Resumen Tema 2: Crawling

José Alberto Benítez Andrades

Diciembre 2010

En este trabajo se resumen las conclusiones obtenidas después de haber realizado la lectura de los artículos propuestos de Sergy Brin amf Lawrence Page, Junghoo Cho, Héctor García-Molina, además de los que tratan sobre el rastreador Mercator, escritos por Allan Heydon y Marc Najork.

1. Introducción : Qué es un crawler.

Un *crawler* (rastreador) es un programa que recupera páginas web, comunmente usado por los motores de búsqueda [Pinkerton 1994]. Los rastreadores comienzan por una web inicial, que podemos llamar P_0, recuperan esta página, extraen las URLs que se encuentran en su interior, y agregan estas URLs a una cola para ser rastreadas. Una vez hecho esto, el rastreador recoge las URLs de la cola en cualquier orden, y repite el proceso. Cada página que es escaneada es dada a un *cliente* que guarda las páginas, crea un índice para las páginas o guarda y analiza los contenidos de las páginas.

Los rastreadores son utilizados especialmente por los motores de búsqueda. A lo largo de estos años, lo que siempre han intentado e intentan, es proporcionar la información más correcta y adecuada a la búsqueda que realiza el usuario por un término concreto. Para conseguir esto, cada motor de búsqueda ha realizado sus propios estudios para la aplicación de diferentes algoritmos de recuperación de información, de indexación de páginas, de lecturas de contenido, etc.

Debido al gran número de páginas web que hay, y sobre todo, al crecimiento constante diario de las mismas, los rastreadores intentan que sus rastreos sean cada vez más rapidos, que además, la información que recojan, sea comprimida lo máximo posible y las estructuras de datos sean elegidas con buen criterio, para que el espacio sea utilizado de forma correcta y eficiente.

En los documentos que se proponen de lectura para poder realizar este resumen, se citan algunos de los rastreadores más importantes a lo largo de estos años, incluso se presentan diferentes comparativas de rendimiento, habiendo ejecutado varias pruebas con cada uno de ellos.

Algunos de los rastreadores que se citan son los siguientes:

- **Matthew Gray's Wanderer:** fue el primer rastreador creado en la primavera del 1993, coincidiendo con la primera versión del NCSA Mosaic.[1]
- **Google:** Se dice de él, que es escalable, que indexa de forma eficiente y que posee unas estructuras de datos optimizadas (según el artículo *The Anatomy of a Large-Scale Hypertextual Web Search Engine*, de *Sergey Brin* y *Lawrence Page*). Este rastreador se compone de un conjunto de servidores con una función concreta:

[1] NCSA Mosaic: Fue el segundo navegador gráfico disponible para visualizar páginas web y el primer navegador gráfico para Microsoft Windows.

- *URLServer* : envía listas de URLs para ser extraídos por los rastreadores.
- *StoreServer* : recibe las URLs extraídas, las comprime y las almacena en un repositorio.
- Cada página tiene asociado un ID que se llama docID y que es asignado cada vez que una nueva URL es analizada.
- La indexación es ejecutada por el *indexador* y el *sorter*. Además el indexador ejecuta una serie de funciones: lee el repositorio, descomprime los documentos y los analiza. Cada documento es convertido en un conjunto de palabras llamados *hits*. El indexador distribuye los hits en un conjunto de *barriles* creando un forward index parcialmente ordenado. Analiza los enlaces externos y las palabras de *anclaje* que hay en cada web, almacenando la información importante en el fichero de *anclaje*. Este fichero contiene información que determina dónde apunta cada enlace y el texto del enlace.
- *URLresolver* : lee el fichero de anclajes y convierte URLs relativas en absolutas y las introduce en docIDs. Se genera una base de datos con pares de docIDs y estos son usados para computar el *PageRank*.
- *Sorter* : Finalmente se encarga de ordenar los *barriles*.

- **The Internet Archive:** Es otro rastreador alternativo, que es comparado con el rastreador *Mercator* en el texto de *Allan Heydon* y *Marc Najork*. Usa múltiples máquinas para rastrear la web, a cada proceso de rastreo se le asigna hasta 64 sitios para rastrear y ningún sitio se le asigna a más de un rastreador.

- **Mercator** : Rastreador escalable, extensible y modular, realizado en *JAVA*[2]. Realizando una comparativa con los dos rastreadores anteriores, se demostró que era mucho más eficiente, ya que, consumía menos recursos, logro descargar más páginas por segundo, a mayor velocidad y con más probabilidad de acierto. Además, la creación de módulos hizo bastante atractiva su utilización, ya que, sin tener que reprogramar el núcleo del rastreador.

- **WebFountain** (Edwards y otros., 2001) es un rastreador distribuido, modular parecido a Mercator desarrollado en el lenguaje C++. Parte una máquina "controladora" que coordina con una serie de máquinas "hormiga". Después de descargar las páginas, se deduce para cada página una tasa de cambio, también llamada, índice de actualización, y se debe utilizar un método de programación no lineal para solucionar un sistema de ecuaciones para logar la máxima "frescura".

- **PolyBot** [Shkapenyuk y Suel, 2002] es un rastreador distribuido escrito en C++ y en Python, compuesto por un "encargado del crawling", uno o más "descargadores" y uno o más "Encargados de resolver DNS". Las URLs recogidas se añaden a una cola en disco, y se procesa más adelante para buscar URLs de forma no continua. Posee una función que determina si un dominio es de segundo o tercer nivel (Por ejemplo: www.indipro.es y www2.indipro.es.com son terceros dominios del nivel).

- **WebRACE** (Zeinalipour-Yazti y Dikaiakos, 2002) es un módulo de crawling implementado en Java, y usado como una parte de un sistema más genérico, eRACE. El sistema recibe peticiones de los usuarios para descargar páginas web, actuando el rastreador como un servidor inteligente. El sistema también maneja los pedidos "suscripciones" a las páginas web que deben ser supervisados: cuando las páginas cambian, deben ser descargadas por el crawler y el suscritor debe

[2] *Lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90.*

ser informado. La característica más excepcional de WebRACE es que, mientras que la mayoría de los crawlers comienzan con un sistema de la "semilla" URLs, WebRACE está recibiendo continuamente URLs nuevos para comenzar.

- **Ubicrawler** (Boldi y otros., 2004) es un rastreador distribuido escrito en Java, y no tiene ningún proceso central. Se compone de un número de "agentes idénticos"; y se calcula la función de la asignación usando el hashing consistente de los nombres de anfitrión. Hay superposición cero, significando que no se recolecta ninguna página dos veces, a menos que un agente de arrastre se estrelle (entonces, otro agente debe recolectar las páginas del agente que falla). El crawler se diseña para alcanzar escalabilidad alto y para ser tolerante a los fallos.

2. Problemas que intenta resolver un crawler.

El objetivo principal que tienen todos los rastreadores, es el de seleccionar las URLs con la información más exacta sobre la consulta que realiza el usuario. Pero para conseguir esto, además de encontrarse con una serie de desafíos técnicos, se encuentra con el problema de seleccionar las páginas más importantes y ordenarlas en función de la información que contienen.

Los crawlers intentan conseguire que las páginas que tienen una mayor relevancia sean visitadas con anterioridad de las que teienen poca relevancia. Para ello, debe conocer un valor por el cual sean las páginas *rankeadas* y surgieron diferentes tipos de Ranking:

- *Similitud a la consulta Q* : Siendo P una web y Q una consulta, se convierten en vectores y se comparan su similitud en las palabras que contienen cada una.
- *Backlink Count* : Analizando el número de enlaces que existen a la web P. A mayor número de enlaces, mejor posición.
- *PageRank* : Algoritmo que calcula el ranking de una web, teniendo en cuenta diferentes parámetros, como por ejemplo, el número de enlaces que hay hacia la web, número de enlaces salientes que hay y también la información que contiene la web, el contenido, entre otros parámetros.
- *Forward Link Count* : Número de enlaces que emanan de P.
- *Location Metric* : Recalca la importancia de la localización de la web a la hora de realizar un rastreo. Para alguien que se encuentra en España, le interesarán webs que estén en España, antes que encontrar webs que sean Rusas, por ejemplo.

Pero además de la importancia de la métrica, para conseguir el objetivo que se marcan los rastreadores, existen diferentes métodos de rastreo. Ya que, no es posible conocer el valor real de ranking de cada web, pero sí un valor aproximado, existen 3 modelos de rastreo que intentan lograr rastrear las webs con más relevancia antes que las que no lo tienen:

- *Crawl & Stop* : Un rastreador C comienza en una página P_0 y para después de visitar K páginas. En este punto, el rastreador perfecto debería haber visitado las páginas $R_1, ..., R_k$ donde R_1 es la página con la información más importante, R_2 es la siguiente más importante, y así sucesivamente. Las páginas que van desde R_1hasta $R_k$se llaman *hot pages*.

- *Crawl & Stop with Threshold:* Un Asumimos de nuevo que el rastreador visita K páginas. Sin embargo, ahora nosotros damos un punto importante G y cualquier página con *I(P)*[3] $>= G$ es considerada como *hot*. Tomamos como valor H que representa el número de *hot pages* totales que hay. $P_{ST}(C)$ es el porcentaje de páginas H que han sido encontradas cuando el rastreador se para. Si $K < H$, entonces el rastreador ideal tendra un rendimiento de *(K·100) / H. Si* $K >= H$, entonces el rastreador ideal tendra un 100 % de rendimiento. Un rastreador aleatorio que revisita páginas tendra *(H/T)* · *H hot pages* cuando este pare. Asi, el rendimiento es de *(K·100)/T*. Sólo si el rastreador aleatorio visita T páginas, tendra un rendimiento del 100 %.

- *Limited Buffer Crawl* : En este modelo consideramos el impacto del almacenamiento limitado en el proceso de rastreo. Asumimos que el rastreador solo puede guardar B páginas en su buffer. Por ello, después de que el buffer se llene, el rastreador debe decidir qué páginas debe borrar para insertar otras nuevas. El rastreador ideal expulsaria las que tienen menos *I(P),* pero como tenemos un rastreador aleatorio, habrá problemas.

3. Problemas con los que se encuentra un crawler.

Los crawlers ejecutan distintas funciones para realizar el rastreo de las páginas web. Mientras realizan este rastreo, se encuentran con una serie de problemas de distintos tipos, como por ejemplo la capacidad de almacenaje que poseen, el ancho de banda (tanto suyo, como de la web rastreada) y otro tipo de problemas que se pueden enumerar del siguiente modo:

- *Alias de URL* : Los alias de las URLs pueden hacer que dos páginas posean URLs distintas, pero en realidad apunten a la misma. Existen cuatro casos fundamentalmente :

  - *Alias en el dominio o nombre del servidor* : Es muy frecuente que tengamos un servidor con una IP, en la cual alojemos distintas páginas web, de manera que, si hacemos un ping a esos alias, todos apunten a la misma IP. Por ejemplo: tengo un servidor contratado para dar servicio a mis clientes, y en un servidor, tengo varios espacios web alojados con sus dominios. El servidor se encuentra en la ip 91.121.164.42 , sin embargo, a esa misma IP apuntan dominios como por ejemplo *www.indipro.es* , *www.indiproweb.com*, *www.indiproweb.es* que apuntan a la misma IP y al mismo directorio. Para no ser penalizado por los buscadores, fue necesario crear una redirección 301, además de configurar correctamente los servidores DNS de cada dominio.
  - *Omisión de número de puerto* : Si no especificamos ningún puerto en la configuración, el navegador y el rastreador, van a interpretar que tiene el puerto por defecto del protocolo en el que nos encontremos, si es HTTP, tomará el 80, si es FTP, tomara el 21.
  - *Rutas alternativas en el mismo servidor* : puede ser que accedamos al mismo fichero escribiendo *www.indipro.es/index.php* que poniendo *www.indipro.es* , realmente muestra el mismo índice con dos URLs distintas.
  - *Réplicas en diferentes servidores* : Puede ser que un documento tenga mucha demanda y se ofrezcan distintos *mirrors* para poder ser descargado. En este caso estamos hablando de contenido duplicado.

[3] *Se utiliza la escritura I(P) para representar el número de enlaces que existen hacia la web P.*

- No obstante, los 2 últimos problemas, se solucionan realizando un test de contenido que verificaría si ese documento ya ha sido descargado anteriormente o no.

- *IDs de sesión en las URLs* : En muchas ocasiones, es necesaria la creación de sesiones para poder crear páginas de forma dinámica, como por ejemplo en los blogs, a la hora de crear páginas nuevas cuando tenemos una cantidad grande de noticias. Esto provoca en ocasiones, que con distintas sesiones, también acabemos accediendo al mismo documento. Para poder prevenir esto, por ejemplo, el rastreador Mercator que mencioné en el apartado anterior, tiene un sistema de *fingerprint* para detectar esto y poder evitar problemas.

- *Crawler Traps* : Esto es, una URL o conjunto de URLs que causan que el rastreador esté rastreando indefinidamente. Algunos *crawler traps* no son intencionados. Por ejemplo, un enlace simbólico en un sistema de ficheros puede crear un ciclo. Pero también los hay intencionados : las personas escriben aplicaciones en CGI que generan dinámicamente webs de forma infinita. Todos los spammers, también estan considerados en este tipo de problemas.

4. Etapas del crawling.

Los rastreadores, para poder conseguir su objetivo, deben pasar por diferentes etapas y funciones. Para realizar un rastreo, una función debe encargarse de encontrar las páginas web, otra debe encargarse de resolver los servidores de nombres para poder acceder a la información, etc. Dependiendo del tipo de rastreador, estas etapas pueden variar un poco.

Los rastreadores deben interactuar con cientos de millones de páginas, por ejemplo, ***Google***, tiene un sistema rapido de rastreo distribuido. Un único servidor URLServer sirve listas de URLs a un número de rastreadores. Ambos, el URLServer y los rastreadores son implementados en *Python*[4]. Cada rastreador tiene aproximadamente 300 conexiones abiertas al mismo tiempo. Esto es necesario para recuperar páginas de una forma rapida. A la velocidad máxima, el sistema puede rastrear unas 100 páginas web por segundo usando cuatro rastreadores. Esta cantidad aproximadamente representa una recuperación de datos a una velocidad de 600K por segundo. La mayor tensión de ejecución es la búsqueda de DNS. Cada rastreador mantiene su propia caché de DNS asi no es necesario hacer una busqueda de DNS antes de rastrear cada documento. Cada una de los cientos de conexiones pueden estar en un número de diferentes estados: buscando DNS, conectando a un servidor, enviando una solicitud y reciviendo una respuesta. Estos factores hacen de un rastreador, un componente complejo del sistema. Usan un sistema de *I/O (In / Out , entrada y salida)* asíncrono para gestionar eventos y un número de colas para mover páginas de un estado a otro.

El rastreador ***WebBase*** ejecuta muchos procesos al mismo tiempo para realizar un rastreo. Estos procesos reciben una lista de URLs para ser descargadas y simplemente devuelven el contenido completo del HTML o algunos errores que han encontrado intentando descargar las páginas. Los procesos de rastreo abren cientos de conexiones al mismo tiempo, a una velocidad de unas 25 páginas por segundo por cada proceso. Existen servidores que tienen una velocidad bastante lenta y permiten descargar poca información rapidamente, así se usan dos tipos de balanceo de carga en el sistema de WebBase. Primero, se dividen todas las URLs que van a ser rastreadas en 500 colas basadas en un *hash* de su nombre de servidor. Con esto se consigue que todas las URLs que pertenecen a un mismo servidor se encuentren en una misma cola. Los rastreadores leen una URL de la cola, la mueven a otra nueva cola

[4]Python es un lenguaje de programación de alto nivel cuya filosofía hace hincapié en una sintaxis muy limpia y que favorezca un código legible.

y asi se aseguran de leer las URLs de una en una hasta 500. Para servidores que tienen un retardo mayor, sólo se les permite una conexión al mismo tiempo con el servidor web. Los webmasters tienen la capacidad de restringir el acceso de estos rastreadores si no quieren ser rastreados.

Otro rastreador en el que podemos fijarnos mejor para distinguir las distintas etapas que hay en un crawling es el Mercator. El Mercator tiene múltiples hilos de trabajo cuando produce un rastreo. Cada hilo se repite tantas veces necesite para poder realizar el trabajo.

1. En primer lugar, se borra una URL absoluta del *URLFrontier* para descargas. Una URL absoluta comienza con un *esquema* (por ejemplo, "http"), que identifica el protocolo de red que debe ser utilizado para descargarla. Concretamente en este rastreador (Mercator) estos protocolos de red son implementados en los *módulos de protocolo*. Los módulos de protocolo para poder ser usados en un rastreo son especificados en el fichero de configuración y son dinámicamente cargados al comenzar con el rastreo. La configuración por defecto incluye protocolos HTTP, FTP y Gopher.

2. Después de haber seleccionado el protocolo correcto para la descarga, el siguiente paso es ejecutar el módulo de *ejecución* para descargar el documento de internet en un hilo *RewindInputStream* (también llamado RIS).

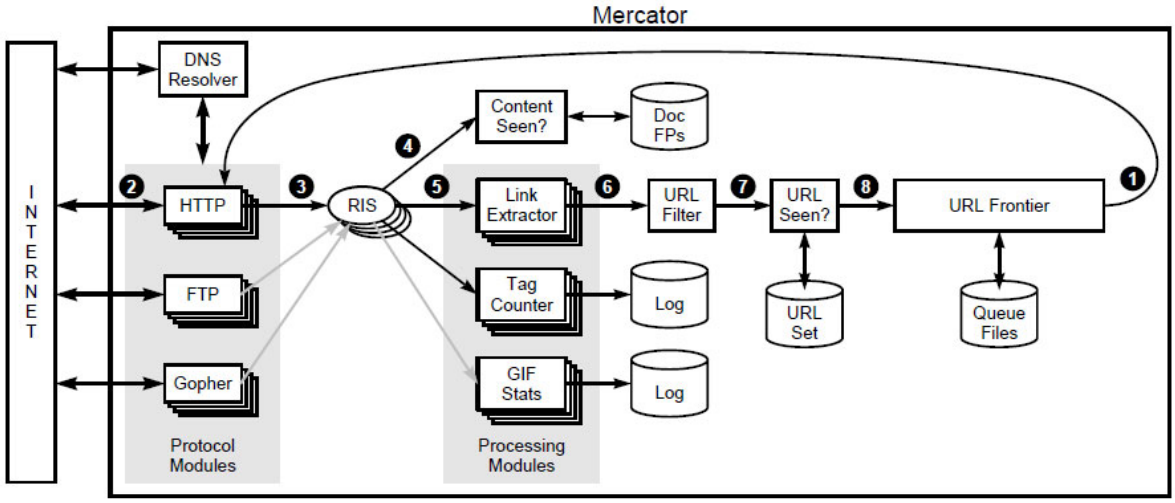
3. Un RIS es una extracción de E/S que es inicializada por una entrada arbitraria de flujo y que subsecuentemente permite que los contendos puedan ser releídos muchas veces. Una vez el documento ha sido escrito en el RIS, el hilo que trabaja, ejecutá el *test de contenido-visto* para determinar si este documento ha sido visto anteriormente.

4. Si es así, el documento no es procesado otra vez y el hilo borra la URL del *URL Frontier*. Todos los documentos descargados, tienen asociados un MIME type. Además de asociar esquemas con módulos de protocolo, el fichero de configuración de *Mercator*, también asocia los *MIME types* con uno o más *módulos de procesamiento*. Un módulo de procesamiento es una abstracción para procesadr documentos descargados, por ejemplo, extraynedo enlaces de páginas en HTML, contando etiquetas encontradas las páginas HTML o coleccionando estadísticas sobre imágenes GIF.

5. Después de descargar los ficheros y asociarlos a un tipo MIME, se selecciona el módulo que debe procesarlo. Por ejemplo, el *Extractor de Enlaces* y el *Contador de Etiquetas* son utilizados para los documentos de tipo *text/html,* y el módulo de *Estadisticas GIF* para los documentos de tipo *images/gif*.

6. Cada enlace es convertido en una URL Absoluta y pasada por un filtro *URL Filter* para determinar si debe ser descargada.

7. Si la URL es aceptada, pasa por el *módulo de prueba de contenido-visto*, que comprueba si la URL ha sido descargada anteriormente, y si la URL no es nueva, se introduce en el *URL Frontier*.



Por ejemplo, otro rastreador llamado WebEater, utiliza el siguiente algoritmo de crawling:

En esta métrica se mide la relevancia de cada página con respecto a un tema o búsqueda que el usuario tenga en mente. Para lo que viene a continuación se considera clave una página si contiene la palabra ordenador en su título. Se guardan dos colas y se visitan primero aquellas que hayan sido añadidas a la Cola Clave

Encolar(Cola_URL, URL_Inicio);
Mientras (No(Vacia (Cola_Clave)) Y No (Vacia(Cola_URL))){
url=Desencolar2(Cola_Clave, Cola_URL);
pagina=minarpagina(url); Encolar(paginas_recogidas,(url,pagina));
lista_url=extraer_urls(pagina);
para cada u en lista_url Encolar(enlaces,(url,u));
Si [u no en cola_url]Y[(u,-) no en paginas_recogidas]Y[u no en Cola_Clave]
Si [u contiene ordenador en vínculo o URL] Encolar(Cola_Clave, u);
Si no Encolar(Cola_URL,u);
Reordenar_Cola(Cola_URL);
Reordenar_Cola(Cola_Clave);
}
Descripción de las funciones
Desencolar2(Cola1,Cola2) : Si (no Vacia(Cola1)) desencolar(Cola1);
Si no desencolar(Cola2);

El proceso finaliza cuando termina de ejecutarse la extracción de los archivos de contenido del directorio. Otros crawlers de código abierto son Heritrix, WebSphinx, JSpider, Java WebCrawler, WebLech, Arachnid, JoBo, o WebHarvest

5. Qué otras áreas de investigación estan relacionadas con el Crawling.

El Crawling esta relacionado con otra serie de áreas de investigación cuya temática principal esta unida a el funcionamiento de los motores de búsqueda principalmente.

Las 3 áreas principales con las que esta relacionado el Crawling son las siguientes:

- Mineria de Datos
- Motores de búsqueda
- Recuperación de información

No obstante, basándonos en toda la información que he recopilado sobre los crawlers, teniendo en cuenta las distintas tareas que se deben realizar, también se puede decir que esta relacionado con las areas de investigación de las siguientes herramientas:

- Servidores de nombres DNS: Domain Name Server, debemos tener unos servidores DNS que funcionen a la perfección, para poder recuperar las páginas web de forma rápida y sin errores.
- Análisis de los diferentes protocolos de red: los protocos más utilizados de descarga y visualización de páginas web son HTTP y FTP, entre otros.
- Extractores de enlaces de documentos HTML.
- Indexadores web.

Después de haber leído todos los artículos propuestos en el enunciado del ejercicio, junto con otras webs que he encontrado en mi búsqueda particular, en mi opinión, los que mejor explican el funcionamiento copleto de un crawler son Allan Heydon y Marc Najork, componentes del equipo de *Compaq Systems Research Center*. Ya que, para explicar su rastreador particular creado en JAVA, han tenido que desarrollar la arquitectura que posee su crawler, detallando una explicación de las funciones de cada uno de los componentes por los que está compuesto, además de explicar posibles errores y trampas con las que se pueden encontrar estos rastreadores.

Además, utilizan muchas comparaciones con otros rastreadores existentes, como el Googlebot y el The Internet Archive bot, detallando mucho el hardware utilizado y los resultados obtenidos, pudiendo seguir un historial de funcionamiento de cada uno.

6. En qué conferencias internacionales se aborda el crawling.

Algunas de las conferencias internacionales que abordan el tema del crawling, son las siguientes:

- Conferences on World Wide Web.
- ACM International Conference on Management of Data (SIGMOD)
- International Conference on Very Large Databases
- IEEE International Conference on Data Engineering
- European Conference on Research and Advanced Technology for Digital Libraries

- Joint Conference on Digital Libraries (JCDL)
- International Conference on Visual Information Systems (Visual99)
- Latin American Web conference
- International Conference on Machine Learning (ICML97)
- International Conference on Autonomous Agents (Agents '98)
- International Conference on Web Information Systems and Technologies

Referencias

[1] Kobayashi, M. and Takeda, K. (2000). "Information retrieval on the web". ACM Computing Surveys (ACM Press) 32 (2): 144–173.

[2] Sergey Brin amf Lawrence Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. Computer Networks and ISDN Systems, vol. 30, 1998.

[3] Junghoo Cho, Hector Garcia-Molina, Lawrence Page. Efficient Crawling Through URL Ordering. Computer Networks and ISDN Systems, vol. 30, 1998.

[4] Allan Heydon and Marc Najork. Mercator: A Scalable, Extensible Web Crawler. In Proceedings of World Wide Web, 1999, pages 219-229.

[5] Brian Pinkerton. Finding what people want: Experiences with the WebCrawler. In Proc. 1st International World Wide Web Conference, 1994.

[6] Castillo, Carlos (2004). Effective Web Crawling. (Ph.D. thesis). University of Chile. Retrieved 2010-08-03.

[7] Gulli, A.; Signorini, A. (2005). "The indexable web is more than 11.5 billion pages". Special interest tracks and posters of the 14th international conference on World Wide Web. ACM Press.. pp. 902–903.

[8] Baeza-Yates, R., Castillo, C., Marin, M. and Rodriguez, A. (2005). Crawling a Country: Better Strategies than Breadth-First for Web Page Ordering. In Proceedings of the Industrial and Practical Experience track of the 14th conference on World Wide Web, pages 864–872, Chiba, Japan. ACM Press.

[9] Menczer, F. (1997). ARACHNID: Adaptive Retrieval Agents Choosing Heuristic Neighborhoods for Information Discovery. In D. Fisher, ed., Machine Learning: Proceedings of the 14th International Conference (ICML97). Morgan Kaufmann

[10] Diligenti, M., Coetzee, F., Lawrence, S., Giles, C. L., and Gori, M. (2000). Focused crawling using context graphs. In Proceedings of 26th International Conference on Very Large Databases (VLDB), pages 527-534, Cairo, Egypt.

[11] Pant, Gautam; Srinivasan, Padmini; Menczer, Filippo (2004). "Crawling the Web". In Levene, Mark; Poulovassilis, Alexandra. Web Dynamics: Adapting to Change in Content, Size, Topology and Use. Springer. pp. 153–178.

[12] Cho, Junghoo; Hector Garcia-Molina (2003). "Estimating frequency of change". ACM Trans. Interet Technol. 3 (3): 256–290.

[13] Baeza-Yates, R. and Castillo, C. (2002). Balancing volume, quality and freshness in Web crawling. In Soft Computing Systems – Design, Management and Applications, pages 565–572, Santiago, Chile. IOS Press Amsterdam.

[14] Risvik, K. M. and Michelsen, R. (2002). Search Engines and Web Dynamics. Computer Networks, vol. 39, pp. 289–302, June 2002.

[15] Zeinalipour-Yazti, D. and Dikaiakos, M. D. (2002). Design and implementation of a distributed crawler and filtering processor. In Proceedings of the Fifth Next Generation Information Technologies and Systems (NGITS), volume 2382 of Lecture Notes in Computer Science, pages 58–74, Caesarea, Israel. Springer.

[16] Dill, S., Kumar, R., Mccurley, K. S., Rajagopalan, S., Sivakumar, D., and Tomkins, A. (2002). Self-similarity in the web. ACM Trans. Inter. Tech., 2(3):205–223.

[17] Abiteboul, Serge; Mihai Preda, Gregory Cobena (2003). "Adaptive on-line page importance computation". Proceedings of the 12th international conference on World Wide Web. Budapest, Hungary: ACM. pp. 280–290. doi:10.1145/775152.775192. ISBN 1-58113-680-3.

[18] Jon Kleinberg, Authoritative Sources in a Hyperlinked Environment, Proc. ACM-SIAM Symposium on Discrete Algorithms, 1998.

[19] Massimo Marchiori. The Quest for Correct Information on the Web: Hyper Search Engines. The Sixth International WWW Conference (WWW 97). Santa Clara, USA, April 7-11, 1997.

[20] Marc Najork and Janet L. Wiener. Breadth-first crawling yields high-quality pages. In Proceedings of the Tenth Conference on World Wide Web, pages 114–118, Hong Kong, May 2001. Elsevier Science.

[21] Web de Princeton: http://www.cs.princeton.edu/courses/archive/spring10/cos435/Notes/web_crawling_topost.pdf.

[22] Archive.org : http://crawler.archive.org/faq.html

[23] CS.CMU.EDU: http://www.cs.cmu.edu/~rcm/websphinx/

[24] Parc.com : http://www2.parc.com/spl/projects/modrobots/chain/polybot/index.html

[25] Crawling the web: http://dollar.biz.uiowa.edu/~pant/Papers/crawling.pdf

[26] Escribiendo un rastreador web en JAVA : http://java.sun.com/developer/technicalArticles/ThirdParty/WebCrawler/