SACARINO (Sonda AutomátiCA para la Recuperación de INformaciÓn en la Web): un robot para recorrer y procesar la Web (1)

José L. Alonso Berrocal
Carlos García Figuerola
Ángel F. Zazo Rodríguez
Universidad de Salamanca (España)

Resumen

El objetivo principal de este trabajo es mostrar los mecanismos necesarios para poder recorrer y procesar la Web, indicando cómo manejarse con el grafo web, qué herramientas tenemos disponibles y qué resultados podemos esperar. Una vez expuestos los conceptos teóricos, presentaremos la herramienta desarrollada por nuestro grupo de investigación y que hemos denominado *SACARINO* (Sonda AutomátiCA para la Recuperación de Información en la Web). Veremos las soluciones aplicadas en SACARINO para poder recoger información de la web y mostraremos la aplicación EloisaBot Tools para realizar los cálculos necesarios y obtener los diferentes índices. Mostraremos, como conclusión, que se trata de una herramienta muy eficaz en la caracterización de la Web que viene siendo utilizada de forma muy habitual por nuestro grupo de investigación en sus trabajos.

Palabras clave: Recuperación de información. Cibermetría. SACARINO.

Abstract

The main aim of this work is to indicate the necessary mechanisms to be able to crawl and process the Web, showing how to handle with the graph Web, what tools are available and what results we can expect. Once the theoretical concepts are shown, we will present a tool developed by our investigation group, that we have named *SACARINO* (Sonda AutomátiCA para la Recuperación de Información en la Web). We will see what solutions can be applied in SACARINO in order to collect information from the Web and will show the application EloisaBot Tools to do the necessary calculations and obtain the different indexes. As a conclusion, we will show that it is a very effective tool in the description of the Web, that is being used by our investigation group in its research.

Keywords: Information Retrieval. Cybermetrics. SACARINO.

1. Introducción

La Web es una colección de billones de documentos escritos de tal forma que pueden ser citados usando hiperenlaces y conformando el denominado *hipertexto*. Estos documentos, o páginas web, tienen unos pocos cientos de caracteres escritos en infinidad de idiomas y que cubren esencialmente todas las materias del saber humano. Estas páginas web se encuentran instaladas en un servidor web y son servidas ante las peticiones del cliente empleando el protocolo HTTP y visionadas por los visores web. Para poder analizar esta enorme cantidad de páginas es necesario elaborar programas automáticos que permitan analizar los documentos hipertexto recorriendo toda la red a través de los hiperenlaces que los conectan.

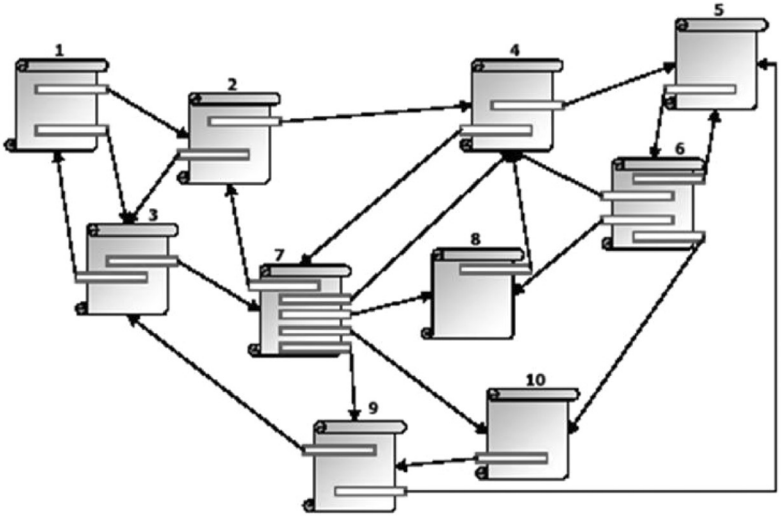

Figura 1. Esquema básico de la Web

La bibliografía existente sobre este particular es extensa y variada, destacando los trabajos de Thelwall (2001), Alonso *et al.* (2003), Chakrabarti (2003) y Castillo (2004), que dan una idea de los mecanismos necesarios para el trabajo con este tipo de herramientas.

Un *web crawler* es un programa de ordenador que es capaz de recuperar páginas de la Web, extrayendo los enlaces desde estas páginas y siguiéndolos. Este trabajo de recorrer todas las páginas web recibe el nombre genérico de *crawling* y los programas desarrollados para hacerlo pueden ser denominados *crawler*, *spider*, *wanderer*, *robot* o *bot*.

Hay varias formas de hacer este recorrido de la web, aunque básicamente existen tres:

1. Recorrido en anchura (*breadth-first*).
2. Recorrido en profundidad (*depth-first*).
3. El mejor posible (*best-first*).

A continuación mostramos un esquema de cómo serían los dos primeros recorridos y los resultados que se obtendrían.

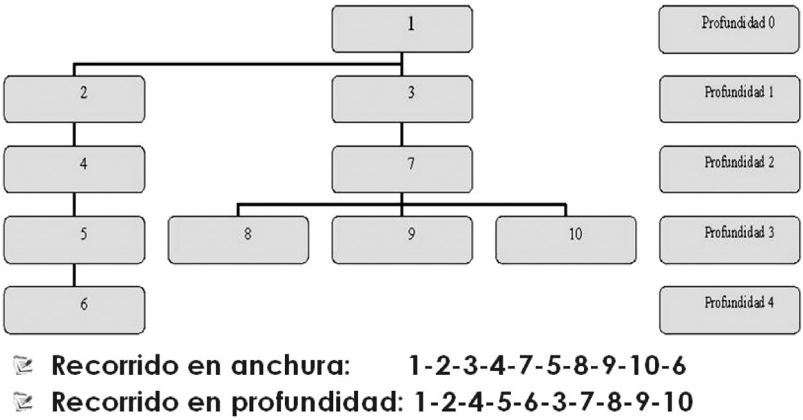

Figura 2. Resultados según el tipo de recorrido

Para el esquema de "el mejor posible", la decisión de cuáles son los enlaces a recorrer se toma en función de distintas técnicas, como por ejemplo la utilización del valor del PageRank, para decidir recorrer en primer lugar los que poseen un PageRank mayor y dejar para el final los que tengan uno menor.

El procedimiento básico de un robot consiste en suministrar una URL inicial o un conjunto de ellas, obtener la página web correspondiente y a continuación extraer todos los enlaces existentes en dicha página. Con los enlaces obtenidos es necesario realizar una serie de operaciones previas de normalización, entre las que podemos indicar las siguientes:

- Convertir URL a minúscula.
- Eliminar anclas.
- Adecuar el sistema de codificación.
- Emplear la heurística para la determinación de la página por defecto.
- Resolver las URL relativas.

A continuación será necesario comprobar las URL que se habían seguido previamente y, en caso de no haberlos recorrido, introducirlos en una cola de URL a seguir. Después, normalmente, almacenamos la información, bien en bases de datos o en estructuras de ficheros con codificación ASCII. Finalmente, obtenemos la URL del siguiente enlace a seguir y comienza de nuevo el proceso. En la figura 3 se muestra un esquema básico de este planteamiento.

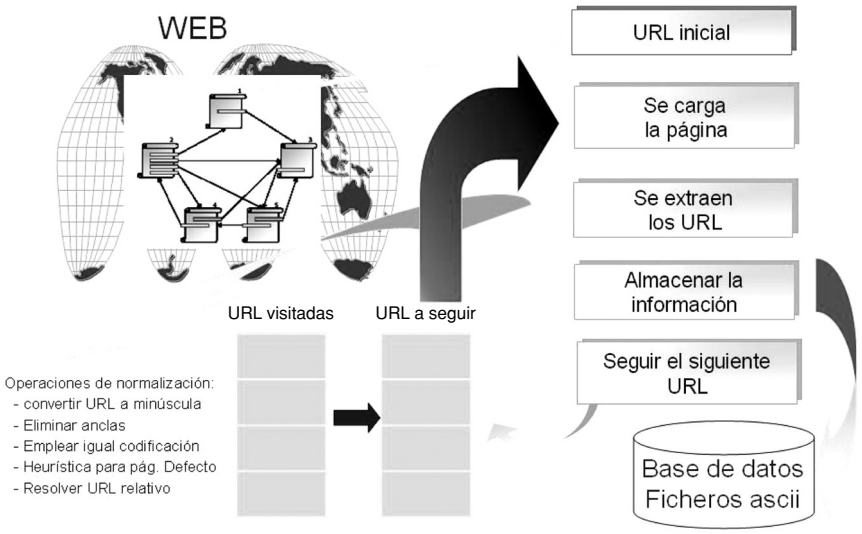

Figura 3. Proceso básico para recorrer la Web

Existen mecanismos más sofisticados, dos de los cuales presentamos dos de ellos en la figura 4, extraídos de http://dbpubs.stanford.edu:8091/~testbed/doc2/WebBase/webbase-pages.html y de Chakrabarti (2003), respectivamente.

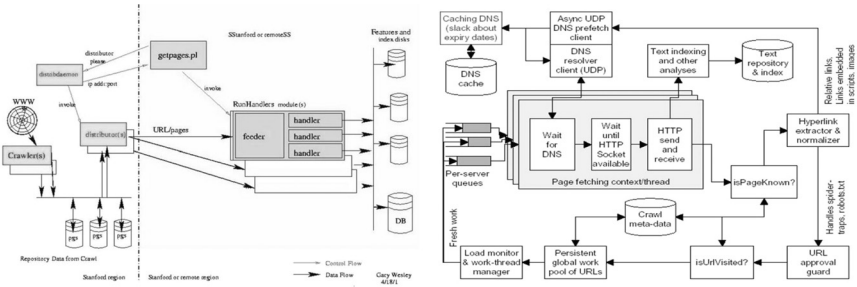

Figura 4. Procedimientos avanzados para recorrer la Web

La lista de bots es realmente muy amplia, pero podemos destacar los siguientes:

- WebBot. Disponible en la dirección http://www.w3.org/Robot/, se trata de un proyecto del World Wide Web Consortium (W3C).
- Harvest-NG. Disponible en la dirección http://webharvest.sourceforge.net/ng/, se trata de un conjunto de utilidades para construir *web crawlers* y está escrito en lenguaje PERL.
- Webvac Spider. Disponible en la dirección http://dbpubs.stanford.edu:8091/~testbed/doc2/WebBase/webbase-pages.html, es un proyecto de la Universidad de Stanford.
- SocSciBot 3 y SocSciBotTools. Disponible en la dirección http://webharvest.sourceforge.net/ng/http://socscibot.wlv.ac.uk/, es una opción interesante con utilidades adicionales.
- WIRE crawler. Disponible en la dirección http://www.cwr.cl/projects/WIRE/, es el desarrollo del Centro de Investigación de la Web (CWR) dirigido por Ricardo Baeza-Yates.
- SacarinoBot y EloisaBot Tools. Se trata de nuestro desarrollo y es el objeto de este artículo.

2. SACARINO

Nuestros primeros desarrollos de un bot para poder recorrer la web se remontan al año 1994, con la obtención del software denominado *sonda ciberdocumental*, que empleamos para los primeros cálculos de tipo cuantitativo (Alonso, 1996). Esta primera versión fue experimentando constantes mejoras y adaptaciones durante varios años. En el año 2003 decidimos reprogramar por completo el bot en un nuevo lenguaje de programación, optimizando las estructuras de datos y dotando al programa de mayor potencia, rapidez y flexibilidad. Así mismo se aplicó toda la experiencia adquirida con la sonda ciberdocumental para mejorar en múltiples aspectos el bot inicial. Entonces fue cuando nació SACARINO, cuyo desarrollo se vio favorecido por la incorporación de María del Carmen Montejo Villa y Faustino Frechilla Daza, alumnos de Ingeniería en Informática de Sistemas, que con sus proyectos fin de carrera impulsaron de forma definitiva la evolución y transformación del bot.

En la figura 5 se muestra la pantalla principal del programa. El punto de partida es la URL inicial o las URL iniciales (a partir de una lista en un fichero) desde de las cuales se recorrerá la web. Podemos modificar el recorrido según tres aspectos:

- Host especificado. Se recorren todas las páginas del host de partida.
- Host y directorio especificados. Se recorren todas las páginas de un host, pero restringido al directorio indicado en la URL solamente.

- Todos los hosts. Se recorren todas las páginas de todos los hosts que aparezcan. Esta posibilidad se puede matizar desde la pestaña de opciones.

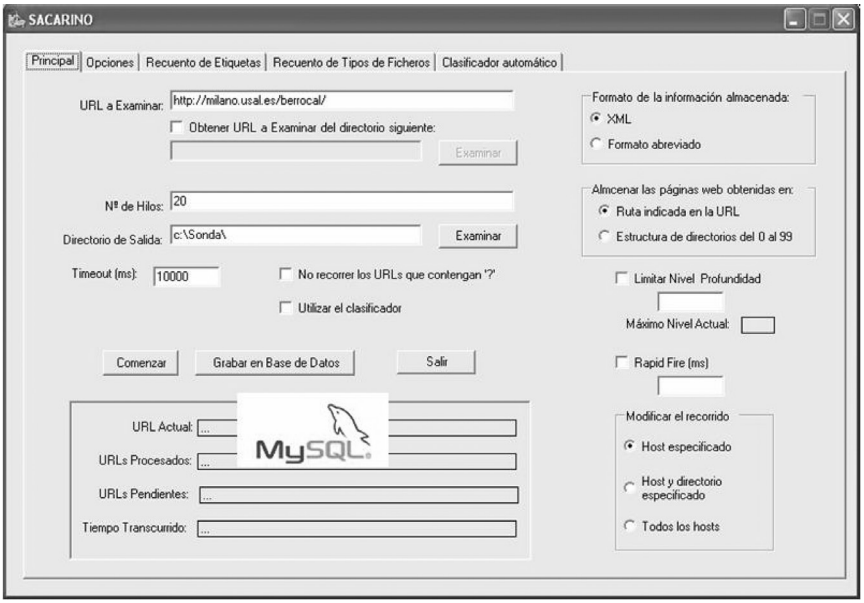

Figura 5. Pantalla principal de SACARINO

Podemos fijar el número de hilos que podemos ejecutar simultáneamente. Esta es una característica esencial en cualquier bot para optimizar y mejorar el proceso de recogida. Podemos especificar un directorio de salida para grabar toda la información correspondiente a la recogida:

- Estructura de enlaces. Se puede almacenar en formato XML o en un formato abreviado similar al propuesto por Thelwall (2004).
- Los ficheros HTML recorridos. Se almacenan las páginas manteniendo la misma estructura de directorios que tienen en la Web o en directorios numerados del 0 al 99 y subdirectorios con la misma estructura.

La información se puede almacenar en base de datos. En nuestro caso utilizamos MySQL, que ha demostrado ampliamente su potencia como sistema gestor de bases de datos. Podemos limitar la recogida eliminando todas las URL con?; limitando por nivel de profundidad, podemos ajustar un RapidFire para evitar saturaciones de los servidores. Una característica distintiva de nuestro bot es la posibilidad de realizar el recorrido de la Web empleando un clasificador automático.

Si se utiliza el clasificador es necesario entrenarlo, de forma que los documentos que cumplan las condiciones de la clasificación serán los que se sigan. Desde esta pantalla principal se indica si se desea emplear el clasificador en la recogida y posteriormente habrá que entrenar el clasificador.

En la figura 6 mostramos la pantalla de opciones, en la que podemos matizar algunas de las operaciones a realizar. En primer lugar, si en la pantalla principal habíamos marcado la opción de todos los hosts, podemos definir limitaciones para permitir o no hosts específicos. Esto nos permite un ajuste perfecto en lo que deseamos recorrer. Podemos definir las extensiones de los ficheros que vamos a considerar válidas a efectos de la recogida y podemos también indicar las URL que no se deben recoger de forma expresa. Podemos fijar estas URL a no seguir directamente en la pantalla de opciones o bien cargarlos desde un fichero de texto. Podemos emplear un sistema de comparación exacta o realizar un truncamiento a la derecha para permitir una mayor flexibilidad y limitar la recogida a las URL que empiezan por algo determinado.

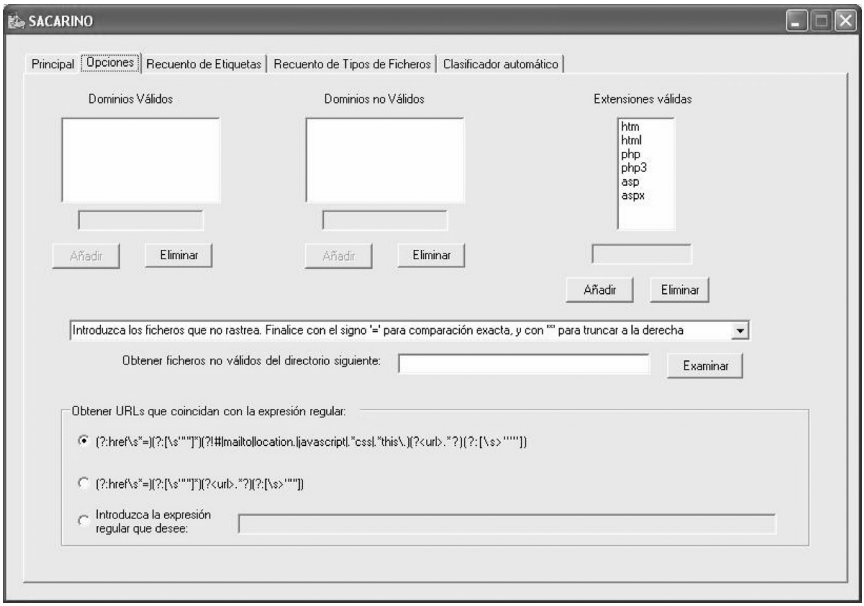

Figura 6. Pantalla de opciones

Finalmente, tenemos la posibilidad de escoger el tipo de expresión regular que vamos a emplear en el proceso de seguimiento de los enlaces. Tenemos dos opciones ya predefinidas y una opción a incluir por el usuario de forma libre.

En las figuras 7 y 8 podemos realizar un recuento de etiquetas o de tipos de ficheros, respectivamente. Es totalmente ajustable a las necesidades de tipo cuantitativo que podamos tener.

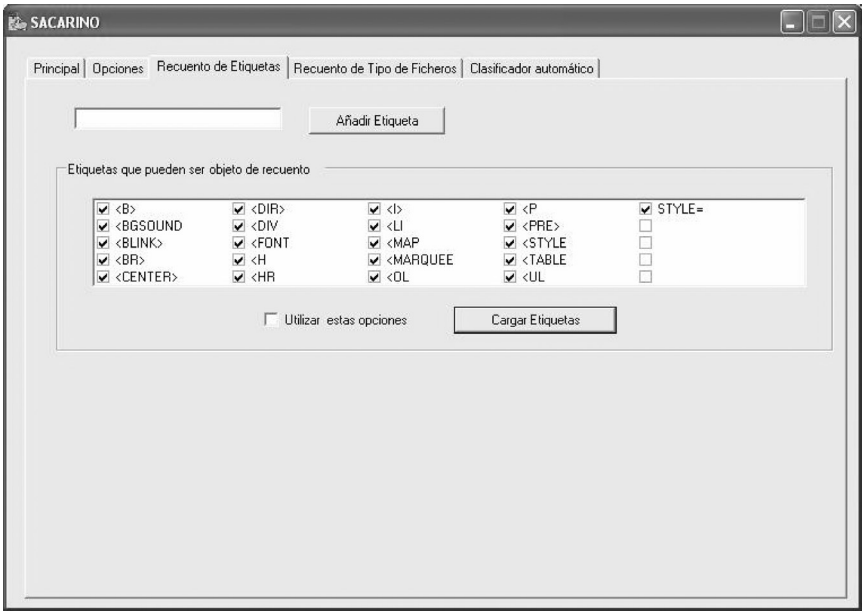

Figura 7. Recogida de etiquetas

En el caso de los ficheros se emplea una sintaxis propia para poder recoger de forma global un tipo de fichero para el que existan varias extensiones. Por ejemplo, HTML:2/HTM/HTML, significa que para un formato genérico HTML tenemos dos extensiones a contabilizar, la HTM y la HTML.

En la figura 9 podemos especificar los datos correspondientes a la utilización del clasificador en el proceso de recogida, si en la pantalla principal se hubiera indicado. Básicamente, tenemos que cargar el directorio donde se ha realizado el entrenamiento, y aparecen a continuación tipos de documentos entrenados. Elegimos la clase correcta y el umbral que deseamos para poderlo considerar como válido. Si no se hubiera realizado ningún entrenamiento, accedemos con el botón correspondiente para ir al clasificador.

En las figuras 10-13 mostramos las pantallas del clasificador automático. Este clasificador se basa en la conocida biblioteca Libbow desarrollada por Andrew McCallum (1996) (disponible en http://www.cs.cmu.edu/~mccallum/bow/). Se trata de una librería con licencia LGPL y que se ha recopilado para entornos Windows.

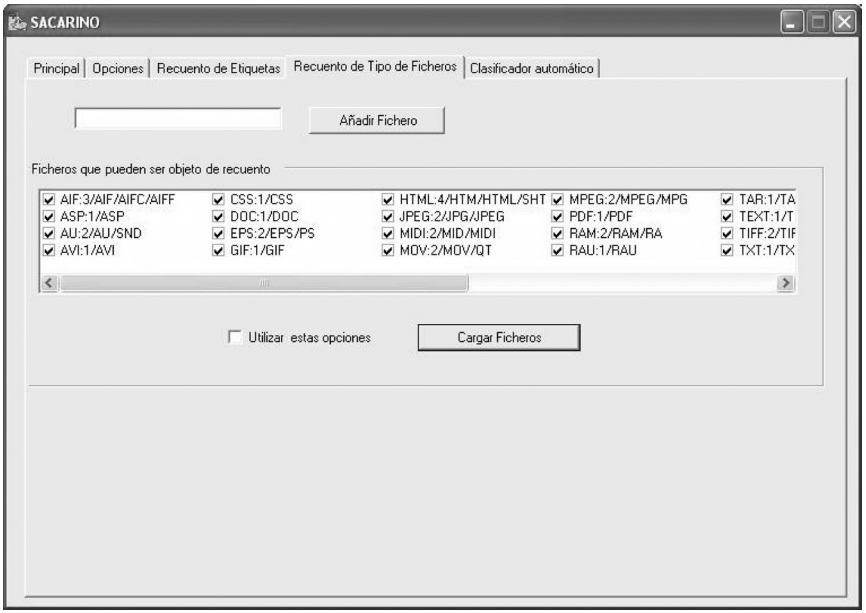

Figura 8. Recogida por tipos de ficheros

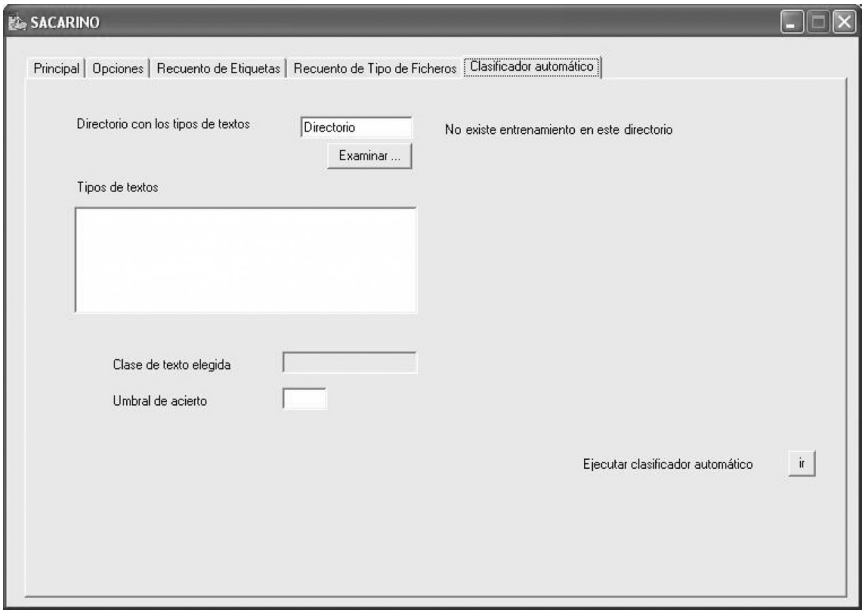

Figura 9. Seleccionar documentos clasificados

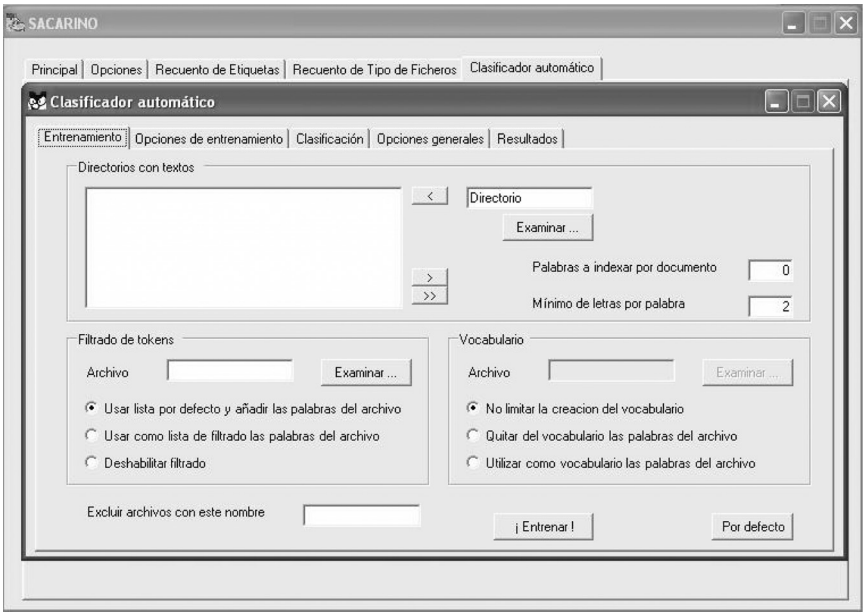

Figura 10. Clasificador automático. Entrenamiento del sistema

Las opciones del clasificador son muy variadas y ofrecen unos niveles de ajuste muy precisos. Los resultados, una vez completada una buena fase de entrenamiento, suelen ser muy buenos.

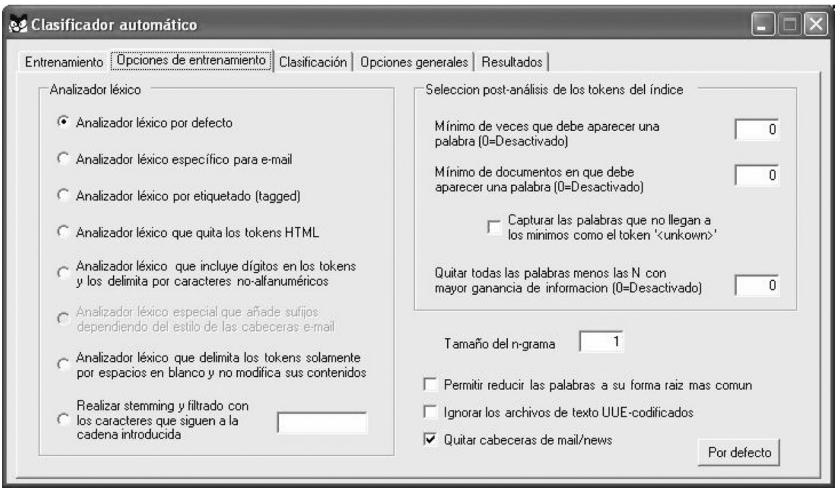

Figura 11. Clasificador automático. Opciones del entrenamiento

Convendría destacar de la figura 12 la cantidad de métodos disponibles para el cálculo de los pesos de los términos.

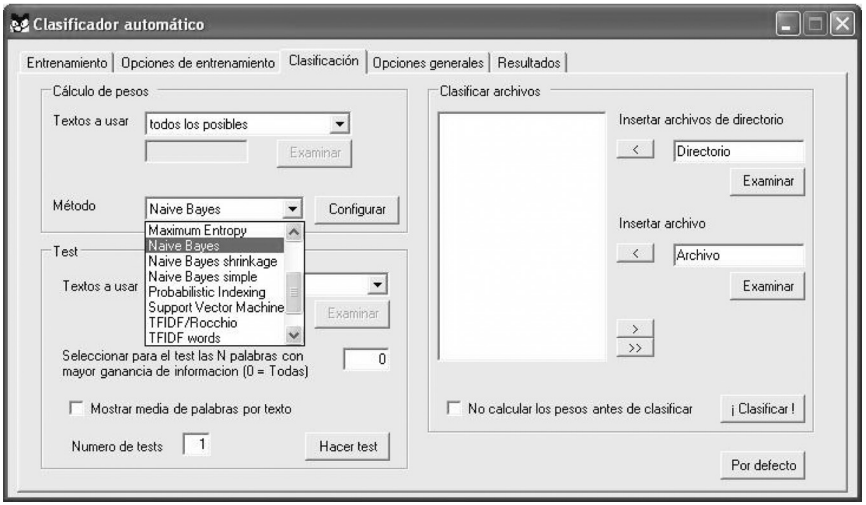

Figura 12. Clasificador automático. Opciones de clasificación

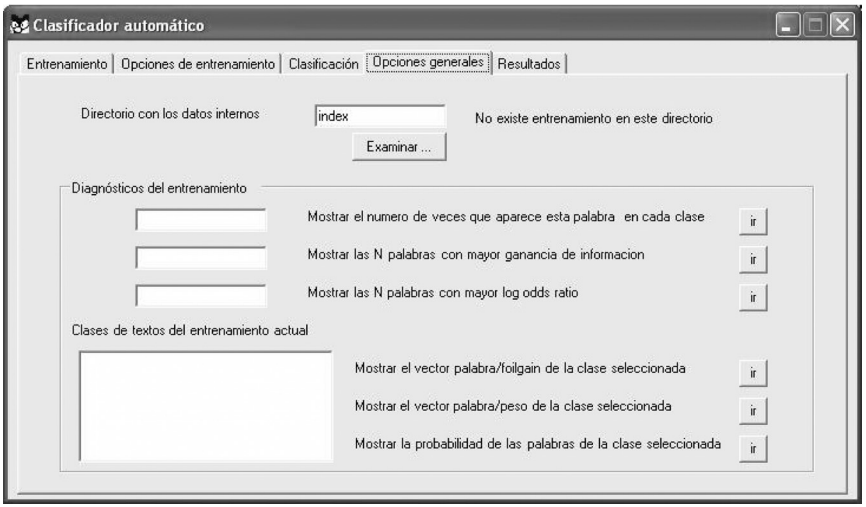

Figura 13. Clasificador automático. Opciones generales del clasificador

3. EloisaBot Tools

Una vez recogida la información con el bot, es necesario procesar toda esa información. Hemos elaborado un conjunto de herramientas, englobadas bajo el nombre *EloisaBot Tools*, que facilitan enormemente este procesamiento de datos. En la figura 14 tenemos la pantalla principal del programa que nos da acceso a las funciones de proceso de datos. En este momento nos encontramos en la fase de agrupar todos los programas elaborados y que teníamos repartidos en diferentes sistemas, para que desde un único entorno podamos acceder a ellos.

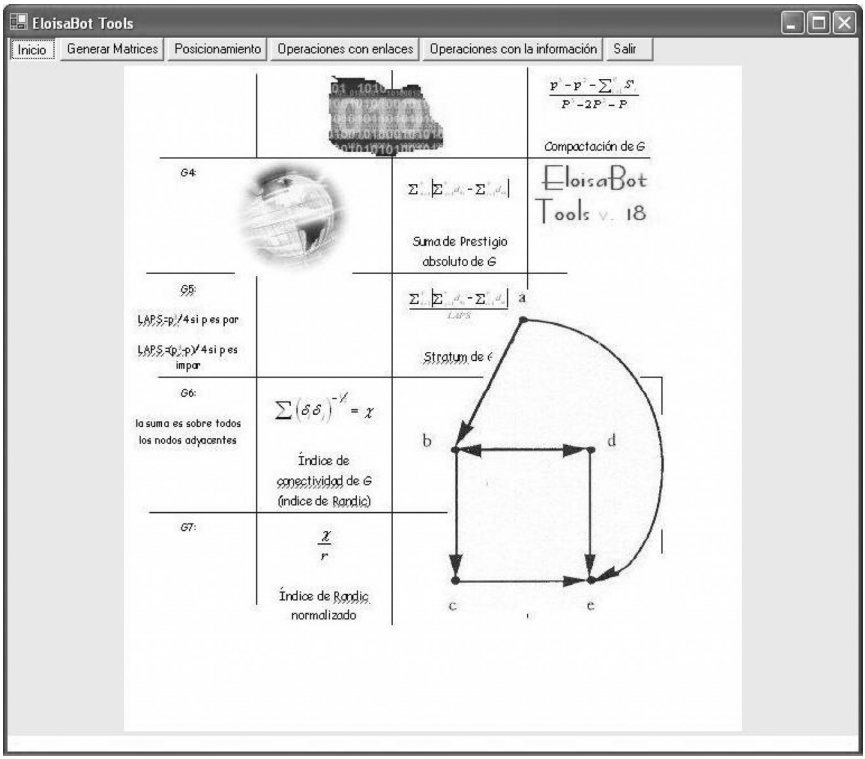

Figura 14. EloisaBot Tools

Básicamente, las funciones implementadas se desarrollaron en Alonso *et al.* (2004) y nos permiten

- Generar las matrices que conforman el grafo web. Se generan en varios formatos, permitiendo la compatibilidad con otros programas, como, por ejemplo, Pajek (Nooy *et al.*, 2005) (http://vlado.fmf.uni-lj.si/pub/networks/pajek/).

- Calcular diferentes algoritmos, como PageRank, HITS o SALSA.
- Diferentes cálculos con los enlaces, como factor de impacto web, visibilidad, densidad, índice de desarrollo hipertextual, índice de endogamia, diámetro, etcétera.
- Diferentes operaciones con la información, obteniendo datos cuantitativos sobre número de ficheros, etiquetas, tamaños, empleo de etiquetas META, etcétera.

Gran parte de las rutinas están elaboradas para Matlab, debido a la potencia de cálculo que nos ofrece, y estamos pensando en transformar todas las rutinas a Octave (http://www.octave.org/), que se encuentra bajo licencia GNU.

4. Resultados y trabajo futuro

Tanto el bot SACARINO como las herramientas EloisaBot están siendo utilizadas de forma intensa por nuestro grupo de investigación y han sido objeto de trabajo en múltiples proyectos y artículos. Sus resultados han sido plenamente satisfactorios en todas las ocasiones y se han mostrado como herramientas fiables de trabajo.

En la actualidad nos encontramos en fase de mejora de SACARINO, centrada sobre todo en el comportamiento del reparto del trabajo de los hilos, implementando técnicas de procesamiento en paralelo. Estamos finalizando la adaptación de todos los programas de tratamiento de datos y están casi todos englobados dentro de EloisaBot.

Seguramente la evolución no terminará nunca, pues es preciso adaptarse a las nuevas realidades. Nuestra intención es que estas herramientas puedan estar disponibles bajo licencia GNU para los usuarios interesados.

Notas

(1) Parte de las aportaciones de este trabajo se enmarcan dentro de los resultados obtenidos por el proyecto de investigación de la Junta de Castilla y León (Proyecto SA089/04) titulado *Nuevas técnicas de* ranking *en la recuperación de información en la Web*.

Referencias

Alonso Berrocal, J. L. (1996). Herramienta software para el análisis de la documentación web: rastreo de dominios, estudio de etiquetas, tipología de ficheros, evolución de los enlaces. Grado de Salamanca, Facultad de Traducción y Documentación. Universidad de Salamanca.

Alonso Berrocal, J. L.; Figuerola, C. G.; Zazo, Á. F.; Rodríguez, E. (2003). Agentes inteligentes: recuperación autónoma de información en la WEB. // Revista Española de Documentación Científica. 26:1 (2003) 11-20.

Alonso Berrocal, J. L.; Figuerola, C. G.; Zazo, Á. F. (2004). Cibermetría: nuevas técnicas de estudio aplicables al Web. Gijón: Trea, 2004.

Castillo, C. (2004). Effective Web Crawling. Tesis doctoral. Departament of Computer Science. University of Chile. URL: <http://www.chato.cl/534/article-63160.html>.

Chakrabarti, S. (2003). Mining the Web: discovering knowledge from hypertext data. San Francisco: Morgan Kaufmann, 2003.

McCallum, A. K. (1996). Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. URL: <http://www.cs.cmu.edu/~mccallum/bow>.

Nooy, W. D.; Mrvar, A.; Batagelj, V. (2005). Exploratory Social Network Analysis witj Pajek. Nueva York: Cambridge UP, 2005.

Thelwall, M. (2001). A web crawler design for data mining. // Journal of Information Science. 27:5 (2001) 319-325.

Thelwall, M. (2004). Link analysis: An information science approach. San Diego: Elsevier AP, 2004.