RCCI Vol. 3, No. 1-2 ENERO- JUNIO, 2009 p. 41-47

Recibido: *Aceptado:*

Empleo de motores de búsqueda de código abierto para la recuperación de información vertical

Take up open source search engines for vertical information

Juan Manuel Fernández Luna, Juan Francisco Huete Guadix, Ramiro Pérez Vázquez, Julio César Rodríguez Cano, Carmen Torres López

[1] Departamento de Ciencias de la Computación e Inteligencia Artificial, E.T.S.I. Informática y de Telecomunicación - CITIC, Universidad de Granada, 18071, Granada, España. {jmfluna,jhg}@decsai.ugr.es

[2] Departamento de Ciencias de la Computación, Universidad de Las Villas, 50300, Santa Clara, Cuba. rperez@uclv.edu.cu

[3] Centro de Desarrollo de Software de Holguín, Universidad de las Ciencias Informáticas, 80100, Holguín, Cuba. jcrcano@uci.cu

[4] Facultad de Informática y Matemática, Universidad de Holguín, 80100, Holguín, Cuba. {jcrodriguez,ctorres}@facinf.uho.edu.cu

*Autor para la correspondencia.

Resumen

Los sistemas de recuperación de información que se utilizan en la actualidad con mayor frecuencia (Google, Yahoo!, y Windows Live Search) consideran como fuente documental toda la información disponible en Internet. Esta fuente documental crece con tanta celeridad que, en ocasiones, los usuarios interesados en algún tópico especializado, tienen que dedicar mucho tiempo y esfuerzo antes de encontrar documentación relevante que satisfaga su necesidad de información. La recuperación de información (RI) temática o vertical, supone ventajas frente a la RI genérica u horizontal, pues al especializarse en un contexto particular, los sistemas RI pueden analizar con mayor exhaustividad la fuente documental, disponer de resultados más actualizados, y ofrecerle al usuario herramientas ad-hoc. En este trabajo se presenta una comparación de cuatro motores de búsqueda de código abierto (Apache Lucene, Minion, Terrier, e Indri) que facilitan en gran medida el desarrollo de sistemas RI por las facilidades de reutilización y extensión de diferentes modelos RI, fundamentalmente el booleano, espa cio vectorial, y probabilístico. Además se describe un caso de estudio donde se propone al plug-in COSME (Code Search MEeting) para el entorno de desarrollo integrado NetBeans en el contexto de la búsqueda colaborativa de código fuente.

Palabras clave: Motor de búsqueda, sistema de recuperación de información, recuperación de información vertical, recuperación de información colaborativa, desarrollo de software guiado por búsquedas, búsqueda de código fuente.

Abstract

Information retrieval systems that nowadays are frequently used (Google, Yahoo!, and Windows Live Search) consider all the available information in Internet as document collection. This collection grows with so much velocity that sometimes, the users interested in some specialized topic, have to dedicate a lot of time and effort before to find relevant information that satisfies their information needs. The thematic or vertical information retrieval (IR), supposes advantages in front of generic or horizontal IR, because it be specialize in a particular context, then the IR systems can analyze with more exhaustiveness the collection, have more recent results, and offer ad-hoc tools to the user. In this work we shows up a comparison of four open source search engines (Apache Lucene, Minion, Terrier, and Indri) that facilitate in great measure the development of IR systems for the reuse facilities and extension of different IR models, fundamentally the boolean, vectorial space, and probabilistic. A case of study is also described to propose COSME (Code Search MEeting) as a plug-in for the integrated development environment NetBeans in the context of collaborative source code search.

Keywords: Search engine, information retrieval system, vertical information retrieval, collaborative information retrieval, search driven software development, source code search.

Introducción

Con los avances tecnológicos de la Informática, la cantidad de información dispuesta en forma de texto plano, páginas Web, imágenes, música, vídeos, entre otros, tiende a crecer exponencialmente a todo lo largo y ancho de Internet. Así, cuando una persona busca dentro de volúmenes tan extensos de información, que además presentan altos grados de incertidumbre, pues esta información en muchos casos no está ordenada, encontrar lo que desea implica dedicar gran cantidad de tiempo y esfuerzo. Esta situación aceleró la evolución de la RI, disciplina encargada del desarrollo de técnicas que permitan encontrar materiales (usualmente documentos) de naturaleza no estructurada (texto normalmente) que satisfagan una necesidad de información a partir de grandes colecciones de información (almacenadas generalmente en ordenadores) de forma eficiente (Manning, 2008).

Una de las herramientas que han surgido en los últimos años para lograr la meta que se persigue en RI son los denominados motores de búsqueda. Éstos consisten en un conjunto de APIs (Application Programming Interface) basadas en algún modelo RI (booleano, espacio vectorial, probabilístico, etc.) y que ayudan al desarrollo ágil de los sistemas RI. El objetivo principal de los sistemas RI consiste en predecir, con un conocimiento previo sobre el usuario y la fuente documental, qué documentos son los más adecuados para que el usuario pueda satisfacer su necesidad de información.

Actualmente existen dos tendencias fundamentales en el desarrollo de sistemas RI según el contexto

y el ámbito de la fuente documental: a) la RI vertical se enfoca en la indexación de fuentes documentales específicas (por ejemplo, la bolsa de empleo de un país determinado), y b) la RI horizontal se enfoca en fuentes documentales generales (por ejemplo, la Web). Tradicionalmente los sistemas RI horizontales han gozado de mayor popularidad, sin embargo, por las ventajas que ofrecen los sistemas RI verticales, varias empresas han extendido sus servicios para incluir la búsqueda en fuentes documentales especializadas, enfocadas en contextos como la medicina, viajes, música, videos, etc. (Arguello, 2009). Por ejemplo, la empresa Google Inc. desarrolló Google Code Search y Google Scholar en los contextos del desarrollo de software guiado por búsquedas (Bajracharya, 2009) y las bibliotecas digitales (Smeaton, 2005) respectivamente. En el presente trabajo se describen cuatro de los motores de búsquedas más populares, tanto en la academia como en la industria de ciencias de la información: Apache Lucene, Minion, Terrier, e Indri. Además se abordará como un caso de estudio de RI vertical la integración de motores de búsquedas al IDE (Integrate Development Environment) NetBeans.

La estructura de este trabajo es la siguiente. En la Sección 2 se abordan elementos importantes a tener en cuenta durante la selección de un motor de búsqueda, que luego será empleado para desarrollar un sistema RI, estableciéndose una comparación entre los cuatro motores de búsqueda mencionados. En la Sección 3 se describe un caso de estudio sobre RI vertical en el contexto de la búsqueda colaborativa de código fuente, donde se detallan las principales características del sistema RI propuesto. Finalmente, en la Sección 4, se resumen las conclusiones y trabajos futuros por donde discurrirá esta investigación.

Materiales y Métodos

Un elemento importante a tener en cuenta durante la selección de un motor de búsqueda es el modelo RI que éste implementa. Un modelo RI es una especificación de la representación de los documentos y las consultas, más la forma en que se compararán para recuperar los documentos relevantes (Fernández-Luna, 2001). Baeza-Yates y Ribeiro-Neto presentan en (Baeza-Yates, 1999) una caracterización formal de dicho concepto. Varios modelos RI se han elaborado desde los inicios de la primera mitad del siglo pasado. Éstos pueden ser clasificados según se plantea en (Dominich, 2008) como modelos clásicos (booleano, espacio vectorial, probabilístico), modelos no clásicos (lógica de la información, teoría de situaciones, asociativo), modelos alternativos o híbridos (agrupamiento, lógica difusa, redes neuronales artificiales, algoritmos genéticos, procesamiento natural del lenguaje, base de conocimientos), y modelos Web (análisis de enlaces, exploración Web, visualización). Sin embargo, aunque estos modelos presentan diferencias sustanciales unos con otros, todos tienen las siguientes características en común:

1. Son típicamente basados en texto, donde se identifican conjuntos de términos, los cuales constituyen el mecanismo fundamental para las búsquedas. El número de ocurrencias de los términos es usado para representar el contenido de los documentos mediante algún formalismo matemático (usualmente asignándole pesos a los términos).
2. Una consulta es concebida como un fragmento de texto (por lo tanto, también puede ser representada como un conjunto de términos).
3. En el caso de las páginas Web, además de analizar su contenido representándolas mediante un conjunto de términos, es importante destacar la característica distintiva que presentan éstas al estar enlazadas unas con otras formando una red.
4. El grado de relevancia o similitud entre una consulta y un documento (texto plano, páginas Web, documentos pdf, etc.) se establece a través de algún cálculo numérico basado en los pesos de los términos o importancia de los enlaces, en el caso de las páginas Web.

El primer paso que debe dar un sistema RI, el cual puede ser elaborado sobre la base de un motor de búsqueda, consiste en procesar la fuente documental, dejándola en un formato cuya manipulación por parte del sistema sea fácil y rápida. Por tanto, a partir de un documento se generará una representación del mismo, formada por una secuencia de términos de indexación, los cuales mantendrán lo más fielmente posible el contenido original del documento. A este proceso general se le denomina indexación automático de texto. Luego, la tarea más común de un sistema RI está centrada en la recuperación; en ella el sistema intenta proveer documentos que son relevantes a una necesidad de información del usuario (representada a través de una consulta en lenguaje natural) a partir de los índices creados durante el proceso de indexación.

Para conocer la calidad de la recuperación, el sistema RI puede incluir además un módulo que

realice la actividad de evaluación. Para facilitar esta actividad, la comunidad científica ha creado métricas y fuentes documentales de prueba. Éstas están definidas por un conjunto de documentos, un grupo de necesidades de información expresadas como consultas, y un conjunto de juicios de relevancia previamente emitidos. En las últimas décadas se han propuesto numerosas colecciones de prueba, entre las más populares está la colección de Cranfield, pionera en permitir medidas cuantitativas precisas de efectividad para RI (actualmente es considerada muy pequeña, excepto para experimentos elementales) y la célebre Conferencia de Recuperación de Texto (TREC).

Otro elemento importante a considerar durante la identificación del motor de búsqueda más apropiado es el costo económico. En este sentido los motores de búsqueda de código abierto resultan mucho más factibles respecto a los comerciales. De forma general, el software de código abierto incluye numerosos beneficios, como es poder conocer las interioridades de lo que se está usando, modificar o adaptar el código según se requiera, identificar errores e ineficiencias a tiempo, no depender de terceros para redistribuir las nuevas versiones mejoradas, entre otras (Feller, 2005). En este trabajo se presenta una comparación entre cuatro de los motores de búsqueda de código abierto más populares:

Apache Lucene: fue escrito originalmente por Doug Cutting, el cual se unió a la familia de la Fundación Apache en septiembre de 2001. En febrero de 2005, Lucene, se convirtió en un proyecto de muy alto nivel. Durante varios años ha sido considerado uno de los motores de búsqueda más populares. Se utiliza en un gran número de sitios y aplicaciones, tales como el IDE Eclipse, la Enciclopedia Británica, la Wikipedia, el buscador de código fuente Krugle. Es el núcleo del motor de búsqueda Web Nutch.

Minion: está diseñado para ser altamente configurable y se propone que sea usado en investigaciones, así como en entornos empresariales. Presenta indexación y recuperación concurrente. En mayo 2008, se publicó su código fuente, y la versión desarrollada hasta el momento es parte del proyecto de Tecnologías para la Búsqueda Avanzada en los laboratorios de la compañía Sun Microsystems/ Oracle. Actualmente es utilizado por el sistema de recomendación Aura, desarrollado en los laboratorios de Sun, así como en aplicaciones Web de la misma compañía.

Terrier: se comenzó a desarrollar en el 2001 en la Universidad de Glasgow, Escocia y ha estado disponible al público desde noviembre de 2004. Su nombre es un acrónimo de TERabyte RetrIEvER. Permite realizar experimentación con fuentes documentales de gran escala. Desde la publicación de su API, alrededor de 40 instituciones académicas y empresariales manifiestan interés en su utilización y mejora continua.

Indri: es un motor de búsqueda basado en el proyecto Lemur; consiste en un esfuerzo cooperativo entre la Universidad de Massachusetts y la Universidad de Carnegie Mellon. El sistema se concluyó en el 2004. Desde la perspectiva empresarial, es eficiente y fácil de integrar en sistemas RI, aunque además es ampliamente utilizado en el ámbito académico.

En la Tabla 1 se presenta una comparación entre estos cuatro motores de búsqueda teniendo en cuenta los siguientes indicadores [1]:

1. Estructura del índice: indica la forma en que el indexador almacena el índice, por ejemplo, usando un motor de base de datos o una estructura de fichero ad-hoc.

2. Índice incremental: indica si el indexador es capaz de añadir ficheros a un índice existente sin la necesidad de regenerar un índice completo.

3. Indexación por campos: se refiere a que en el índice, los documentos estarán estructurados por campos, lo que posibilita poder realizar búsquedas avanzadas.

4. Tipo de documentos que indexa: son los tipos de documentos que el indexador es capaz de analizar gramaticalmente.

5. Modelo RI: modelo que implementa el motor de búsqueda correspondiente.

6. Creación de resumen: indica si tiene la capacidad de hacer un resumen del contenido de un documento recuperado, generalmente, se resaltan los términos que coincide con las consultas.

7. Tipo de operadores de búsqueda: tipo de búsquedas que es capaz de hacer, y si acepta operadores de consulta (tiene en cuenta el lenguaje de consulta de cada motor).

8. Lenguaje de programación: Lenguaje de programación usado para implementar el indexador y las herramientas de búsqueda. Esta información es útil en orden de extender las funcionalidades o integrarlas a un sistema RI.

9. Lematización: expresa si el indexador es capaz de hacer operaciones de lematización (términos con el mismo origen) sobre las palabras.

[1] Esta comparación fue realizada tomando como base el trabajo realizado por Middleton y Baeza-Yates (Middleton, 2007).

10. Índice sensible a mayúsculas y minúsculas: El índice tiene en cuenta las mayúsculas y minúsculas del texto en cada documento.
11. Plataforma: Indica la plataforma o sistema operativo sobre el se puede desplegar el sistema RI correspondiente.
12. Licencia: Determina las condiciones para usar y modificar el motor de búsqueda.

Tabla 1. Comparación entre los motores de búsqueda.

Características	*Apache Lucene*	*Minion*	*Terrier*	*Indri/Lemur*
Estructura del índice:	Índice invertido	Índice invertido	Índice invertido	Índice invertido
Índice incremental:	Sí	Sí	No	Sí
Indexación por campos:	Sí	Sí	Sólo para fuentes documentales TREC.	Sí
Tipo de documentos que indexa:	HTML, TXT, PDF, XML, PPT, DOC, RTF	Texto plano	HTML, TXT, PDF, XML, PPT, DOC, RTF, y fuentes documentales TREC	HTML, TXT, PDF, XML, PPT, DOC, RTF y fuentes documentales TREC
Modelo RI:	Combinación del modelo espacio vectorial y el booleano	Combinación del modelo booleano, consultas relacionales y de proximidad	Divergencia de aleatoriedad, basado en el modelo probabilístico	Combinación del modelo de redes de inferencia y modelado de lenguaje natural
Creación de resumen:	Sí	Sí	No	Sí
Tipo de operadores de búsqueda:	Booleanos, frases, términos wildcard, y campos	Booleanos, proximidad, pasajes, y campos	Booleanos, operadores +/-, frases, proximidad, y campos	Booleanos, frases, sinónimos, términos wildcard, pasajes, y campos
Lenguaje de programación:	Java, C++, C#, Python, Perl, Ruby, y PHP	Java	Java	Java, C++, C#, y PHP
Lematización (Stemming):	Sí	Sí	Sí	Sí
Índice sensible a mayúsculas y minúsculas:	Sí	Sí	Sólo para fuentes documentales TREC	Sí
Plataforma:	Windows, Mac OS, GNU/Linux, y Unix	Windows, Mac OS, GNU/Linux, y Unix	Windows, Mac OS, GNU/Linux, y Unix	Windows, Mac OS, GNU/Linux, y Unix
Licencia:	Apache Software Foundation (ASF)	GNU Public License (GPL)	Mozilla Public License (MPL)	Berkeley Software Development (BSD)

Resultados y Discusión

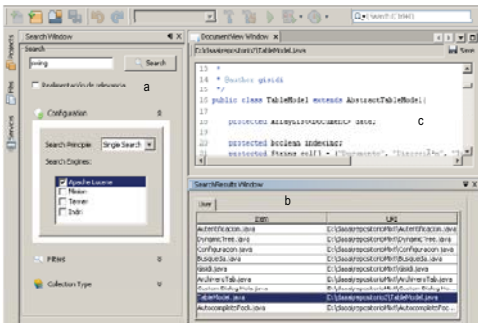

Figura 2. Ventanas colapsables que muestran algunas de las principales funcionalidades de COSME Client: a) Panel de búsqueda, b) Resultados de búsqueda, y c) visualizador de resultados.

El desarrollo de software guiado por búsquedas constituye un casos especial de RI vertical, dado que recientemente los sistemas RI se han convertido en herramientas ampliamente utilizadas por los desarrolladores de software (Bajracharya, 2009). Actualmente existen varios sistemas RI especializados en la búsqueda de código fuente, entre los más destacados se encuentran Codase, Codefetch, Google Code Search, Koders, y Krugel. K. Krugler y J. D. Mitchell enfatizan en (Krugler, 2007) que "Alrededor del 25% del tiempo de un desarrollador es ocupado en búsquedas de información técnica. Aunque este tiempo está bien empleado, porque encontrar código reutilizable puede propiciar que un proyecto esté terminado en tiempo y con resultados de alta calidad". Ellos ejemplifican además con cinco razones sobre por qué la búsqueda es una herramienta importante en el desarrollo de software:

1. La búsqueda que ya se realiza cada día (búsquedas de soluciones a un problema en blogs y listas de correo, búsquedas de ejemplos de código fuente).
2. La gran cantidad de información técnica existente (se puede valorar en millones la cantidad de libros que contienen información técnica y

fragmentos de código fuente disponibles en Internet).
3. Encontrar código para reutilizar (A menudo ocurre que alguien ha trabajado con el mismo API, enfrentado el mismo error y resuelto el mismo problema).
4. Comprensión del código justo a tiempo (Para ser eficientes se requiere de una búsqueda altamente efectiva, que pueda ser usada cuando se necesite para encontrar las respuestas justo a tiempo).
5. Motores de búsqueda centrados en código (Las soluciones más débiles proveen un poco más que búsquedas basados en texto sobre ficheros que son solo código, mientras que las soluciones más fuertes están construidas para entender realmente el código como lenguajes de programación).

Por otro lado, estudios efectuados han demostrado que la búsqueda es también una actividad colaborativa, incluida la búsqueda de código fuente. La disciplina RI colaborativa es la encargada de establecer técnicas para satisfacer necesidades compartidas de información de grupos de usuarios, a partir de la extensión del proceso de recuperación de información con el conocimiento sobre las consultas, el contexto y los hábitos de colaboración explícita entre los usuarios (Rodríguez-Cano, 2010). Sin embargo, la mayoría de los sistemas RI enmarcados en el contexto del desarrollo de software guiado por búsquedas cuentan con interfaces orientados a una interacción individual, dificultando así la búsqueda colaborativa de código fuente. Esta dificultad fue la principal motivación para el desarrollo del plug-in COSME (ver Fig. 2).

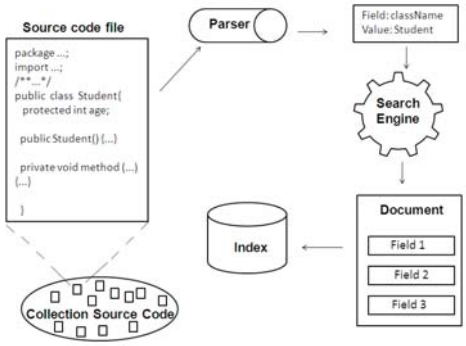

Figura 3. Indexación por campos en un fichero de código fuente.

COSME es un plug-in para el IDE NetBeans desarrollado sobre la base de NBMs (NetBeans Modules) y la plataforma CIRLab (Fernández-Luna, 2009). Este plug-in permite que los desarrolladores de software con necesidad de información compartida puedan realizar búsquedas colaborativas de código fuente; básicamente cuenta con dos aplicaciones: I) COSME Server es una herramienta interactiva para la administración de los servicios de indexación (ver Fig. 3) de las fuentes documentales (repositorios de código fuente Java, bibliografía en formato pdf, etc.) y de la comunicación con diferentes motores de búsqueda y técnicas RI colaborativas, y II) COSME Client es una aplicación embebida en el IDE NetBeans basada en un conjunto de ventanas que le facilitan al desarrollador interactuar con las diferentes funcionalidades que soporta COSME.

Un NBM es un grupo de clases Java que proveen un API específico para desarrollar extensiones al IDE NetBeans. Para el desarrollo de los NBMs utilizados en COSME se instanció NetBeans Platform (marco de trabajo sobre el cual está desarrollado el propio IDE) a través de los siguientes APIs:

- Dialog System: Este módulo maneja los diálogos y la infraestructura usada en el NetBeans.
- File System: Se utiliza para acceder a ficheros del usuario en el disco, ficheros dentro de JARs y datos de configuración para el IDE.
- Module System: Permite obtener una lista de los módulos instalados, inspeccionar sus propiedades y dependencias, cómo son adicionados y administrados.
- UI Utilities: Provee métodos de ayuda para varias tareas relacionadas a la UI.
- Utilities: Provee métodos de ayuda y definiciones básicas que incluyen acciones, lookup, manejo de imágenes, ejecución paralela y otros.
- Window System: Permite a los módulos incluir componentes de ventanas, principalmente a través de los componentes visuales embebidos conocidos por top-components.
- Settings: permite almacenar las configuraciones definidas por el módulo en varios formatos.
- Datasystems: constituye una capa por encima del Filesystems API, que reconoce y agrupa objetos de datos y les da una semántica en particular. Es responsable de escanear ficheros de un directorio en el disco, y para determinar el tipo de dato que cada uno representa.
- Nodes: se utiliza para la visualización de objetos en el NetBeans, tales como estructuras de árboles y otros.

Para lograr incorporar en COSME funcionalidades de RI colaborativa tales como la división del trabajo, la transferencia de conocimiento y la conciencia de grupo, se instanció la plataforma

CIRLab a través de las siguientes APIs:

- CIRLabCommon: contiene las clases necesarias para la manipulación de cada uno de los motores de búsqueda que incluye (Apache Lucene, Minion, Terrier, e Indri), así como las clases que corresponden al control de eventos, de excepciones, manejo de la comunicación y el trabajo con las sesiones de búsqueda colaborativa.
- CIRLabServer: posee las clases para la persistencia, para la evaluación, los diferentes controladores y garantiza la atención a solicitudes realizadas por las aplicaciones clientes empleando el patrón Modelo Vista Controlador (MVC).
- CIRLabUI: contiene modelos personalizados para el trabajo con la actualización síncrona y asíncrona de componentes visuales a partir del modelo de eventos distribuidos que emplea CIRLab.
- CIRLabClient: incluye las clases que permiten realizar las diferentes peticiones al servidor según las acciones realizadas por los usuarios.

Conclusiones

En este trabajo se presentó una comparación entre cuatro de los motores de búsqueda que actualmente gozan de gran popularidad en la comunidad RI. Además se presentó al caso de estudio de RI vertical COSME, el cual evidenció la sinergia que existe entre las disciplinas Desarrollo de Software Guiado por Búsqueda y la Recuperación de Información Colaborativa. Durante la elaboración de COSME se utilizó la plataforma CIRLab, la cual agilizó en gran medida el desarrollo de este plug-in para el IDE NetBeans, al incluir varias técnicas de RI colaborativa, modelos de eventos distribuidos y estar basada en el patrón MVC. Como trabajos futuros de esta investigación se desprende, por un lado, la utilización de técnicas de RI peer-to-peer (P2P) con el objetivo de darle soporte a una de las prácticas comunes de los desarrolladores de software, la cual consiste en utilizar sus propias estaciones de trabajo como fuentes documentales relevantes, y por otro lado, la utilización de técnicas de realimentación por relevancia para la transferencia de conocimiento y la división del trabajo.

Agradecimientos

Esta investigación ha sido desarrollada con el apoyo de la Asociación Universitaria Iberoamericana de Postgrado (AUIP), la Junta de Andalucía y el Ministerio de Ciencia e Innovación de España, bajo los términos del Proyecto TIN2008-06566-C04-01. Además queremos agradecer a Jeffrey H. Alexander, investigador de la Corporación Oracle y uno de los autores principales de Minion, por aclarar nuestras dudas sobre este motor de búsqueda desarrollado recientemente.

Referencias

Arguello, J., et al. Sources of Evidence for Vertical Selection. En: SIGIR'09, Boston, Massachusetts, USA, 2009. 19–23 p.

Baeza-Yates, R., and Ribeiro-Neto, B. Modern Information Retrieval. Addison Wesley Longman, New York, 1999.

Bajracharya, S., Kuhn, A., and Ye, Y. Sourcerer: An Internet-Scale Software Repository. En: First International Workshop on Search-Driven Development - Users, Infrastructure, Tools and Evaluation (SUITE). Vancouver, Canada, 2009.

Dominich, S. The Modern Algebra of Information Retrieval. Springer Verlag, Berlin Heidelberg, 2008.

Feller, J. Perspectives on Free and Open Source Software. MIT Press, 2005.

Fernández-Luna, J. M. Modelos de Recuperación de Información Basados en Redes de Creencia. Tesis doctoral, Universidad de Granada, España, Mayo de 2001.

Fernández-Luna, J. M, Huete, J. F., Pérez, R., Rodríguez-Cano, J. C. CIRLab: A groupware framework for collaborative information retrieval research. Information Processing & Management, 2009. In press.

Krugler, K. and Mitchell, J.D. Search-driven development: Five reasons why search is your most powerful tool. [January 2007] ; Available from: http://www.linuxworld.com/news/2007/012907-search.html.

Manning, C. D., Raghavan, P., and Schütze, H. An Introduction to Information Retrieval. England, Cambridge University Press, 2008. 544 p.

Middleton, C., and Baeza-Yates, R. A Comparison of Open Source Search Engines. Volume, 46, 2007.

Rodríguez-Cano, J.C. Técnicas de Recuperación de Información Colaborativa en Entornos Distribuidos. Diploma de Estudios Avanzados, Universidad de Granada, España, Febrero de 2010.

Smeaton, A. F., and Callan, J. Personalization and Recommender Systems in Digital Libraries. En: International Journal of Digital Libraries, 2005. 299-308 p.

Rafael Villares Orellana De la serie: De los objetos
y otras manipulaciones " El Precio "
1er Lugar

VI Salón de Arte Digital
Centro Cultural Pablo de la Torriente Brau