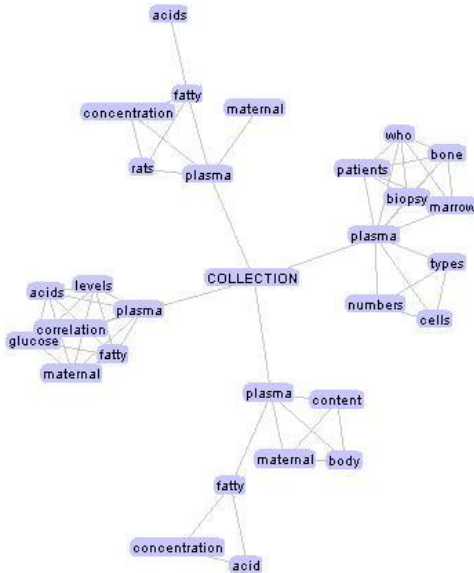


Figura 27: Representación visual de la ontología generada por FOG en función del nivel jerárquico

En la figura 28 se representa la proximidad entre entidades en función de una entidad elegida como entidad central. En el caso del ejemplo, la entidad central es *collection*. Para que un sistema de recuperación de información pueda hacer uso de este grafo conceptual, debe estar

representado en un lenguaje que el sistema de recuperación de información pueda manejar. La representación final del grafo conceptual podría ser en un lenguaje como RDF, OWL ó FUZZYWOWL. Se hará uso de una representación u otra en función de lo especificado por el usuario y de las restricciones que se incluyan dentro de la ontología.

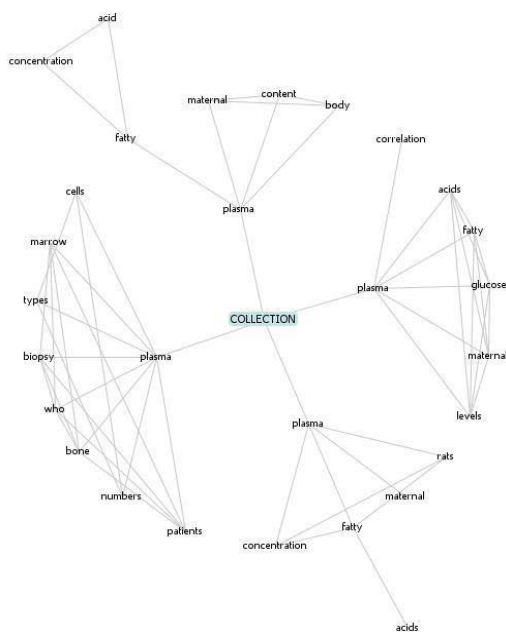


Figura 28: Representación visual de la ontología generada por FOG en función de la entidad central *collection*

## Estudio y uso de las relaciones causales

En los trabajos llevados a cabo principalmente por Cristina Puente y Alejandro Sobrino (Puente, 10) se estudian y usan las relaciones causales en el campo de la recuperación de información. La idea de que el conocimiento causal es una característica fundamental para el entendimiento del mundo es muy antigua. Ya Aristóteles, en la obra *Metaphysics* sostiene que el conocimiento está fundamentado en las causas. En la historia de la causalidad ha habido dos concepciones que han tenido gran influencia sobre el resto: la causalidad como una ilusión, defendida por Hume y la causalidad como base epistemológica para la ciencia, defendida por Kant o por Mill. Dentro

del ámbito científico, en las ciencias empíricas como la física, las relaciones causales son una manera típica de generar conocimiento y aportar explicaciones. En las ciencias sociales, como en el derecho, la causalidad se utiliza para establecer una conexión entre conducta y responsabilidad, por ejemplo, como justificación en la imposición de una pena. Así, un jurado puede determinar si la negligencia de un médico en el tratamiento de la víctima fue la causa inmediata de su muerte y, en función de eso, condenarle o no.

Se ha diseñado un proceso para detectar y clasificar oraciones condicionales en documentos de texto. Estas sentencias se extraen de acuerdo a 20 patrones predefinidos (figura 29) que abarcan los casos más comunes de condicionales en el idioma inglés. Para ello se utilizó el analizador morfológico de libre distribución Flex junto con el lenguaje de programación C, dando lugar a un proceso capaz de seleccionar oraciones de un texto que encajen dentro de esos patrones.

- Structure 1: if + present simple + future simple.
- Structure 2 : if + present simple + may/might.
- Structure 3 : if + present simple + must/should.
- Structure 4 : if + past simple + would + infinitive.
- Structure 5 : if + past simple + might/could.
- Structure 6 : if + past continuous + would + infinitive.
- Structure 7 : if + past perfect + would + infinitive.
- Structure 8 : if + past perfect + would have+ past participle.
- Structure 9 : if + past perfect + might/could have + past participle.
- Structure 10 : if + past perfect + perfect conditional continuous.
- Structure 11 : if + past perfect continuous + perfect conditional
- Structure 12: if + past perfect + would + be + gerund
- Structure 13: for this reason, as a result.
- Structure 14: due to, owing to.
- Structure 15: provided that.
- Structure 16: have something to do, a lot to do.
- Structure 17: so that, in order that.
- Structure 18: although, even though.
- Structure 19: in the case that, in order that.
- Structure 20: on condition that, supposing that.

Figura 29: Patrones para la detección de frases condicionales en inglés

En un principio se analizaron las formas en inglés pertenecientes al condicional en su modo más típico, “if x then y”, así como otras fórmulas equivalentes que pudieran dar lugar a este tipo de oraciones



(*due to, caused by, provided that, have something to do with*). Para comprobar el nivel de fiabilidad de la aplicación se analizaron manualmente una serie de documentos pertenecientes a diferentes ámbitos (50 páginas por categoría), y se calcularon las medidas de *recall* y precisión, como se puede apreciar en la tabla 1. La relevancia R es el número de oraciones causales correctamente clasificadas por el sistema dividido por el número de oraciones causales clasificadas manualmente. La precisión P es el número de oraciones causales correctamente clasificadas por el proceso dividido por el total de sentencias devueltas. La medida F (*F-Measure*) como se ha comentado anteriormente, determina la relación entre la relevancia y precisión:

<b>Tipo de texto</b>	<b>Relevancia</b>	<b>Precisión</b>	<b>F-Measure</b>
Científicos	0,65	0,84	0,73
Médicos (Medlars) <sup>25</sup>	0,77	0,91	0,83
Novelas	0,32	0,79	0,41
Noticias (Reuters) <sup>26</sup>	0,58	0,79	0,67
Evangelios	0,50	0,70	0,58

Estos resultados indican que los textos científicos y médicos tienen unos mejores resultados en relevancia y precisión que el resto de dominios tratados, por lo que los experimentos a partir de este punto se efectuaron dentro del ámbito médico.

Por tanto, el siguiente paso consistirá en la localización de todas aquellas oraciones condicionales, previamente seleccionadas, en las que se mencione el concepto solicitado. Se ha creado otra aplicación mediante el analizador morfológico Flex<sup>27</sup> que se encarga de generar un fichero de texto plano con las frases seleccionadas. Cada una de estas oraciones pasará directamente a una aplicación programada en C, donde se analizará si el concepto introducido en el proceso está dentro del contexto condicional de la oración. En caso afirmativo se presentará un esquema con las conjunciones, modificadores y demás términos relevantes para su posterior representación en el grafo causal. De una manera general, el esquema completo del proceso es el representado en la figura 30, donde se parte de un conjunto de documentos para, a través del proceso descrito, generar un grafo causal con la información relacionada con el concepto introducido por el usuario. El proceso desarrollado es capaz de simplificar una

<sup>25</sup> <[www.nlm.nih.gov/bsd/mmshome.html](http://www.nlm.nih.gov/bsd/mmshome.html)>.

<sup>26</sup> <[www.daviddlewis.com/resources/testcollections/reuters21578/](http://www.daviddlewis.com/resources/testcollections/reuters21578/)>.

<sup>27</sup> <[www.gnu.org](http://www.gnu.org)>.

sentencia compleja, extrayendo los conceptos y modificadores que pudieran estar relacionados con el buscado.

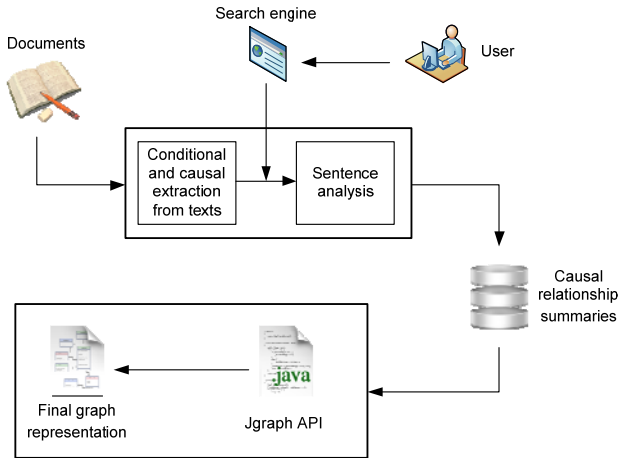


Figura 30: Proceso completo de representación de grafos causales

## Ejemplo

La relación entre dos conceptos como *smoking* y *lung cancer*. El conjunto de documentos seleccionados para este experimento fueron tomados de la página Web de la clínica Mayo<sup>28</sup>, la sociedad americana de oncología<sup>29</sup>, el centro de control y prevención de enfermedades<sup>30</sup> y los portales de *emedicina salud*<sup>31</sup> y *cáncer de pulmón online*<sup>32</sup>. La primera parte en este experimento se encarga de extraer y clasificar las oraciones condicionales y causales de acuerdo a los 20 patrones definidos. Los resultados fueron bastante satisfactorios, en parte por el tipo de lenguaje utilizado en los textos médicos, mucho más claro y preciso que en otras áreas. Después se filtraron aquellas frases que contienen las palabras *lung cancer* o *smoking*. El algoritmo devolvió 82 frases de las que se seleccionaron 15 para representar un grafo de tamaño mediano, y por tanto más manejable. Una vez que las frases han sido seleccionadas, el programa tiene que dividir las en *tokens*. Finalizado el análisis, el proceso de esquematización localizará aquellas etiquetas relativas a partículas causales, por ejemplo *due\_to*.

<sup>28</sup> <[www.mayoclinic.com](http://www.mayoclinic.com)>.

<sup>29</sup> <[www.cancer.net/patient/Cancer+Types/Lung+Cancer/](http://www.cancer.net/patient/Cancer+Types/Lung+Cancer/)>.

<sup>30</sup> <[www.cdc.gov/cancer/lung/basic\\_info/](http://www.cdc.gov/cancer/lung/basic_info/)>.

<sup>31</sup> <[www.cdc.gov/cancer/lung/basic\\_info/](http://www.cdc.gov/cancer/lung/basic_info/)>.

<sup>32</sup> <[www.lungcanceronline.org/info/index.html](http://www.lungcanceronline.org/info/index.html)>.

Esta etiqueta asocia las palabras como nodos causa y efecto. Para crear los dos nodos, el algoritmo busca dentro de cada etiqueta utilizando los números asociados a cada palabra. Por ejemplo, para crear el nodo causante, el algoritmo buscará entre las etiquetas la palabra *smoke*. Si la etiqueta es un modificador, el programa incluirá la palabra asociada en el nombre del nodo. El algoritmo devuelve un esquema de la información útil de la oración que se almacena dentro de una base de datos. Una vez que todas las frases se han terminado de procesar y esquematizar, se representará automáticamente a través de la API de Java *Jgraph* el grafo causal con los datos almacenados. A la hora de representar el grafo, algunas de estas oraciones comparten el mismo nodo antecedente o consecuente, como por ejemplo el nodo *risk lung cancer*, que aparece varias veces. Este nodo hereda todos los modificadores que aparecen en todas las frases. Una relación está especificada por un modificador si la flecha entra dentro del nodo apuntando al modificador implicado. En caso contrario, la flecha de la relación señalará al borde del nodo. Tanto causas como efectos pueden ser descritos de forma general (*pulmonary hypoplasia*) o más específicamente: los médicos generalmente hablan de *pulmonary hypoplasia* localizada en *right lung* o *right upper lobe pulmonary hypoplasia*. Por esto, en los textos médicos es frecuente encontrar localizaciones (*right*) o especificaciones (*upper lobe*) en frases nominales. Si la relación tiene más de un modificador, la flecha apuntará a un nodo intermedio que a su vez dirigirá otras flechas a los modificadores correspondientes. La intensidad de la relación, si existe, estará marcada en rojo, y el tipo de enlace causal en negro. Como se puede ver en el grafo (figura 31), hay cuatro nodos con la palabra *smoking*, o similar, y otros cuatro con las palabras *lung cancer*. El resto de nodos se han creado durante el proceso. Por lo tanto, con un grafo como éste se pueden establecer nuevas relaciones ocultas a primera vista, como por ejemplo la existente entre *smoking* y *fluid being acumúlate around the lungs*, con un cierto grado de relación, teniendo en cuenta los modificadores y cuantificadores que aparecen en el camino causal que enlazan estos dos nodos.

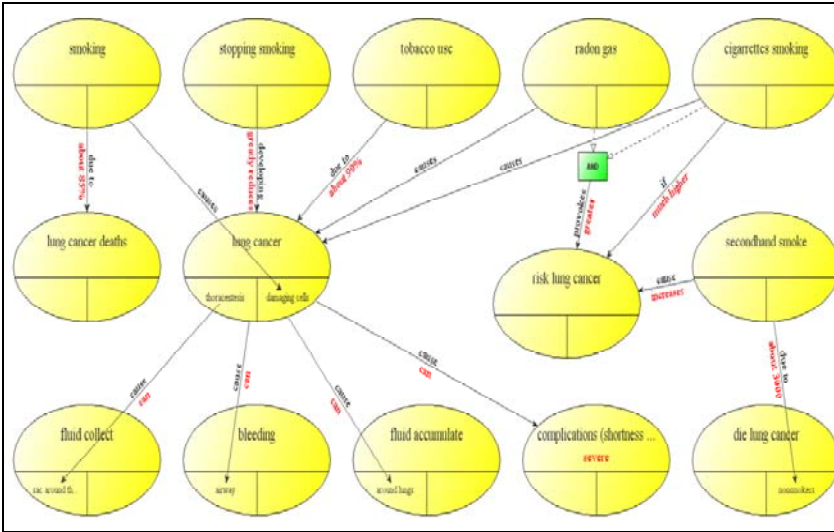


Figura 31: Grafo causal creado automáticamente que refleja la relación entre *smoking* y *lung cancer*

## Conclusiones y líneas de actuación

En este trabajo se ha esbozado qué es un sistema de recuperación de información, se han descrito con bastante detalle las principales herramientas de búsqueda existentes, principalmente los buscadores Web y los Metabuscadors, y como mejorar la eficacia en su uso cotidiano. Para ello se ha presentado el papel fundamental que juegan los usuarios y el lenguaje natural en el rendimiento de las búsquedas Web. Posteriormente se ha tratado de presentar qué papel juegan y podrían jugar las técnicas de Inteligencia Artificial y en particular las relacionadas con el *Soft-computing* para mejorar la eficacia de estas herramientas. Por último se han dibujado unas propuestas ya desarrolladas, tanto de herramientas reales como conceptuales, que tratan de ir un paso más allá en la mejora de la eficacia de las búsquedas Web.

Teniendo en cuenta estas ideas y otras que se puedan aportar, sería posible crear realmente buscadores borrosos, o lo que es lo mismo, buscadores con la capacidad de realizar búsquedas en términos de significados aproximados. El principal punto de referencia para estos buscadores debería ser la Web, no tanto para los buscadores generales sino más bien para los Metabuscadors, porque éstos usan como base los motores de búsqueda general.

El hecho de tener buscadores borrosos ofrecería la posibilidad de hacer interesantes *tests* y experimentos. La Inteligencia Artificial es un área donde se mezclan distintas lógicas, porque las aproximaciones en el análisis formal de una frase pueden ser diferentes. La forma lógica de la siguiente frase, un poco larga pero no extraña: “Supongo que crees que te llamaré un poco más tarde”, implica el uso de diferentes lógicas para su formalización: creencia, no monotónica, borrosa, temporal,... Pero el problema es más complejo, porque, para las palabras con significado vago puede haber varias modalidades de lógicas borrosas. La elección no es un factor que haya sido estudiado demasiado. Los Metabuscadors podrían proveer de un marco muy útil, restringido por el lenguaje que manejen, para investigar en la variedad de formalismos que la lógica borrosa provee.

El uso de perfiles de usuario en Metabuscadors Web podría aportar algunas ventajas para mejorar la búsqueda. El perfil de usuario puede ser otro parámetro a tener en cuenta para expandir las consultas (con

conceptos relacionados con el perfil: sinónimos, hipónimos, etc.), para seleccionar los buscadores y adaptar las consultas a ellos y para elegir y establecer un *ranking* entre los resultados de la búsqueda. Las técnicas de *Soft-Computing* pueden ayudar en el aprendizaje y las tareas de representación.

¿*Meta Sistemas Pregunta-Respuesta?*, quizá el próximo objetivo podría ser construir Meta Sistemas Web Pregunta-Respuesta, los cuales analizan la pregunta del usuario y generan un conjunto de consultas precisas (consultas expandidas) que envían al directorio o motor de búsqueda más adecuado, para lograr “aislar” la respuesta concreta y correcta a la pregunta formulada por el usuario. Las técnicas de *Soft-Computing* y principalmente la lógica borrosa, como herramienta más cercana a las expresiones humanas, pueden jugar papel esencial para detectar la intención y el significado correcto de las preguntas del usuario del sistema.

## Bibliografía

- Agichtein, E., Brill, E., Dumais, S. (2006). "Improving Web Search Ranking by Incorporating User Behavior Information". *En Proc. of the SIGIR'06 Conference*.
- Armstrong, R., Freitag, D., Joachims, T., Mitchell, T. (1995). "WebWatcher: A Learning Apprentice for the World Wide Web". *En Proc. of the 1995 AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed environments*.
- Aslam, J. A., Montague, M. (2001). "Models for Metasearch". *En Proc. of 24th International ACM SIGIR*, (pp. 276-283).
- Baeza-Yates, R., Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Essex, UK: Addison-Wesley.
- Baeza Yates, R. (2000). "Desenredando La Madeja". *NOVATICA*, May-jun 2000, 25 aniversario, (pp. 72-77).
- Bates, M. J. (1979). "Information Search Tactics". *Journal of the American Society for Information Science*, 30, 205-214.
- Belkin, N. J., Oddy, R. N., Brooks, H. M. (1982). "ASK for information retrieval: Part I. Background and theory". *Journal of Documentation*, 38(2), 61-71.
- Bezdek, J. C., Pal, S. K. (eds.) (1991). *Fuzzy Models for Pattern Recognition (Methods that search for structures in data)*. IEEE Press.
- Bollacker, K. D. (2000). "Discovering Relevant Scientific Literature on the Web". *IEEE Intelligent Systems* 15(2), 42-47.
- Bowman, C. M., Danzig, P. B., Hardy, D. R., Manber, U., Schwartz, M. F. (1994). "The Harvest information discovery and access system". *En Proc. 2nd WWW Conference*, (pp. 763-771).
- Brin, S., Page, L. (1998). "The Anatomy of a Large-Scale Hypertextual Web Search Engine". *En Proc. of 7th WWW Conference*.
- Broder, A. (2002). "A taxonomy of web search". *SIGIR Forum* 36(2).
- Bruza, P. D. (1993). "Stratified Information Disclosure: A Synthesis between Information Retrieval and Hypermedia". PhD thesis, University of Nijmegen, Nijmegen, The Netherlands.
- Buckley, C., Salton, G., Allan, J., Singhal, A. (1995). "Automatic Query Expansion Using SMART: (TREC-3)". *En Proc. of The Third Text Retrieval Conference (TREC-3)*.

- Casasola, E., Gauch, S. (1997). "Intelligent Information Agent for the World Wide Web". *Information and Telecommunication Technology Center*, Technical Report: IITTC-FY97-11100-1.
- Castro, J. L., Trillas, E., Zurita, J. M. (1998). "Non-monotonic fuzzy reasoning". *Fuzzy Sets and Systems* 94, 217-225.
- Chen, Z. (2000). "WebSail: from on-line learning to Web search". *En Proc. of the First Int. Conf. on Web Information Systems Engineering, Vol. 1*, (pp. 206-213).
- Cooley, R., Mobasher, B., Srivastaba, J. (1997). "Grouping web page references into transactions for mining world wide web browsing patterns". Technical report TR 97-021, University of Minnesota, Minneapolis.
- De la Mata, J.; Olivas, J. A.; Serrano-Guerrero, J. (2004). "Overview of an Agent Based Search Engine Architecture". *En Proc. of the Int. Conf. on Artificial Intelligence IC-AI'04, Las Vegas, USA. CSREA Press, Vol. 1*, (pp. 62-67).
- Delgado, M., Martín-Bautista, M.J., Sánchez, D., Serrano, J.M., Vila, M.A. (2003). "Association rules and fuzzy associations rules to find new query terms". *En Proc. of the EUSFLAT'03 Conference*, (pp. 49-53).
- Efthimiadis, E. N. (1996). "Query Expansion". *ARIST* 31, 121-187.
- Fensel, D. (1999). "On2Broker: Semantic-Based Access to Information Sources at the WWW". *En Proc. of the World Conf. on the WWW and Internet (WebNet 99)*, (pp. 25-30).
- Fernández, S., Sobrino, A. (2000). "Hacia un tratamiento computacional de la sinonimia". *En Revista de la Sociedad Española para el Procesamiento del Lenguaje Natural (SEPLN)* 26, 89-95.
- Fernández López, M., Gómez-Pérez, A., Pazos, J., Pazos, A. (1999). "Building a Chemical Ontology Using Methontology and the Ontology Design Environment". *IEEE Intelligent Systems* 14(1), 37-46.
- Fox, S., Karnawat, K., Mydland, M., Dumais, S. T., White, T. (2005). "Evaluating implicit measures to improve the search experience". *ACM Transactions on Information Systems*.
- Gruninger, M., Fox, M. S. (1995). "Methodology for the Design and Evaluation of Ontologies", *En Proc. of Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI-95*, Montreal.
- Furnas, G. W., Landauer, T. K.; Gomez, L. M.; Dumais, S. T. (1987). "The Vocabulary Problem in Human-System Communication". *Communications of the ACM*, 30(11), 964-971.
- Garcés, P., Olivas, J. A., Romero, F. P. (2006). "Concept-matching IR systems versus Word-matching IR systems: Considering fuzzy interrelations for indexing web pages". *Journal of the American*



*Society for Information Science and Technology JASIST*, 57(4), 564-576.

Glover, E. J., Lawrence, S., Virmingham, W. P., Giles, C. L. (1999). "Architecture of a Metasearch Engine that Supports User Information Needs". *En Proc. of the Eighth International Conference on Information Knowledge Management, (CIKM-99)*, (pp. 210-216).

Guarino, N. (1999). "OntoSeek: content-based access to the Web". *IEEE Intelligent Systems*, 14(3), 70-80.

Herrera-Viedma, E., Pasi, G. (2003). "Fuzzy approaches to access information on the Web: recent developments and research trends". *En Proc. of the Third Conference of the EUSFLAT*, (pp. 25-31).

Hull, D., Grefenstette, G. (1996). *A detailed analysis of English stemming algorithms*. Rank Xerox Research Centre: XEROX Technical Report.

Hofstede, A. H. M., Proper, H. A., Van Der Weide, P. (1996). "Query Formulation as an Information Retrieval Problem". *The Computer Journal* 39, 255-274.

Ingwersen, P. (1992). *Information Retrieval Interaction*. London: Taylor Graham.

Ingwersen, P., Willet, P. (1995). "An introduction to algorithmic and cognitive approaches for information retrieval". *Libri*, 45, 160-177.

Joachims, T., Granka, L., Pang, B.; Hembrooke, H., Gay, G. (2005). "Accurately Interpreting Clickthrough Data as Implicit Feedback". *Proc. of the ACM Conf. on Research and Development on Information Retrieval*.

Kelly, D., Teevan, J. (2003). "Implicit feedback for inferring user preference: A bibliography". *En Proc. of the SIGIR 03 Conference*.

Kim, H. J., Lee, S. G. (2003). "Building topic hierarchy based on fuzzy relations". *Neurocomputing* 51, 481-486.

Kowalski, G. (1997). *Information retrieval systems: Theory and implementation*. The Netherlands: Kluwer Academic Publishers.

Kerschberg, L., Kim, W., Scime, A. (2001). "A Semantic Taxonomy-Based Personalizable Meta-Search Agent". *Second Int. Conf. on Web Information Systems Engineering (WISE'01)*.

Kohonen, T. (1984). *Self-Organization and Associative Memory*. Berlín: Springer-Verlag.

King-Ip Lin, Kondadadi, R. (2001). "A similarity-based soft clustering algorithm for documents". *En Proc. of the 7th International Conference on Database Systems for Advanced Applications DASFAA '01*, (pp. 40-47).

Kiryakov, A. K., Simov, K. I. (1999). "Ontologically supported semantic matching". *En Proc. of NODALIDA '99: Nordic Conference on Computational Linguistics, Trondheim*.

- Krovetz, R. (1993). "Viewing morphology as an inference process". *En Proc. of ACM SIGIR 93*, 16, (pp. 191-202).
- Kuhlthau, C. (1988). "Longitudinal case studies of the information search process of users in libraries". *Library and Information Science Research 10*, 257-304.
- Lafourcade, M., Prince, V. (2001). "Relative Synonymy and conceptual vectors". *En Proc. the Sixth Natural Language Processing Pacific Rim Symposium*, (pp. 127-134).
- Lawrence, S., Giles, C. L. (1999). "Accessibility of Information on the Web". *Nature 400*, 107-109.
- Leacock, C., Chodorow, M. (1998). "Combining local context and Wordnet similarity for word sense disambiguation". *En WordNet, an Electronic Lexical Database, MIT Press, Cambridge Ma*, (pp. 285-303).
- Lee, J. H. (1997). "Analyses of multiple evidence combination". *En Proc. of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, (pp. 267-275).
- Lempel, R., Moran, S. (2000). "The stochastic approach for link-structure analysis (SALSA) and the TKC effect". *En Proc. of the 9th WWW Conference*.
- Lempel, R., Moran, S. (2004). "Rank Stability and Rank Similarity of Link-Based Web Ranking Algorithms in Authority Connected Graphs", *Information Retrieval 8*(2).
- Lewandowski, D. (2006). "Query types and search topics of German web search engine users". *Information Services & Use 26*.
- Li, Z., Wang, Y., Oria, V. (2001). "A New Architecture for Web Meta-Search Engines". *En Proc. of Seventh Americas Conference on Information Systems*, (pp. 415-422).
- Loupy, C., El-Bèze, M. (2002). "Managing synonymy and polysemy in a document retrieval system using WordNet". *En Proc. of the LREC2002: Workshop on Linguistic Knowledge Acquisition and Representation*.
- Lovins, J. B. (1968). "Development of a stemming algorithm". *Mechanical Translation and Computational Linguistics 11*(1-2), 22-31.
- Martin, P., Eklund, P. W. (2000). "Knowledge retrieval and the World Wide Web". *IEEE Intelligent Systems 14*(3), 18-25.
- Martin-Bautista, M.J., Vila, M., Kraft, D., Chen, J. (2001). "User profiles and fuzzy logic in web retrieval". *En Proc. of the 1st BISC Int. Workshop on Fuzzy Logic and the Internet*, UC Berkeley, USA, (pp. 19-24).

- Meng, W., Yu, C., Liu, K-L. (2002). "Building Efficient and Effective Metasearch Engines". *En Proc. of ACM Computing Surveys* 34(1), (pp. 48-89).
- Miller, G. (ed.) (1990). "WORDNET: An online lexical database". *International Journal of Lexicography*, 3(4), 235-312.
- Miller, G. (1995). "WordNet: A lexical database for English". *Communications of the ACM* 11, 39-41.
- Mizzaro, S. (1996). "How many relevances in IR?". *En Proc. of the Workshop Information Retrieval and Human Computer Interaction*, GIST Technical Report GR96-2, Glasgow University, UK, (pp. 57-60).
- Odgen, C. K. (1972). *The Meaning of meaning. A Study on the Influence of Language upon Thought and of the Science of Symbolism*. London: Routledge and Kegan Paul.
- Olivas, J. (2000). "Contribución al estudio experimental de la predicción basada en categorías deformables borrosas". Tesis Doctoral, Universidad de Castilla-La Mancha, España.
- Olivas, J. A., Garcés, P. J., Romero, F. P. (2003). "An application of the FIS-CRM model to the FISS metasearcher: Using fuzzy synonymy and fuzzy generality for representing concepts in documents". *Int. Journal of Approximate Reasoning* 34, 201-219.
- Ounis, I., Amati, G., Plachouras, V., He, B., Macdonald, C., Lioma, C. (2006). "Terrier: A high performance and scalable information retrieval platform". *En Proc. of the twenty-ninth annual International ACM/SIGIR conf. on research and development in information retrieval*, ACM Press.
- Paice, C. D. (1990). "Another stemmer". *En Proc. of ACM SIGIR 90*, (pp. 56-61).
- Pasi, G. (2002). "Flexible information retrieval: some research trends". *Mathware and Soft Computing* 9, 107-121.
- Porter, M. F. (1980). "An Algorithm for Suffix Stripping". *Program*, 14(3), 130-137.
- Puente, C., Sobrino, A., Olivas J. A., Merlo, R. (2010). "Extraction, Analysis and Representation of Imperfect Conditional and Causal sentences by means of a Semi-Automatic Process". *En Proc. of the 2010 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'10)*.
- Quilliam, M. R. (1968). "Semantic Memory". Minsky, M. ed., *Semantic Information Processing*. Cambridge (MA), MIT Press, (pp. 27-70).
- Ramakrishnan, G., Prithviraj, B. P., Deepa E. (2004). "Soft word disambiguation". *En Proc. of the Second Global WordNet Conference*, 2004.

- Ricarte, I., Gomide, F. (2001). "A reference model for intelligent information search". *En Proc. of the 1st BISC Int. Workshop on Fuzzy Logic and the Internet*, UC Berkeley, USA, (pp. 80-85).
- Romero, F. P., Olivas, J. A., Garcés, P. (2007). "FzMail: Using FIS-CRM for e-mail classification". *Journal of Advanced Computational Intelligence and Intelligent Informatics (JACIII) 11(1)*, 40-50.
- Rose, D. E., Levinson, D. (2004). "Understanding user goals in web search". *En Proc. of the 13th international conference on World Wide Web*, ACM.
- Salton, G., McGill, M. H. (1983). *Introduction to Modern Information Retrieval*. New York: McGraw-Hill.
- Salton, G., Fox, E. A., Wu, H. (1983). "Extended Boolean information retrieval". *Communications of the ACM*, 26(11), 1022-1036.
- Selberg, E., Etzioni, O. (2000). "On the instability of web search engines". *En Proc. of Content-Based Multimedia Information Access (RIAO)*, (pp. 223-235).
- Serrano-Guerrero, J.; Romero, F. P.; Olivas, J. A. (2008). "Filtering Short Queries by means of Fuzzy Semantic-Lexical Relations for Meta-Searchers using WordNet". *En Proc. of the IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology (WI-IAT 2008)*, (pp. 269-272).
- Serrano Guerrero, J. (2009). "Characterization of Soft Computing-based Semantic Distances for Internet Search". Tesis Doctoral, Universidad de Castilla-La Mancha, España.
- Serrano-Guerrero, J., Romero, F., Olivas, J., de La Mata, J., Soto, A. (2009). "BUDI: Architecture for Fuzzy Search in Documental Repositories". *Mathware and Soft Computing 16*, 71-85.
- Shavlik, J., Eliassi-Rad, T. (1998). "Building intelligent agents for web-based tasks: A theory-Refinement approach". *En Proc. of the Conf. on Automated Learning and Discovery: Workshop on Learning from Text and the Web*, Pittsburgh, PA.
- Soto, A., Olivas, J. A., Prieto, M. E. (2007). "Using Generalized Constraints and Protoforms to Deal with Adjectives". *En Proc. of the FUZZ-IEEE 2007, IEEE International Conference on Fuzzy Systems*, Imperial College, London.
- Spink, A., Bateman, J., Jansen, B. J. (1999). "Searching the Web: survey of Excite users". *Internet Research: Electronic Networking Applications and Policies 9(2)*, 117-128
- Tang, Y., Zhang, Y. (2001). "Personalized library search agents using data mining techniques". *En Proc. of the 1st BISC Int. Workshop on Fuzzy Logic and the Internet*, (pp. 119-124).

- Terveen, L., Hill, W.; Amento, B., McDonald, D., Creter, J. (1997). "Phoaks: a system for sharing recommendations", *Communications of the ACM* 40(3), 59-62.
- Trillas, E. (2004). "Lógica borrosa y narrativa: un párrafo de Vila-Matas". *Actas del XII Congreso Español sobre Tecnologías y Lógica Fuzzy ESTYLF'04*, Universidad de Jaen, (pp. 15-20).
- Vertexera Inc.: "Benefits of Search Engine Optimization". [en línea]. Consultado el 19 de septiembre de 2011 en <<http://www.vertexera.com/wp/SEO.pdf>>.
- Wang, Z. (2004). "Improved Link-Based Algorithms for Ranking Web Pages". *En Proc. of The Fifth International Conference on Web-Age Information Management, LNCS 3129*, (pp. 291-302).
- Whaley, J. M. (1999). "An application of word sense disambiguation to information retrieval". Dartmouth College Computer Science Technical Report PCS-TR99-352.
- Widyantoro, D., Yen, J. (2001). "Incorporating fuzzy ontology of term relations in a search engine". *En Proc. of the 1st BISC Int. Workshop on Fuzzy Logic and the Internet*, UC Berkeley, USA, (pp. 155-160).
- Xue, G. R., Zeng, H. J., Chen, Z., Yu, Y., Ma, W. Y., Xi, W. S., Fan, W. G. (2004). "Optimizing web search using web click-through data". *En Proc. of the Conf. on Information and Knowledge Management*.
- Yager, R. R. (1988). "On ordered weighted averaging aggregation operators in multicriteria decisionmaking". *IEEE Transactions on Systems, Man and Cybernetics* 18(1), 183-190.
- Zadeh, L. A. (2003). "From search engines to Question-Answering system: The need for new tools". *En Proc. of the Atlantic Web Intelligence Conference - AWIC'2003*, LNCS, Springer.
- Zamir, O., Etzioni, O.: (2001): "Grouper: A dynamic clustering interface to Web search results". *Computer Networks*, 31(11-16), 1361-1374.

ESTA PUBLICACIÓN SE TERMINÓ DE IMPRIMIR  
EN EL MES DE OCTUBRE DE 2011,  
EN LA CIUDAD DE LA PLATA,  
BUENOS AIRES,  
ARGENTINA.

