Arquitectura Optimizada para un Motor de Búsqueda Web Eficiente: Crawling de Contenido Útil y Actualizado

Víctor Manuel Prieto Álvarez

Departamento de Tecnoloxías da Información e as Comunicacións

UNIVERSIDADE DA CORUÑA

Directores:

Dr. Fidel Cacheda Seijo

Dr. Manuel Álvarez Díaz

2013

A mi padre y a mi hermano,
sendero y asfalto
en el camino a mis metas.

Agradecimientos

No me lo puedo creer, parece que esto se termina... Quiero volver a comenzar, y compartir con vosotros una de las frases que me ayudo a entender como debía ser mi trabajo diario para lograr realizar esta tesis. La frase la dijo Thomas Alva Edison: "No fracasé, sólo descubrí 999 maneras de como no hacer una bombilla". Esta frase para mi representa, el valor de un trabajo diario, de la constancia y la superación. Durante estos años, he descubierto 999 de no mejorar los sistemas de crawling, pero he descubierto una forma de hacerlo. El trabajo diario, unido a vuestro apoyo, ha dado el fruto de lo que es este trabajo hoy en día. Por ello, os quiero dedicar mis agradecimientos.

Gracias a mis codirectores, Fidel Cacheda y Manuel Álvarez. A Fidel, por su experiencia y su visión positiva, aportando la perspectiva adecuada en cada una de las investigaciones. A Manuel, por su ayuda diaria y continuas correcciones, han sido muchas las horas de charlas y pensamientos. Sin ellas, hubiera sido imposible.

Gracias a mis compañeros de laboratorio. Por apoyarme día a día, y en los momentos difíciles, por ayudarme a ver el camino más sencillo, y por supuesto por todos los momentos que hemos pasado juntos. Nunca los olvidaremos ;)

Gracias a mi familia y amigos, por aguantarme cuando se que soy inaguantable, por no hacerme caso cuando no hay que hacérmelo y por saberme distraer y alegrar.

Gracias a ti, por estar cuando tienes que estar y ser como eres.

Entre todos, me habéis ayudado a aprender lo más importante: En la vida siempre hay que crear y tener nuevos retos e ideas por las que luchar.

Resumen

La Web constituye el mayor repositorio de información jamás construido. Por este motivo se hace imprescindible la utilización de buscadores web que permitan localizar la información apropiada en cada momento.

Uno de los módulos de un buscador es el formado por los crawlers, programas software que aprovechan la estructura basada en hipervínculos de la Web, para recorrerla y crear un repositorio con los recursos web sobre el que poder realizar búsquedas. Pero el recorrido de la Web presenta numerosos desafíos para los crawlers, entre los que destacan: el tratamiento de la Web Oculta del lado cliente/servidor, la detección de páginas "basura" (Spam y Soft-404) o la actualización de contenidos.

Las técnicas existentes para la detección de Web Spam y páginas Soft-404 presentan multitud de deficiencias tanto a nivel de eficacia como de eficiencia. Además, no han sido diseñadas para su uso en sistemas de crawling.

Respecto al recrawling de la Web, los métodos existentes se centran en analizar la frecuencia de cambio de las páginas o el comportamiento de los buscadores para proponer políticas de recrawling. Estos estudios se basan en datos estadísticos que intentan aproximar el instante de modificación de las páginas.

Esta tesis presenta el diseño de una arquitectura de búsqueda web que plantea soluciones a las problemáticas asociadas a recursos que no deben de ser procesados: porque no son útiles (Web Spam o páginas Soft-404) o porque no han cambiado desde la última vez que se accedió a ellos. En primer lugar presenta dos estudios para caracterizar la Web. El primero de ellos analiza la Web Oculta y su tratamiento por parte de los crawlers, y el segundo analiza la evolución en la Web de la edad y la similitud de las páginas, para su uso en el recrawling de los contenidos, y de otras características que ayuden en la detección de Web Spam y páginas Soft-404.

Para la detección de páginas "basura", se proponen técnicas basadas en contenido, que permiten detectar Web Spam y páginas Soft-404 de forma más eficaz y eficiente que las presentes en la literatura. De este modo, el crawler no dedicará recursos a descargar, indexar y mostrar este tipo de páginas, mejorando la calidad de sus repositorios.

Para la actualización de contenidos, se ha propuesto un sistema que permite detectar en "tiempo real" modificaciones en páginas web. Nuevamente, se mejora el rendimiento del crawler debido a que, por una parte, no procesará páginas que no hayan cambiado, y por otra parte, las páginas del repositorio serán más actuales.

Abstract

The Web constitutes the biggest repository of information ever created. Due to this reason, it is essential to use web search engines to locate appropriate information at all times.

Crawlers comprise one of the modules of a search engine. These are software programs that leverage the hyperlink structure of the Web to traverse it and create a repository of web resources on which one can perform searches. However, the traversal of the Web presents several challenges for crawlers, among which are the treatment of the client/server side Hidden Web, "garbage" detection (Spam and Soft-404 web pages) or content update.

Existing techniques for detecting Web Spam and Soft-404 web pages have multitude of limitations in terms of effectiveness and efficiency. Furthermore, they were not originally designed to be used by crawling systems.

Regarding recrawling of the Web, existing methods focus on analyzing the frequency of change of the pages or the behavior of search engines to propose recrawling policies. These studies are based on statistics that attempt to approximate when a web page has been changed.

This thesis presents the design of a web search architecture that provides solutions to the problems associated with resources that should not be processed either because they are useless (Web Spam and Soft-404 pages) or because they have not changed since last time they were accessed. First, it presents two studies that characterize the Web. The first one analyzes the Hidden Web and how it is processed by crawlers. The second one analyzes the evolution of age and similarity of the pages on the Web and its use when recrawling web content. It also studies other features that help to detect Web Spam and Soft-404 pages.

Content-based techniques are proposed for "garbage" page detection. These techniques detect Web Spam and Soft-404 pages more effectively and efficiently than the ones described in existing literature. In this way, the crawler will not waste resources in downloading, indexing and showing this kind of web pages, therefore improving the quality of their repositories.

To update the content, we propose a system that can detect "real time" changes in web pages. Again, performance of the crawler is improved because, on the one hand, it will not process pages that have not changed and, on the other hand, the web pages of the repository will be more up-to-date.

Resumo

A Web constitúe o maior repositorio de información xamais construído. Por ese motivo faise imprescindible a utilización de buscadores web que permitan localizar a información apropiada en cada intre.

Un dos módulos dun buscador é o formado polos crawlers, programas software que aproveitan a estrutura baseada en hipervínculos da Web, para percorrela e crear un repositorio cos recursos web sobre os que poder facer búsquedas. Pero o percorrido da Web presenta numerosos desafíos para os crawlers, entre os que destacan: o tratamento da Web Oculta do lado cliente/servidor, a detección de páxinas "lixo" (Spam e Soft-404) ou a actualización de contidos.

As técnicas existentes para a detección de Web Spam e páxinas Soft-404 presentan multitude de deficiencias tanto a nivel de eficacia coma de eficiencia. Ademais, non se deseñaron para o seu uso en sistemas de crawling.

Respecto o recrawling da Web, os métodos existentes céntranse en analizar a frecuencia de cambio das páxinas ou o comportamento dos buscadores para propoñer políticas de recrawling. Estos estudos baséanse en datos estadísticos que intentan aproximar o instante da modificación das páxinas.

Esta tese presenta o deseño dunha arquitectura de busca web que plantexa solucións ás problemáticas asociadas a recursos que non deben ser procesados: porque non son útiles (Web Spam e páxinas Soft-404) ou porque non cambiaron dende a última vez que se accedeu a eles. En primeiro lugar presenta dous estudos para caracterizar a Web. O primeiro deles analiza a Web Oculta e o seu tratamento por parte dos crawlers, e o segundo analiza a evolución na Web da idade e a similitude das páxinas, para o seu uso no recrawling de contidos, e doutras características que axuden na detección de Web Spam e páxinas Soft-404.

Para a detección de páxinas "lixo", propóñense técnicas baseadas no contido, que permiten detectar Web Spam e páxinas Soft-404 de forma máis eficaz e eficiente que as presentes na literatura. Deste modo, o crawler non dedicará recursos a descargar, indexar e mostrar este tipo de páxinas, mellorando a calidade dos seus repositorios.

Para a actualización de contidos, propúxose un sistema que permite detectar en "tempo real" modificacións en páxinas web. Novamente, mellórase o rendemento do crawler debido a que, por unha parte, non procesará páxinas que non cambiasen, e por outra parte, as páxinas do repositorio serán máis actuais.

Índice general

Lista de Figuras

Lista de Tablas

I

Planteamiento y Contribuciones

I.1 Ámbito

I.1.1 Crawling de la Web

El concepto de Web fue propuesto por Tim Berners-Lee en 1989, y fue al año siguiente cuando se realizaron con éxito las primeras pruebas de concepto sobre un prototipo funcional [BLCL+94]. Desde ese momento, el volumen de datos de la Web ha crecido exponencialmente. Actualmente, gran cantidad de tareas diarias, (comercio electrónico, investigación, entretenimiento, etc.) están presentes en la Web.

La WWW podría considerarse como el repositorio de información más grande, público y no estructurado jamás construido. Según el estudio presentado por Gulli y Signorini [GS05b] en 2005, la Web ya estaba formada por miles de millones de páginas.

Debido al gran tamaño de la Web y a su compleja estructura [BYSJC02], [Sch97], es imprescindible el uso de herramientas que permitan filtrar aquellos documentos que son relevantes para una temática o usuario. Se trata de los conocidos como motores de búsqueda, complejos sistemas, que permiten, entre otras cosas: recopilar los recursos web, almacenarlos y gestionarlos para permitir su acceso y localización. La tarea de recuperación de los recursos web es llevada a cabo por los crawlers, también conocidos como Web Spiders, Web Robots o simplemente Bots. Los crawlers son sistemas capaces de recorrer y analizar la Web en un cierto orden, siguiendo para ello los enlaces existentes entre las diferentes páginas.

Un sistema de crawling comienza con un conjunto de URLs iniciales, que son descargadas, procesadas y analizadas para encontrar nuevos enlaces. Estos enlaces apuntan a páginas web que no han sido descargadas todavía y que serán añadidas a una cola de URLs para su posterior descarga. En el siguiente paso, el crawler selecciona uno de los URLs almacenados en la cola y comienza de nuevo el proceso. El sistema continuará realizando este proceso hasta que la condición de parada se cumpla. A continuación, se muestra el algoritmo básico de funcionamiento de un sistema de crawling:

```
Crawling( Conjunto de páginas iniciales S )

ColaURLs <- S
HACER {
    p <- SeleccionarURL( colaURLs )
    contenido <- Descargar( p )
    (texto, enlaces, estructura, ...)  <- Extraer( contenido )
    colaURLs <- AgregarEnlaces( colaURLs, enlaces )
} HASTA( Condición )
```

Fue en 1993, cuando Matthew Gray, del MIT, desarrolló un script que descargaba automáticamente páginas web. El proyecto fue llamado WWWW (World Wide Web Wanderer), y constituyó el primer crawler de la historia. Un año después, en 1994, apareció el primer índice sobre el que realizar búsquedas. A partir de ese momento, aparecieron multitud de motores de búsqueda basados en sistemas de crawling. Dichos sistemas trataban de competir con los servicios de directorio ofrecidos por AOL y Yahoo!, que en aquel momento dominaban el mercado. Actualmente, todos los motores de búsqueda usan sistemas de crawling, y representan el 10% de las visitas recibidas en los sitios web.

La evolución de la Web ha hecho que surjan diferentes tipos de sistemas de crawling en función de su ámbito de aplicación:

- Sistemas de crawling general: son aquellos crawlers orientados a la Web en general. Debido al gran volumen de datos presentes en la Web, este tipo de crawlers tratan de buscar un equilibrio entre la calidad y la cantidad de los recursos web procesados.

- Sistemas de crawling dirigido: están diseñados para recuperar páginas web y recursos de una temática en particular. Este tipo de crawlers usan estrategias mucho más precisas que los sistemas de crawling generalistas para guiar su recorrido por la Web. Parten de un conjunto de páginas de una misma temática o de un conjunto de documentos de ejemplo, a partir de los cuales deben detectar nuevas páginas web de su misma temática.

Este trabajo está orientado a mejorar el rendimiento de los sistemas de crawling generalistas. Para conocer su funcionamiento es necesario explicar la arquitectura de los mismos. A grandes rasgos, está formada por tres módulos principales.

- Módulo de almacenamiento: es el encargado de almacenar el contenido de los documentos recuperados y su metainformación. Almacena información esencial sobre las páginas descargadas, que será utilizada por el módulo de planificación para poder aplicar las diferentes políticas de crawling definidas.

- Módulo de descarga: encargado de recuperar y procesar el contenido del correspondiente URL.

- Módulo de planificación: es el responsable del mantenimiento de la cola de URLs, y de seleccionar y enviar los URLs en un cierto orden a uno o más procesos de descarga.

En el capítulo IV se realiza una descripción, en más detalle, de la arquitectura y funcionamiento de un crawler.

I.1.2 Desafíos del recorrido de la Web

La tarea de un sistema de crawling presenta numerosos desafíos, debido tanto a la cantidad, como a la variabilidad de la información que tiene que recopilar. Además del ancho de banda y la capacidad de almacenamiento, el desarrollo de un sistema de crawling debe tratar de solucionar muchos otros desafíos.

Entre otros, los desafíos que presenta la Web para un sistema de crawling son los siguientes:

- Resolución DNS: en 1997, Mike Burner [Bur97], lo define como uno de los mayores problemas para los sistemas de crawling. La resolución DNS presenta un problema, debido a la multitud de errores que suceden en el proceso de resolución (errores temporales de resolución, registros DNS erróneos, eficiencia en la resolución, etc.). Parte de esos problemas no pueden ser resueltos desde el punto de vista del crawler ya que no depende del mismo. Sin embargo, los crawlers actuales para no colapsar los servidores DNS, con la posible consecuencia negativa que esto les acarrearía a ellos mismos, realizan DNS caching. El DNS caching consiste en guardar la dirección IP de cada dominio para evitar tener que realizar una resolución DNS en cada página que se desea descargar. Este proceso, además de evitar saturar los servidores DNS, logra mejorar la eficiencia del sistema de crawling.

- Procesamiento de páginas mal formadas: muchas de las páginas web tienen su código HTML con multitud de errores. Además de este problema, hay una gran cantidad de páginas que no incluyen especificación alguna sobre el lenguaje HTML utilizado, que facilite su procesamiento. Por estas razones, para realizar un procesamiento del HTML, o para crear el árbol DOM de la página a analizar, es necesario realizar un preprocesamiento que corrija los errores detectados. Sin embargo, el proceso de corrección del HTML es muy complejo ya que existen multitud de errores que no son triviales de detectar o corregir. El hecho de que el HTML de una página esté correctamente codificado es algo muy importante desde el punto de vista de un crawler, ya que es habitual procesar la página para obtener información estructurada que permita una mejor indexación y búsqueda de los contenidos.

  Los crawlers actuales han mejorado mucho las técnicas de detección y corrección de errores HTML. Algunas de las técnicas existentes son las presentadas en [LRNdS02] y [HLC+06]. Por otro lado, muchos de los crawlers de los principales buscadores penalizan aquellas páginas y sitios web que contienen páginas mal formadas, para así fomentar que los administradores y dueños de páginas web utilicen código HTML correctamente formado.

- Contenido duplicado: actualmente se puede decir que una tercera parte de la Web contiene contenidos duplicados. Existen multitud de estudios que demuestran este hecho. En 1997, Broder *et al.* [BGMZ97a], estiman que un tercio de la Web contiene contenidos casi idénticos. En el año 2004, Fetterly *et al.* [FMN03] demuestran que aproximadamente el 22% de la Web contiene contenidos duplicados. A pesar de estos

resultados, es importante resaltar que parte de dichos contenidos duplicados son legítimos, como es el caso de los sitios web espejo.

Los sistemas de crawling deben evitar procesar contenidos duplicados que no aportan nueva información, ni nuevos enlaces y sí provocan el consumo de recursos del sistema.

Para ello, los crawlers actuales siguen un proceso similar al que se describe. En primer lugar, cuando procesan por primera vez una página web, generan un hash del documento descargado, que por un lado representa a un único documento y por otro es un resumen de su contenido. Tras esto, comprueban que dicho hash no aparece en sus listas, es decir no han procesado ya ese contenido, y, puesto que dicho hash es capaz de resumir el contenido, detectar si existen contenidos no idénticos, pero si muy similares. De esta forma evitarán procesar documentos ya procesados y páginas que se dedican a plagiar contenidos de otros sitios web.

El problema que presentan estos métodos es que la detección del duplicado se realiza una vez se ha visitado la página, procesado y generado el hash, lo cual tiene un alto coste de recursos asociado.

- Web Oculta: los crawlers convencionales sólo pueden acceder a la parte de la Web que se encuentra enlazada como páginas estáticas. Este tipo de páginas contienen una gran cantidad de información, pero representan una pequeña porción de toda la información disponible en la Web. Existe una gran cantidad de páginas y de información que es generada dinámicamente por un servidor en respuesta a eventos del usuario. Uno de los ejemplos más representativos de este tipo de páginas, son las generadas como respuesta a una consulta efectuada por un usuario sobre un formulario web. Este tipo de páginas forman lo que se conocen como Web Oculta del lado servidor.

  Por otro lado, existen otro tipo de páginas web que son accesibles mediante acciones generadas por los usuarios web sobre scripts del lado cliente (por ejemplo, JavaScript). Este tipo de páginas web constituyen lo que se denomina como Web Oculta del lado cliente.

  Debido a que estos conjuntos de páginas representan una gran parte de la Web, los sistemas de crawling deben ser capaces de procesar, de forma automática, las diferentes acciones de los usuarios, necesarias para la extracción del contenido. Para el acceso a la Web del lado servidor los crawlers hacen uso de diferentes técnicas que le permiten detectar formularios y autocompletarlos, ya sea utilizando todas las opciones de respuesta del formulario, o utilizando palabras del dominio (temática) al que pertenece el sitio web. En lo que respecta a la Web del lado cliente, los sistemas de crawling utilizan mini-navegadores que, aunque con una pérdida de rendimiento permiten, simular la ejecución de las acciones generadas por los usuarios web sobre scripts.

  Por otro lado, los grandes buscadores fomentan el uso de etiquetas HTML que faciliten el procesamiento de esta parte de la Web asignando mayor relevancia a aquellos sitios web que las usen. Por último, en el caso de sitios web de relevancia significativa, se crean acuerdos entre los buscadores y propietarios para

acceder a los datos de una forma directa, haciendo innecesario el procesamiento automatizado de la Web Oculta.

- Normalización de URLs: la Web contiene un gran volumen de URLs que apuntan hacia el mismo contenido. Detectar este tipo de enlaces es vital para un crawler, ya que se evitará acceder, descargar y procesar dichos recurso web, para posteriormente detectar que ya habían procesado esa misma página web. A continuación se muestran diferentes URLs que apuntan hacia el mismo contenido.

```
http://dominio.es/x/y/../ -> http://dominio.es/x/ -> http://dominio.es/x/index.html
```

  Actualmente, los crawlers tratan de detectar dichos enlaces eliminando nombres de ficheros por defecto y parámetros de las sesiones de usuario. Sin embargo, muchas de las reglas para detectar que diferentes URLs apuntan al mismo contenido, no son universales, y dependen de la implementación del servidor web.

- Web Spam: según un estudio realizado por Jansen y Spink [JS03], aproximadamente el 80% de los usuarios de buscadores no acceden a las respuestas obtenidas mas allá de la tercera página de resultados. Este resultado junto con el hecho de que la mayor parte de la población usa los buscadores para acceder a datos y servicios, ha hecho que las empresas y los desarrolladores web se preocupen por situar sus sitios web en las primeras posiciones del ranking. Para ello, es importante mejorar la calidad y la actualización de los contenidos del sitio web, así como ser enlazado desde otros sitios web relevantes en la temática tratada en el sitio web. Sin embargo, una parte de los SEOs (Search Engine Optimizers) no usan métodos éticos para lograr dicha relevancia, sino que utilizan técnicas de Web Spam [GGM04].

  Existen multitud de personas y organizaciones interesadas en realizar Web Spam. Entre las motivaciones que les impulsan están las siguientes:

  – Aumento de PageRank para lograr mejorar su posicionamiento y con ello mejorar sus ingresos por publicidad.
  – Dañar a terceros, usualmente utilizado entre empresas competidoras.

  Según el estudio presentado por Ntoulas *et al.* [NM06], en torno al 70% de las páginas del dominio .biz son Web Spam, el 35% de las páginas del dominio .us, y el 15% y 20% de las páginas web bajo los dominios .com y .net, respectivamente. Estos resultados indican que la presencia de Spam en la Web es significativa, y por tanto, su repercusión en la calidad y rendimiento de los crawlers es también muy importante.

Por tanto, la tarea de los sistemas de crawling es evitar el procesamiento de este tipo de páginas, ya que de esta forma, se mejorará la calidad de los contenidos indexados por el buscador y el rendimiento del sistema. Los sistemas de crawling presentes en la literatura utilizan métodos de detección de Web Spam o bien basados en contenido o en análisis de enlaces entre sitios y páginas web. No obstante, como se explicará más adelante, las aproximaciones actuales presentan multitud de deficiencias para su implantación en crawlers. Además, debido a que son muchos los intereses y los medios involucrados en el Web Spam, las técnicas de Spam evolucionan y mejoran día a día.

- Páginas Soft-404: las diferentes implementaciones del protocolo HTTP presentes en la Web dificultan la tarea de los crawlers. Existen multitud de servidores web que ante peticiones de recursos web desconocidos, realizan una redirección hacia otro contenido sin enviar el correspondiente código de error HTTP. A este tipo de páginas se les conoce como páginas Soft-404. Las páginas Soft-404 afectan a la calidad de las páginas procesadas por los crawlers y al rendimiento del sistema, debido a que se dedicarán recursos a procesar y extraer enlaces de esas páginas, en lugar de utilizar esos recursos para tratar páginas con contenido útil. La calidad y la cantidad de páginas procesadas afectará a la calidad del motor de búsqueda y por lo tanto a los resultados mostrados al usuario web.

  Según el estudio realizado por Bar-Yossef *et al.* [BYBKT04], el 29% de los enlaces rotos apuntan a este tipo de páginas web. Este dato demuestra que las páginas Soft-404 constituyen un problema significativo para la Web que debe ser considerado por los crawlers. A pesar de dicha importancia, en el estado del arte sólo existen dos aproximaciones para su detección. Una de ellas está basada en el análisis de la respuesta de los servidores web a peticiones de páginas web aleatorias, determinando si dicho servidor web responde con páginas Soft-404 o no. Sin embargo, este método no detecta todos los tipos de páginas Soft-404 y su eficiencia computacional es baja. La otra aproximación existente analiza los enlaces entre las páginas de un mismo host para determinar si una página es Soft-404. El problema de esta aproximación es la necesidad de procesar todas las páginas de al menos un host antes de poder detectar una página Soft-404.

- Actualización de contenidos: debido a que el contenido de la Web está cambiando continuamente, cuando un crawler ha concluido un recorrido de la Web, debe comenzar nuevamente, con el objetivo de mantener lo más actualizados posible los repositorios utilizados por los buscadores. Sin embargo, existen dos problemas: a) un crawler tiene unos recursos limitados para tratar de mantener totalmente actualizados todos los contenidos procesados, y b) no todas las páginas cambian con la misma frecuencia.

  En algunos casos, puede ser suficiente un crawling total periódico para mantener los repositorios actualizados, pero para otros contenidos que cambian con alta frecuencia, el tiempo que transcurre entre un recorrido completo y el comienzo del siguiente, puede ser demasiado grande como para garantizar la calidad en las búsquedas de los usuarios. Por este motivo, los buscadores consideran otros elementos para intentar mejorar la calidad y actualización de sus contenidos, con el menor impacto posible en el rendimiento. Actualmente, los sistemas de crawling definen políticas de refresco basadas en la frecuencia

de cambios y la edad y longevidad de los contenidos de los sitios web. Sin embargo, dichas políticas están basadas en datos estadísticos, por lo que: a) frecuentemente los cambios realizados en páginas y sitios web son procesados mucho tiempo después de que estos hayan sido realizados, lo que provoca pérdida de calidad en las páginas mostradas al usuario, o b) el crawler visita y procesa una página cuando ésta no ha sido modificada, lo que conlleva un consumo de recursos considerable.

Un crawler de altas prestaciones, como lo son los utilizados por los motores de búsqueda, debe tratar de mitigar o solucionar los problemas descritos. De esta forma, conseguirá que los contenidos obtenidos por el crawler sean de mayor calidad y que la gestión de sus recursos sea más eficiente. Esto último significa tener más recursos libres para procesar mayor cantidad de documentos válidos de la Web.

I.1.3 Optimización de un Crawler Web

Los sistemas de crawling no son capaces de descargar y procesar todas las páginas de la Web, sino que se limitan a una fracción bastante reducida. Por este motivo, es muy importante que la parte considerada contenga la mejor y mayor cantidad de páginas posible. Para ello, el crawler debe seleccionar de forma adecuada las páginas que desea procesar. El principal objetivo de un sistema de crawling consiste en procesar el mayor y mejor número de documentos web con el menor número de recursos y en el menor tiempo posible.

Como se ha discutido previamente, la tarea de un sistema de crawling durante su recorrido de la Web presenta numerosos desafíos. Entre otros, destacan: a) las tecnologías del lado servidor [RGM01] o del lado cliente [Ber01] utilizadas en la construcción de los recursos que componen la Web, b) Web Spam [GGM04] [FMN04], c) repetición de contenidos [KG09], contenidos de mala calidad, etc.

Para lograr optimizar la fracción de la Web a la que se limitan los crawlers, estos deben solucionar aquellos desafíos que empeoran su rendimiento e impiden que se traten más y mejores páginas web.

Existen diferentes estudios que tratan de mejorar el rendimiento de los sistemas de crawling proponiendo arquitecturas o sistemas distribuidos que mejoren su escalabilidad y tratando de reducir los recursos necesarios en ciertas tareas como pueden ser el acceso a disco o el URL caching.

La presente tesis propone una aproximación diferente para mejorar el rendimiento de un crawler. Se trata de una arquitectura que optimiza la calidad de los contenidos que un crawler procesa. Para ello se centra en evitar el procesamiento de páginas cuyo contenido no es adecuado para ser indexado (Web Spam y páginas Soft-404) y optimizar los tiempos de actualización en páginas con contenido relevante.

Existen métodos para detección de Web Spam y páginas Soft-404, pero presentan las siguiente deficiencias que deben ser mejoradas:

- Las técnicas de detección de Spam y Soft-404 basadas en análisis de enlaces precisan procesar uno o

múltiples hosts antes de determinar si es o no Spam o Soft-404. Es decir, no permiten una identificación previa de las páginas evitando el procesamiento de recursos inválidos.

- No detectan todo tipo de páginas de Spam y Soft-404.
- Los resultados de detección en términos de precisión y recall pueden ser significativamente mejorados.
- La complejidad computacional y el tiempo de computo de los métodos existentes deben ser mejorados para su uso en entornos de alto rendimiento.
- Los métodos y técnicas propuestos no han sido diseñados para su uso en sistemas de crawling.

De la misma forma, los métodos existentes para la actualización de contenidos ya procesados y para la detección de cambios web presentan una serie de limitaciones:

- La cabecera Last-Modified del protocolo HTTP no es fiable ya que comúnmente está mal implementada.
- Los protocolos de exclusión de robots ("robots.txt" y "sitemap.xml") necesitan acceso al directorio raíz del sitio web, algo que la mayor parte de los usuarios no tiene.
- Los sistemas RSS no son usados por todos los sitios web y su utilización requiere un gran trabajo adicional por parte de sus creadores.
- Los estudios presentes en el estado del arte se basan en datos estadísticos (histórico de cambios, edad y longevidad de los contenidos, etc.) para construir modelos (usualmente Poisson) con los que detectar modificaciones en páginas web. El problema radica en que se trata de datos estadísticos, lo que implica que en gran parte de las ocasiones ha pasado mucho tiempo desde que realizó el cambio hasta que el sistema de crawling lo procesa, o que el sistema de crawling visita una página y ésta no ha sufrido ningún cambio.

Este trabajo presenta soluciones a las limitaciones comentadas, para, de esta forma, plantear una arquitectura de crawling significativamente más eficiente que las actuales. Las ventajas y desventajas de las arquitecturas de crawling existentes, así como los detalles de los métodos de detección de Web Spam, páginas Soft-404 y actualización de contenidos, serán discutidos en el capítulo II.

En resumen, la arquitectura propuesta evita el tratamiento de este tipo de páginas "basura", consiguiendo proteger, por un lado al usuario final, y por otro a las empresas de motores de búsqueda. Estas últimas son unas de las mas afectadas puesto que pierden prestigio al mostrar páginas de Spam y Soft 404 entre sus resultados a la vez que malgastan sus recursos en analizar, indexar y mostrar resultados de páginas que realmente no deberían ser mostradas. No procesando estos tipos de páginas, se logra que la calidad de la colección indexada

mejore, y a su vez, se evitan posibles riesgos al usuario. Además, la arquitectura propuesta trata de refrescar los contenidos en el momento más adecuado para minimizar los refrescos innecesarios a la vez que su nivel de obsolescencia.

Para ello incorpora tres módulos, uno de detección de páginas de Spam, otro de páginas Soft-404 y un tercer módulo de detección de cambios basado en los accesos de los usuarios. Estos módulos se analizan y discuten en los capítulos V, VI y VII, respectivamente.

I.2 Objetivos

La presente tesis doctoral estudia la construcción de un sistema de crawling eficiente. Para ello, se han realizado diversos estudios que han ayudado a la identificación de los principales desafíos que deben abordar en su operación. Tras esto, se ha analizado el estado del arte de los diferentes problemas que se desean solucionar, y se ha propuesto una estructura global para construir crawlers que procesen la Web de una forma eficiente. Esta arquitectura contiene nuevos módulos y técnicas, que extienden la arquitectura usual de un crawler, para abordar: la detección de Web Spam y páginas Soft-404, y la detección de cambios realizados en páginas web, que mejorará el proceso de re-crawling. Estos nuevos módulos ayudarán a mejorar la calidad de los contenidos procesados y a mejorar la utilización de los recursos del sistema. La arquitectura y las técnicas desarrolladas se han validado experimentalmente con páginas y sitios web reales.

Los objetivos de este trabajo son:

1. Proponer una arquitectura eficiente para sistemas de crawling de la Web. Para ello, se mejorará la calidad y la cantidad de las páginas web procesadas. Por un lado, evitando el procesamiento de páginas y sitios web de Spam y de páginas Soft-404. De esta forma, se lograrán mejorar los contenidos que procesa el sistema y por lo tanto la calidad de las páginas sobre las que el motor de búsqueda realiza las consultas. También se logrará aumentar el uso de recursos libres, ya que no se perderán recursos analizando páginas web inadecuadas.

   Por otro lado, se conseguirá mejorar el proceso de actualización de los contenidos web procesados. Logrando, nuevamente, mejorar los contenidos procesados, ya que serán más actuales, y evitando perder recursos acudiendo a páginas web que no han cambiado todavía.

   Los módulos encargados de realizar las mejoras descritas tendrán que ser lo más eficaces y eficientes posible, para que las mejoras que se obtengan en la calidad no afecten al rendimiento global del sistema. En resumen, la arquitectura propuesta logrará mejorar la calidad y la cantidad de las páginas procesadas por el crawler.

2. Realizar una serie de estudios que ayuden a conocer las características de la Web, su evolución y las

diferencias entre los contenidos de la Web de los diferentes países. Este conjunto de estudios ayudará a identificar los problemas presentes en la Web para su procesamiento por los sistemas de crawling. En dichos estudios se analizarán temas tales como: la similitud de contenidos, la Web Oculta del lado cliente, la edad de las páginas, el crecimiento de la Web, las tecnologías web del lado cliente y del lado servidor, etc. Los problemas en los que se hará más énfasis son: el Web Spam, las páginas Soft-404 y la frecuencia de cambio de las páginas web.

Las conclusiones y datos obtenidos de estos estudios permitirán proponer un conjunto de políticas de actuación que mejoren el rendimiento de los crawlers. Además, estos datos y conclusiones ayudarán a la consecución de los restantes objetivos de esta tesis. Por un lado, permitirán detectar aquellos desafíos más relevantes dentro del tratamiento de la Web, y que, por ello, son significativos para el rendimiento de los sistemas de crawling, y por otro, ayudarán a descubrir nuevos métodos que solucionen dichos desafíos.

3. Proponer nuevas técnicas y algoritmos para permitir a los sistemas actuales de crawling detectar páginas web de Spam. La detección de este tipo de páginas se realizará desde el análisis del contenido, tratando de detectar la mayor parte de los tipos de Web Spam y las diversas combinaciones de técnicas que existen. Para ello, se estudiarán las técnicas de Spam presentes en la Web y se propondrán heurísticas para su detección. La detección de Web Spam se realizará mediante la combinación del conjunto de heurísticas propuestas. Este conjunto de técnicas formarán parte de un nuevo módulo de la arquitectura de crawling propuesto, que se encargará de evitar el procesamiento de páginas de Spam. El análisis y detección se realizará en un primer momento tras ser descargada, de esta forma se logrará mejorar los contenidos procesados y se evitará la pérdida de recursos asociada de ser detectada a posteriori.

4. Proponer nuevas técnicas y algoritmos para permitir a los sistemas actuales de crawling detectar páginas Soft-404. Del mismo modo que las páginas de Spam, la detección se realizará desde el análisis del contenido. Se propondrán heurísticas para su detección, que formarán parte de un nuevo módulo de la arquitectura de crawling propuesta, que se encargará de evitar el procesamiento de páginas Soft-404. El análisis y detección se realizará en un primer momento tras ser descargada. De esta forma se logrará mejorar el contenido procesado y se evitará la perdida de recursos del sistema.

5. Proponer un sistema que permita conocer las modificaciones realizadas en páginas y sitios web. El sistema formará parte de un módulo de la arquitectura de crawling propuesta y será el encargado de notificar al sistema cuándo una página web ha sido modificada. Se basará en una arquitectura colaborativa entre el cliente (navegador) y el servidor, por lo que deberá ser poco intrusiva y altamente escalable. De esta forma el sistema de crawling evitará realizar accesos a recursos web que no hayan sido modificados y no perderá modificaciones en el tiempo transcurrido entre dos procesamientos. Esto repercutirá en la calidad de los contenidos procesados y en el uso de los recursos disponibles, lo que permitirá que estos sean usados en el procesamiento de nuevos recursos web.

6. Validar la eficacia y eficiencia de las técnicas y métodos propuestos con experimentos sobre páginas y sitios web reales, demostrando que es posible la construcción de sistemas de crawling capaces de detectar y evitar páginas web de Spam Soft-404, así como de realizar un re-crawling eficiente de los contenidos previamente indexados.

I.3 Principales Contribuciones Originales

Este trabajo presenta las siguientes aportaciones principales:

1. Una arquitectura de crawling eficiente para la recopilación de información de la Web, evitando para ello el procesamiento de recursos sin contenido válido/útil a la par que minimizando el tiempo que un documento permanece desactualizado. Para ello, parte de la arquitectura de sistemas de crawling existentes para extenderla integrando componentes para la detección de Web Spam y páginas Soft-404, así como para mejorar el proceso de actualización de los contenidos del repositorio utilizando el menor número de recursos. La descripción y discusión de la citada arquitectura se realiza en el capítulo IV, y ha sido presentada en:

   - Architecture for a Garbage-less and Fresh Content Search Engine [PALGC12c]. 5th Internacional Conference on Knowledge Discovery and Information Retrieval (IC3K 2012).

2. Un conjunto de estudios que analicen y caractericen la Web tanto a nivel global como a nivel nacional. La idea de dichos estudios es conocer en más detalle con que problemas se encuentran los sistemas de crawling durante su recorrido de la Web y su evolución en el tiempo.

   Se han realizado dos estudios. El primero de ellos analiza la Web Oculta del lado cliente, su presencia en la Web y la capacidad de los crawlers actuales para procesar las tecnologías web usadas en el lado cliente. Esto permitirá conocer las deficiencias que tienen los crawlers actuales, y que parte de la Web no está siendo tratada. Este estudio se describe en la sección III.1, y ha sido presentado previamente en:

   - A scale for crawler effectiveness on the client-side hidden web. Computer Science and Information Systems, vol. 9, issue 2. Journal indexado en JCR [PALGC12d].
   - Analysing the Effectiveness of Crawlers on the Client-Side Hidden Web. Agents and Multi-agent systems for Enterprise Integration (ZOCO 2012). Lecture Notes in Advances in Soft Computing, vol. 157 [PALGC12a].
   - Web Oculta del Lado Cliente: Escala de Crawling. 10th Jornadas de Ingeniería Telemática (JITEL 2011) [PALGC11].

El segundo de los estudios, se centra en un análisis de diferentes características de la Web Global y Española. Entre otras características se han estudiado: la similitud de contenidos, la Web Oculta del lado cliente, la edad de las páginas, el crecimiento de la Web, las tecnologías web del lado cliente y del lado servidor, etc. Los resultados de este análisis constituyen el punto de partida para el diseño de la arquitectura de crawling propuesta y de los módulos que ésta contiene. Este último estudio se describe y discute en la sección III.2 y ha sido publicado previamente en:

- Evolución de la Web Española y sus Implicaciones en Crawlers y Buscadores. 2th Congreso Español de Recuperación de Información (CERI 2012) [PAC11].
- En revisión: Evolution of the Web and its implications on crawlers and search engines. Journal of Web Engineering. Journal indexado en JCR.

3. Nuevas técnicas para la detección automática de páginas de Web Spam. En el proceso de crawling de la Web existen multitud de recursos inadecuados que no deben ser procesados, entre los que se encuentran las páginas de Web Spam. Las técnicas presentadas están basadas en un conjunto de heurísticas que analizan el contenido de las páginas para detectar la mayor parte de los tipos de Web Spam (Content Spam, Cloaking, Redirection Spam y Link Spam). Además de las heurísticas, también se presenta un método de combinación del conjunto de heurísticas propuestas, para su posterior uso como un módulo que extienda a un sistema de crawling y le permita detectar y evitar páginas de Spam.

   El análisis de cada una de las heurísticas, su método de combinación para la creación final de un módulo de detección y los resultados obtenidos y su comparativa con los presentes en el estado del arte, se presentan en el capítulo V. Tanto las heurísticas como el método de combinación han sido presentados previamente en: [PALGC12b] y [PALGC12c].

   - Architecture for a Garbage-less and Fresh Content Search Engine [PALGC12c]. 5th Internacional Conference on Knowledge Discovery and Information Retrieval (IC3K 2012).
   - Analysis and Detection of Web Spam by Means of Web Content. 5th Information Retrieval Facility Conference (IRFC 2012). Lecture Notes in Computer Science, vol. 7356 [PALGC12b].
   - SAAD, a Content based Web Spam Analyzer And Detector. Journal of Systems and Software. Journal indexado en JCR [PAC13b].

4. Nuevas técnicas para la detección automática de páginas Soft-404. Este tipo de páginas, al igual que las de Web Spam, no deben ser procesadas, ya que su contenido y enlaces no son relevantes. Las técnicas propuestas se basan en una serie de heurísticas que caracterizan el contenido de las páginas Soft-404 para posibilitar su detección. Dichas técnicas forman parte de un módulo de la arquitectura de crawling propuesta que permite detectar dicho tipo de páginas. De esta forma se mejora la calidad de las páginas

procesadas y también el uso de los recursos, ya que aquellos recursos que no traten este tipo de páginas pueden ser utilizados para procesar otras páginas web.

Otra contribución importante fue la creación de datasets de páginas Soft-404. A diferencia de Web Spam, no existe un dataset público de páginas Soft-404 donde las heurísticas y método puedan ser probadas. Por ello, fue necesario crear 4 datasets de diferentes tamaños (65625 y 100000 páginas) y con diferentes porcentajes de páginas Soft-404 sobre el número total de páginas (25% y 50%).

El capítulo VI contiene la descripción del conjunto de heurísticas propuestas, el método usado para su combinación y los datasets generados. Estas tres contribuciones han sido presentadas en:

- Architecture for a Garbage-less and Fresh Content Search Engine [PALGC12c]. 5th Internacional Conference on Knowledge Discovery and Information Retrieval (IC3K 2012).
- Analysis and Detection of Soft-404 Pages [PAC13a] Third International Conference on Innovative Computing Technology (INTECH 2013).

5. Un nuevo método para la detección de cambios en páginas web en "tiempo real", que permita a los sistemas de crawling mantener su contenido lo más actualizado posible. El nuevo método se basa en la colaboración de las páginas a ser monitorizadas, de modo que sean éstas las que informen de sus cambios cuando son accedidas por los usuarios. Esto permite, por un lado, mejorar la calidad de los contenidos indexados, ya que serán más actuales, lo cual repercute en la calidad de las búsquedas y en la experiencia del usuario web, y por otro, mejorará el uso de recursos del sistema, ya que se evitará volver a procesar páginas que no han cambiado.

   La descripción de los componentes y funcionamiento del sistema se realiza en el capítulo VII, y ha sido presentado en:

   - Architecture for a Garbage-less and Fresh Content Search Engine [PALGC12c]. 5th Internacional Conference on Knowledge Discovery and Information Retrieval (IC3K 2012).
   - En revisión: Distributed and Collaborative Web Change Detection System. Knowledge and Information Systems. Journal indexado en JCR.

Este trabajo describe y discute la arquitectura de crawling eficiente centrándose en las aportaciones presentadas. Otras partes de la misma, tales como los mecanismos de extracción o normalización de URLs, son descritos a nivel general para proporcionar el adecuado contexto para los objetivos de este trabajo.

I.4 Estructura de la Tesis

El capítulo II, "Estado del Arte", realiza una introducción sobre la Web y su procesamiento. Se identifican y describen los principales desafíos que debe abordar un sistema capaz de realizar el procesamiento de la Web. El capítulo se centra en las técnicas propuestas en la literatura para abordar las diferentes tareas identificadas. No obstante, se hace especial énfasis en las técnicas y métodos más relacionados con las aportaciones de este trabajo. Se analizarán en detalle técnicas de detección de Web Spam, de páginas Soft-404 y del mantenimiento actualizado de los repositorios.

En el capítulo III, "Estudios de Caracterización de la Web", se presentan dos estudios realizados sobre la Web, que constituyen las bases y aportan los datos necesarios para el desarrollo de las diferentes técnicas y arquitectura propuestas. Por un lado, se realiza un estudio sobre la Web Oculta del lado cliente, las tecnologías más usadas y su presencia en la Web. Se presenta una escala que mide la capacidad de los sistemas de crawling para el tratamiento de las tecnologías del lado cliente. La finalidad de este estudio es conocer qué parte de la Web no esta siendo procesada y si, por lo tanto, es necesario mejorar el procesamiento de la misma.

Por otro lado, se presenta un estudio sobre la evolución de las principales características de la Web Española y Global durante 3 años, realizando una comparativa entre ambas y analizando la evolución de los resultados obtenidos. El análisis realizado se ha hecho desde el punto de vista de los crawlers y motores de búsqueda. Por último, se discuten las conclusiones obtenidas de los estudios.

El capítulo IV, "Arquitectura de Crawling Eficiente", describe en detalle la arquitectura del sistema de crawling propuesta en este trabajo. Se comienza con una descripción global de la arquitectura propuesta, detallando posteriormente las funciones y la estructura de cada uno de los módulos que la componen. Cada uno de los módulos que componen la arquitectura se analizarán en detalle en capítulos posteriores.

El capítulo V, "Módulo de Detección de Web Spam", describe detalladamente el módulo de detección de Web Spam incluido en la arquitectura propuesta. Se comienza analizando el conjunto de técnicas de detección de Spam presentes en la literatura. Tras esto, se discuten y analizan el conjunto de técnicas propuestas para detectar Web Spam. El análisis de dichas heurísticas se realiza desde el punto de vista de la eficacia y la eficiencia de las mismas. Por último, se presentan los resultados obtenidos aplicando dichas técnicas sobre diversos datasets y se comparan los resultados con los obtenidos con los métodos y técnicas presentes en el estado del arte.

El capítulo VI, "Módulo de Detección de Páginas Soft-404", analiza el módulo de detección de páginas Soft-404 que se incluye en la arquitectura de crawling propuesta. En primer lugar se presenta un análisis de las técnicas utilizadas actualmente para la detección de este tipo de páginas. Posteriormente, se discuten las nuevas técnicas presentadas, analizando su eficacia y eficiencia en comparación con las técnicas presentes en la literatura.

El capítulo VII, "Módulo de Detección de Cambios en Páginas Web", describe el método encargado del

proceso de actualización de los contenidos procesados por el sistema de crawling. Este módulo incluido en la arquitectura propuesta es el responsable de detectar cambios en páginas y sitios web previamente procesados, y notificar dichos cambios al sistema para que el correspondiente proceso de crawling regrese a dicha página web. El capítulo comienza describiendo los sistemas y métodos actuales utilizados en los procesos de actualización de los motores de búsqueda. A continuación, se describe la arquitectura del módulo propuesto, sus componentes y el funcionamiento del mismo. Por último, se discuten las mejoras obtenidas frente a los métodos utilizados por los sistemas de crawling convencionales.

Finalmente, el capítulo VIII, "Conclusiones y Trabajo Futuro", discute los resultados y aportaciones presentadas en este trabajo, realizando una comparativa con los trabajos presentes en el estado del arte. Posteriormente, se exponen las conclusiones obtenidas de esta tesis doctoral, y finalmente se comentan las líneas de trabajo futuro del autor.

II

Estado del Arte

Este capítulo realiza una introducción sobre la Web y los diferentes desafíos que presenta su procesamiento de forma automatizada. En primer lugar, se analizan diferentes estudios existentes sobre la caracterización de la Web. A continuación se comentan los principales trabajos existentes en la literatura que abordan los diferentes desafíos que plantea el procesamiento de la Web, dando especial énfasis a aquellos que están más relacionados con las aportaciones de este trabajo.

La sección II.1 es una introducción general sobre las características de la Web y las problemáticas que plantea la recuperación de información en este entorno, y presenta la arquitectura general de un crawler global. La sección II.2 explica diferentes aproximaciones para implementar un crawling eficiente de la Web. La sección II.3 se centra en los desafíos de la Web que afectan al filtrado de los contenidos procesados, para lograr una mejor calidad de los mismos y un mejor uso de los recursos. La sección II.4 explica la problemática del mantenimiento actualizado de los repositorios de los buscadores, y comenta en detalle las diferentes aproximaciones existentes en la literatura. Por último, la sección II.5 muestra las conclusiones obtenidas del análisis del estado del arte, discutiendo las ventajas y desventajas de las principales arquitecturas de crawling y de las técnicas existentes para abordar los desafíos planteados.

II.1 La Web

La naturaleza de la Web basada en hipervínculos, la hace diferente a cualquier otra colección de datos. En base a un estudio presentado por Broder *et al.* [BKM+00], donde hace un análisis de los enlaces entre páginas del mismo dominio, entre dominios del mismo país y entre dominios globales, se puede decir que la estructura de la Web puede entenderse como una pajarita. En la Figura II.1 se muestra la estructura de la Web.

El centro de la pajarita, contiene, aproximadamente, el 28% de las páginas presentes en la Web, y está formado por un conjunto de páginas que están fuertemente conectadas unas con otras. Por otro lado, el 21% de la Web contiene hipervínculos con los que llegan a páginas del núcleo, pero no a la inversa. Existe otro 21% de páginas web que pueden ser alcanzadas desde las páginas del núcleo, pero que no pueden llegar al núcleo. Estos dos grupos constituyen los lazos de la pajarita, la entrada y la salida de páginas hacia el núcleo. Saliendo y entrando de los lazos de la pajarita aparece lo que se denomina como tentáculos, que contienen aproximadamente el 22% de la Web, y que son caminos sin salida, con excepción de unos pocos que conectan el grupo de entrada con el de salida. Por último, el 8% restante forman conjuntos de páginas web conectadas entre sí, pero no con el resto de la Web, y se conoce con el nombre de islas.

La estructura en forma de pajarita permite que un sistema que recorra de forma automatizada la Web. Partiendo de un conjunto adecuado de páginas web y con recursos y tiempo infinito, dicho sistema podría alcanzar todos los recursos web existentes.

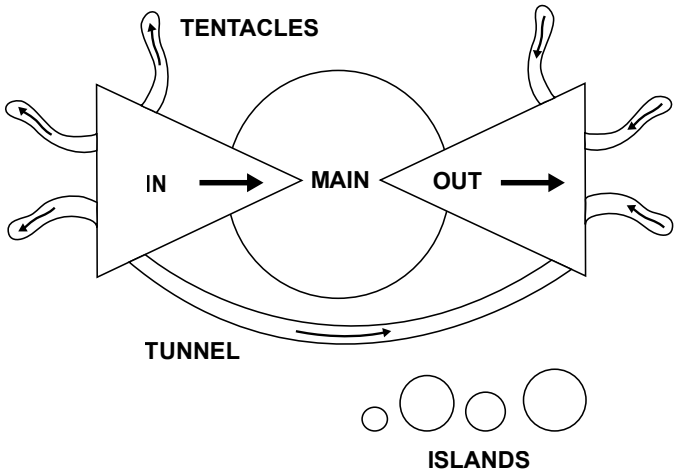


Figura II.1: Estructura de la Web

II.1.1 Datos estadísticos de la Web

Son numerosos los trabajos existentes que abordan el análisis de diferentes aspectos de la Web. A continuación se describen los datos más relevantes que ayudarán a conocerla y comprenderla mejor.

- Tamaño: es uno de los datos más representativos y relevantes de la Web y de su evolución. Sin embargo, estimar el tamaño de la Web es muy complejo, debido principalmente a su naturaleza altamente dinámica. Por ello, los estudios sobre el tamaño de la Web se centran en el análisis de la Web indexable [GS05b], es decir, aquella parte de la Web que está siendo considerada por los motores de búsqueda.

  En los siguiente estudios se puede observar el rápido crecimiento que ha experimentado la Web a lo largo del tiempo:

  - 1997: Bharat y Broder [BB98] determinan que el tamaño de la Web indexada es de 200 millones de páginas.
  - 1998: en [LG00], Lawrence y Giles presentan un estudio donde la Web indexada es de 800 millones de páginas, aproximadamente 6 terabytes de texto. Esto indica que en un año la Web ha crecido en más de 600 millones de páginas.
  - 2005: un estudio llevado a cabo por Gulli y Signorini [GS05b] indicaba que en Enero de 2005 la Web indexada era de 11,5 mil millones de páginas. Esto significa que en 7 años el tamaño de la Web se había multiplicado casi por 15.

Sin embargo, todas estas estimaciones están obsoletas. Tanto buscadores como crawlers precisan estimar el tamaño de la Web para mejorar sus políticas de actuación y su gestión de recursos.

- Web Oculta: es el conjunto de páginas web que son generadas como respuesta a una consulta efectuada por un usuario sobre un formulario web (Web Oculta del lado servidor), o mediante acciones desencadenadas en el lado cliente durante la navegación de los usuarios por los sitios web (Web Oculta del lado cliente). El estudio llevado a cabo por Bergman en 2001 [Ber01] que analiza la Web Oculta del lado servidor, permite obtener las siguientes características:

  - La información contenida en la Web Oculta es entre 400 y 500 veces más grande que la comúnmente definida como Web.
  - La Web Oculta contiene aproximadamente unos 7500 terabytes, frente a los 19 terabytes de la Web normal.
  - En 2001 ya existían más de 200000 sitios web que formaban parte de la Web Oculta.
  - Se estima que la calidad de los contenidos de la Web Oculta es entre 1000 y 2000 veces mayor que el existente en la Web convencional, por tratarse normalmente de información almacenada en bases de datos subyacentes, y por lo tanto altamente estructurada.
  - De media, los sitios web de la Web Oculta reciben un 50% más de tráfico mensual que los sitios web comunes.

  Estos datos hacen evidente la importancia de la Web Oculta ya en 2001. Y desde entonces no ha dejado de crecer, representando actualmente una gran parte de la Web. Por este motivo, su tratamiento es esencial desde el punto de vista de los crawlers y buscadores.

- Idiomas: otro aspecto importante para comprender la distribución de los contenidos de la Web, es conocer los idiomas en los que estos están escritos. En el estudio [Net01] realizado sobre 2024,7 millones de páginas web determinan que el 56,4% de la Web se encuentra escrito en Inglés. En la Tabla II.1 se muestran todos los resultados.

  Los datos muestran una amplia diferencia entre el número de páginas en Inglés y en el segundo lenguaje, el Alemán, con un 7,7% de las páginas web. Tras el Alemán, aparecen el Francés con un 5,6% y el Japones con un 4,9%. La diferencia entre los 4 idiomas más usados y los siguientes es significativa, pasando de 98,3 millones de páginas en Japones a casi la mitad, 59,9 millones, en Español.

Además de los estudios presentados, que se centran en la Web Global, existen otros estudios orientados al análisis de la Web Nacional de diversos países. Si bien estos trabajos no analizan la Web Global, sí pueden ayudar a conocer diversas características de la misma, ya que la Web Global está formada por un conjunto de

Idioma	Número de páginas web (millones)	%
Inglés	1142,5	56,4%
Alemán	156,2	7,7%
Francés	112,1	5,6%
Japones	98,3	4,9%
Español	59,9	3,0%
Chino	48,2	2,4%
Italiano	41,1	2,0%
Holandés	38,8	1,9%
Ruso	33,7	1,7%
Koreano	30,8	1,5%
Portugués	29,4	1,5%
Otros	232,7	11,4%
Total	2024,7	100%

Tabla II.1: Distribución de los idiomas presentes en la Web [Net01]

Webs Nacionales con algunas características diferenciadoras. Entre dichos estudios destaca el de Baeza-Yates *et al.* en el año 2007 [BYCE07], donde realizan un análisis de diversas características de 8 Webs Nacionales. A continuación se resumen los datos más destacados del estudio:

Web Nacional	Tamaño medio de página
África	13 KB.
Brasil	24 KB.
Chile	22 KB.
Grecia	22 KB.
Korea del Sur	14 KB.
Portugal	21 KB.
Tailandia	10 KB.

Tabla II.2: Tamaño medio de página en la Web [BYCE07]

- Tamaño de páginas web: En la Tabla II.2 se muestran los tamaños medios de las páginas web de sitios Web Nacionales analizados en el estudio. Aunque, en general, se observa que la mayor parte de los

sitios Webs Nacionales estudiados tienen un tamaño medio de página web superior a 20 KB, se observan diferencias, que pueden ser debidas distintas formas de codificar el HTML usado en las páginas.

- Edad: La edad representa el tiempo de validez de una página, es decir el tiempo que ha pasado desde que fue creada hasta que desaparece o es modificada. Para determinar la edad de una página es habitual utilizar la información de la cabecera "Last-Modified" presente en las respuestas HTTP de los servidores web. Los resultados se muestran en la Tabla II.3.

Web Nacional	Edad (Meses)
Brasil	11,6
Chile	13
Grecia	17,7
Korea del Sur	7,3

Tabla II.3: Edad media de las páginas web en diversas Web Nacionales [BYCE07]

- Páginas web por sitio: Otra característica importante sobre la Web y su evolución es el número de páginas por sitio web. La segunda columna de la Tabla II.4 muestra los resultados medios obtenidos en cada uno de los países estudiados.

Web Nacional	Número medio de páginas por sitio web	Número medio de sitios web por dominio
Brasil	66	2,1
Chile	58	1,1
Grecia	150	1,2
Indochina	549	1,6
Italia	410	1,3
Korea del Sur	224	26,1
España	52	2,5
Inglaterra	248	1,2

Tabla II.4: Número medio de páginas por sitio web y número medio de sitios web por dominio [BYCE07]

- Sitios web por dominio: Representa el número medio de sitios web que suele contener cada dominio. El resumen de los resultados se muestran la tercera columna de la Tabla II.4.

  Los resultados indican que, de media, los dominios presentan entre 1,1 y 2,5 sitios web. Es importante explicar el resultado obtenido para Korea del Sur, donde cada dominio contiene en media 26,1 sitios web.

Según lo que indican Baeza-Yates *et al.* en su estudio [BYCE07], esto es consecuencia de la gran cantidad de Spam presente en la Web de este país.

- Enlaces: Una de las características más importantes de la Web es su estructura basada en hipervínculos. La estructura de la Web está creada en base a enlaces entre páginas, sitios web y dominios, en diferentes niveles. En la Tabla II.5 se muestra el resultado del análisis del número de los enlaces entrantes y salientes en páginas web.

  Se puede observar que el grado de enlaces de entrada varía desde 8,3 de Chile hasta los 26,2 de Indochina. En lo que respecta al grado de enlaces de salida, este varía desde los 3,6 de España hasta los 31,8 de Indochina. Considerando estas Web nacionales como representativas de la Web Global, se podría decir que la Web Global tiene de media un grado de enlaces de entrada de 12,22 y una media de grado de enlaces de salida de 19,98.

Web Nacional	Grado de enlaces de entrada	Grado de enlaces de salida
Brasil	14	21,1
Chile	8,3	13,7
Grecia	10,3	17,2
Indochina	26,2	31,8
Italia	27,9	31,9
Korea del Sur	15,7	18,7
España	11,2	3,6
Inglaterra	16,2	18,9

Tabla II.5: Grado de entrada y de salida de enlaces [BYCE07]

Por último, cabe destacar un conjunto de estudios que caracterizan la Web desde diferentes puntos de vista. Entre los estudios centrados en la estructura de la Web, destaca el presentado por Baeza-Yates *et al.* [BYSJC02] en el que analiza la estructura de la Web, su dinamismo y su relación con la calidad del contenido. Otro estudio interesante relacionado con la estructura de la Web de un país, fue el realizado por Baeza-Yates y Poblete [BYP06] sobre la Web chilena.

Baeza-Yates *et al.* en [BYCE07] realizan un análisis de diversas características de la Web en varios niveles: página web, sitio web y dominios nacionales. En el año 2000, Sanguanpong *et al.* [SPnK$^+$00], presentaron un estudio sobre los servidores web y la Web de Tailandia. Baeza-Yates *et al.*, presentaron dos artículos, [RBYL05] y [BYC04], que se centran en el análisis de las características de la Web Española y Chilena, respectivamente. En el año 2002, Boldi *et al.* [BCSV02], presentaron un interesante artículo sobre la Web Africana, analizando

su contenido, estructura y grafo. Gomes *et al.* [GS05a], han llevado a cabo un estudio para la caracterización de la Web Portuguesa. En dicho estudio se analizan características tales como: el número de dominios por sitio, el tamaño de los documentos de texto, etc. Años después, Miranda y Gomes [MG09], realizaron un estudio que presenta la evolución de la Web Portuguesa. El estudio de la evolución se hizo comparando los resultados obtenidos en dos análisis realizados con un intervalo de 5 años. En el 2005, Modesto *et al.* [MPZ+05], presentaron un artículo que se centra en en análisis del dominio .br. Por último, un estudio similar a los anteriores, fue el realizado por Efthimiadis y Castillo [EC04], donde los autores se han centrado en caracterizar la Web Griega.

Estos y otros estudios sobre la Web, permiten tener un mayor conocimiento sobre la misma. Sin embargo, ninguno de ellos realiza un estudio en profundidad sobre la Web Global y aquellas características determinantes desde el punto de vista de los sistemas de crawling y buscadores. Además, no existen trabajos que comparen la Web Global y la Nacional de un país concreto. Este análisis aportaría ideas y políticas sobre cómo deben comportarse los sistemas de crawling y buscadores frente a sus diferencias y similitudes. Por último, son muy pocos los trabajos que estudian la evolución de las características de la Web en diversos períodos para poder contrastar los resultados para obtener conclusiones fiables.

II.2 Crawling Eficiente de la Web

La Web puede ser considerada como el repositorio de información más grande jamás construido. En el año 2005, un estudio de Gulli y Signorini [GS05b], indicaba que la Web estaba formada por miles de millones de páginas web.

Para lograr acceder a tal cantidad de información se hace indispensable el uso de herramientas que permitan su procesamiento y búsqueda de forma automática. Esto es posible debido a su estructura basada en hipervínculos, que permite alcanzar una gran parte de la Web únicamente siguiendo los enlaces para obtener nuevos documentos. A los sistemas capaces de procesar la Web y realizar búsquedas sobre ella se les denomina motores de búsqueda. La arquitectura básica de un motor de búsqueda está formada principalmente por tres componentes: el módulo de crawling, el módulo de indexación y el módulo de consulta.

En la Figura II.2 se muestra el esquema básico de un motor de búsqueda, como se describe en [ACGM+01].

El módulo de crawlers recibe un conjunto de URLs que apuntan a un conjunto de recursos web. Por una parte, extrae el conjunto de enlaces existentes entre los diferentes documentos descargados. Por otra, almacena en el repositorio el conjunto de páginas web descargadas.

El módulo de control del crawler, es el encargado de decidir qué enlaces se van a visitar y en que orden. En otras palabras, es el responsable de gestionar la dirección de exploración del crawler y establecer su criterio de parada.

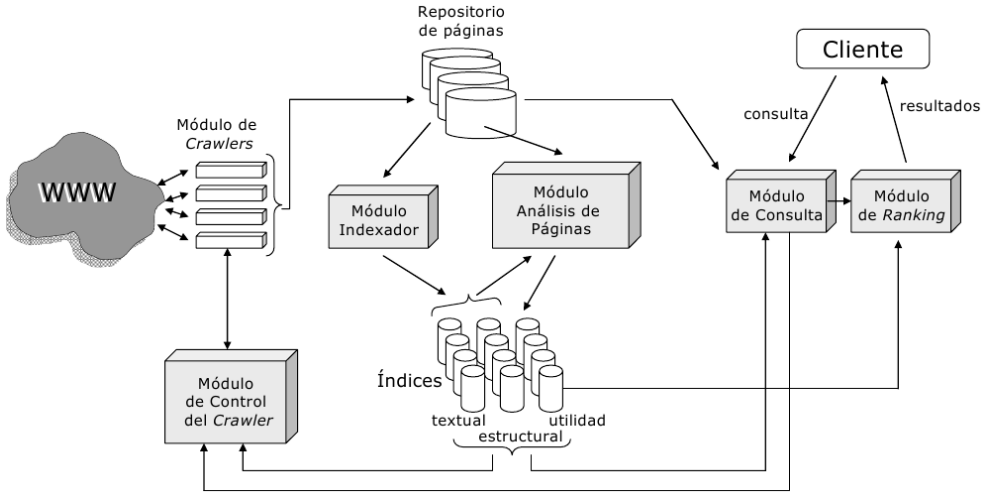


Figura II.2: Arquitectura de un motor de búsqueda

El módulo de indexación es el encargado de la generación del índice o índices sobre los que se realizará la búsqueda. Para ello, extrae todas las palabras de cada página y le asocia el URL en el que cada palabra ha aparecido. Tras esto, se genera un índice que contiene el conjunto de palabras con los documentos donde aparece cada una.

El repositorio de páginas es donde se almacena el conjunto de páginas descargadas por el crawler.

Por último, el módulo de consulta, es el responsable de procesar las búsquedas de los usuarios, y devolver un conjunto de documentos web ordenados por orden de relevancia. Para ello, accede a la información almacenada en los índices, la procesa, y, aplicando algún tipo de algoritmo de ordenación, obtiene el listado de documentos para el usuario final.

Hemos descrito la arquitectura básica de un motor de búsqueda y su funcionamiento. Pero para que dicho motor de búsqueda sea capaz de escalar a una porción significativa de la Web, son muchos los aspectos que requieren un tratamiento especial y por lo tanto una descripción más detallada. Son muchos los desafíos que presenta el tratamiento de la Web por parte de los crawlers. Entre ellos, se pueden citar: a) resolución DNS, b) Web Oculta, c) normalización de URLs, d) caching de URLs, e) almacenamiento, f) Spam, g) refresco de contenidos, etc.

Actualmente, la mayor parte de los motores de búsqueda web pertenecen a grandes empresas, que han realizado importantes inversiones en investigación para el desarrollo de sus sistemas. Por este motivo, sus resultados y detalles sobre la arquitectura concreta de sus sistemas no se han hecho públicos por considerarse

de valor estratégico. Existen pocos documentos de carácter público, y entre ellos destaca la implementación del crawler "Mercator", realizada por Compaq para Altavista [HN99] y una descripción del primer crawler utilizado por Google [BP98].

No obstante, existe un conjunto de estudios de carácter público, que tratan de mejorar la eficiencia de los sistemas de crawling desde diferentes aproximaciones, abordando de forma parcial las problemáticas asociadas a los distintos desafíos que presenta el procesamiento automatizado de la Web.

A continuación se comentan los principales trabajos existentes en la literatura, agrupados en base al desafío web que abordan para optimizar el funcionamiento de un sistema de crawling.

- Ordenación de los recursos que se van a procesar: Debido a que un crawler no puede volver a visitar todas las páginas ya procesadas en un periodo de tiempo razonable, es necesario ordenarlas para procesar antes aquellas que considera más relevantes. Entre las diferentes aproximaciones existentes en el estado del arte, destaca la presentada por Cho *et al.* [CGmP98]. Los autores calculan la relevancia de una web, y con ello la prioridad para que se vuelva a procesar, atendiendo a los siguientes parámetros: a) enlaces a la página web desde otras páginas, b) enlaces que contiene la página, c) PageRank, d) sitio web al que pertenece la página y e) relevancia de la página web para una temática en concreto. Aplicando el método que proponen, los autores afirman lograr una importante mejora en los tiempos en que las páginas más importantes son visitadas de nuevo.

- URL caching: Un crawler debe comprobar continuamente que páginas ya ha procesado y cuales debe procesar. Esta comprobación se realiza para las miles de páginas que está descargando, por ello necesita que este proceso sea rápido.

  Debido a que el número de recursos presentes en la Web es extremadamente grande, las listas de URLs procesadas y pendientes de un sistema de crawling, tienen que almacenarse en disco. Realizar estas operación sobre disco es mucho más lento que en cache o RAM, sin embargo el tamaño de estas memorias no es suficiente para mantener las listas completas. La solución a este problema pasa por mantener en caché un subconjunto "ideal" de las URLs que deben descargarse próximamente. Para la creación de ese subconjunto "ideal" existen diferentes políticas (reemplazo aleatorio, cache estática, LRU o CLOCK, entre otras). En [BNW03], Broder *et al.* analizan los diferentes algoritmos de caching y concluyen que el uso de caching es muy efectivo, y que, concretamente con una cache de, aproximadamente 50000 registros, se logra una tasa de éxito del 80%.

- Mayor número y calidad en los recursos procesados: Aumentando los contenidos procesados y la calidad de estos se mejorarán los resultados mostrados al usuario. Para mejorar el número de documentos recuperados por un sistema de crawling y la calidad de los mismos, son numerosos los trabajos orientados hacia la obtención de información de la Web Oculta. Según el estudio llevado a cabo por Bergman [Ber01],

más del 80% de los contenidos de la Web se encuentran en ella, y son de mayor calidad. El desafío que representa la Web Oculta para los sistemas de crawling es doble: por una parte la complejidad para el acceso a sus contenidos y por otra parte el coste computacional que requiere el procesamiento de un número tan grande de recursos. Por ello, un camino para conseguir un crawling eficiente de la Web será el tratamiento eficiente de la Web Oculta.

Entre las aproximaciones presentes en la literatura destacan los trabajos realizados por Álvarez *et al.* En su trabajo final [Álv07], Álvarez *et al.* definen una arquitectura y un conjunto de técnicas que posibilitan y mejoran la eficiencia del acceso a información de la Web Oculta, tanto del lado cliente como del lado servidor. Para el tratamiento de la Web Oculta del lado cliente, los autores proponen el uso de mini-navegadores que permitan simular la ejecución de las acciones de los usuarios. En lo que respecta al lado servidor, describen novedosas técnicas para identificar y consultar automáticamente formularios web. Por último, proponen el uso de un conjunto de técnicas que permiten extraer automáticamente datos estructurados obtenidos de consultas realizadas sobre formularios web.

Otra aproximación para mejorar el número y calidad de los documentos procesados consiste en evitar el procesamiento de contenidos inadecuados o de baja calidad, como Web Spam o páginas Soft-404, para liberar recursos de crawling que podrán ser utilizados para poder procesar otros recursos de mayor calidad. Debido a que el presente trabajo incluye en su arquitectura algoritmos para la detección de dicho tipos de páginas, estos desafíos se analizarán en mayor detalle en las siguientes secciones.

- Detección de contenidos repetidos: Actualmente existe en la Web gran cantidad de contenido duplicado o muy similar. Estudios como el presentado por Fetterly *et al.* [FMN03], estiman que aproximadamente el 22% de la Web contiene contenidos casi idénticos. Los sistemas de crawling deben evitar procesar contenidos duplicados que no aportan nueva información, ni nuevos enlaces y sí provocan el consumo de recursos del sistema.

  Los sistemas de crawling existentes utilizan aproximaciones similares a la presentada por Broder *et al.* [BGMZ97a]. Utilizan un sistema que "firma" los documentos descargados sin realizar ningún análisis lingüístico de su contenido. Para generar la firma de un documento, se divide en la lista de las palabras que lo forman, eliminando las stopwords. Cada palabra de dicha lista viene representada por ella misma y por sus palabras adyacentes, formando lo que se denomina un token. De esta forma, un documento estará formado por n palabras y m tokens, donde siempre $n > m$. Tras esto, cada uno de los tokens (grupos de palabras) son "firmados" generando un hash. Aquellos documentos que tengan un mayor número de hashes en común serán más similares. Por último, con el fin de reducir el número de firmas que es necesario comparar, los autores proponen un método que les permite generar un hash que resume todos los hashes de los conjuntos de palabras que contiene el documento. De esta forma el coste computacional para la detección de documentos idénticos o similares se reduce considerablemente.

  Los resultados de la aplicación sobre 30 millones de páginas de AltaVista permiten concluir que la tercera

parte de los documentos analizados eran idénticos o muy similares a otros presentes en el conjunto de datos de estudio.

En esta tesis doctoral se aborda la optimización del proceso de crawling en base a la detección de contenidos inadecuados, para permitir disponer de más recursos para procesar contenidos de calidad. Concretamente, la arquitectura de crawling propuesta, filtrará las páginas de Spam y Soft-404, y utilizará un sistema de detección de cambios en páginas web en tiempo real para optimizar la calidad y actualidad de los contenidos. En las secciones II.3.1 y II.3.2 se explican las técnicas presentes en la literatura para la detección de páginas de Spam y de Soft-404, respectivamente. En la sección II.4 se analizan las diferentes políticas y métodos presentes en estado del arte para el mantenimiento actualizado de los repositorios.

II.3 Crawling de Contenido Válido

En esta sección se discuten los estudios presentes en el estado del arte, relacionados con la caracterización y detección de aquellos contenidos que un crawler debe filtrar para evitar su procesamiento. Concretamente, se describen las técnicas y métodos de detección existentes para páginas de Spam y de Soft-404. De esta forma, se podrán conocer las limitaciones de las aproximaciones actuales, y así lograr mejorar su eficacia y rendimiento, para poder adaptarlas a los sistemas de crawling.

II.3.1 Web Spam

Existen diversas definiciones de Web Spam en el estado del arte, sin embargo, una de las más claras y simples es la realizada por Google [1].

"Cuando tratemos de decidir si una página Web es Spam, debemos hacernos esta pregunta: ¿Después de eliminar el contenido copiado, la publicidad y los enlaces a páginas externas, nos queda algo de valor en la página? Si la respuesta es no, la página probablemente es Spam."

Un ejemplo de Web Spam es el mostrado en la Figura II.3. Entre su contenido se observa que existen multitud de palabras clave repetidas ("car", "cheap", "discount", "drive", etc.), así como bloques de enlaces no relacionados con el contenido de la página, cuya única finalidad es tratar de que el usuario visite una nueva página web. Desde el punto de vista de un buscador esta página es relevante ante búsquedas de alquiler de coches y alquiler de coches baratos, cuando en realidad no ofrece dicho servicio. Esto es debido a que tras un análisis automático el contenido de está página es relevante. En resumen, los spammers logran mejorar el ranking de la página en determinadas búsquedas y aumentar el tráfico, lo cual implicará un aumento de los beneficios por publicidad.

[1] http://www.seobook.com/official-mahalo-com-spam-according-googles-internal-spam-documents

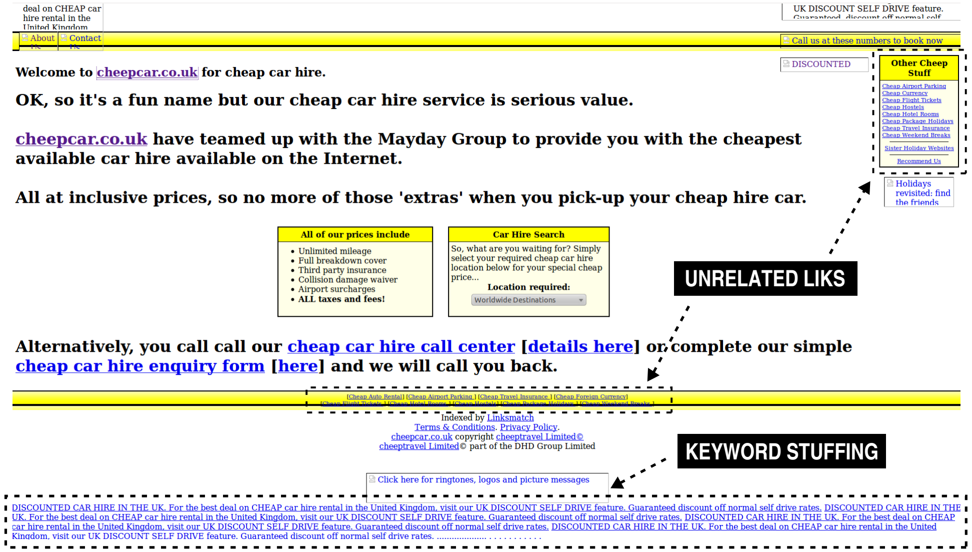


Figura II.3: Ejemplo de página Web de Spam

En la actualidad son múltiples las motivaciones para generar contenido Spam, aunque la más evidente es el obtener un beneficio económico. Entre éstas, se pueden destacar las siguientes:

- Mejora del PageRank para obtener un mejor ranking y de esta forma poder aumentar sus beneficios económicos mediante publicidad o venta de enlaces.
- Dañar a terceras partes, usualmente compañías competidoras.

Ya en el año 2002 Henzinger *et al.* hicieron patente en [HMS02] la importancia del fenómeno de Web Spam en detrimento de la calidad de los resultados de los buscadores. En el año 2006, Ntoulas *et al.* realizaron un estudio para analizar el impacto del Web Spam en la Web ([NM06]). En dicho estudio los autores crearon, de forma aleatoria, un dataset compuesto por páginas de los dominios de mayor relevancia (biz, us, com, de, net, etc.). El análisis y clasificación de las páginas en Spam o no-Spam, se hizo de forma manual. Según dicho estudio, y como vemos en la Figura II.4, en torno al 70% de las páginas del dominio .biz son Web Spam, seguido del 35% de .us. Los dominios .com, .de, .net tienen entre el 15% y el 20% de Web Spam, lo cual aún siendo menor, no deja de ser un alto porcentaje. En el lado opuesto a estos datos se encuentra el dominio .edu, del cual no se observó que alguna de sus páginas fuera Web Spam. En el mismo estudio se analizó la relación entre el lenguaje de la página y el Web Spam. Se observó que aproximadamente el 25% de las páginas en Francés o el 22% de páginas en Alemán son páginas de Web Spam.

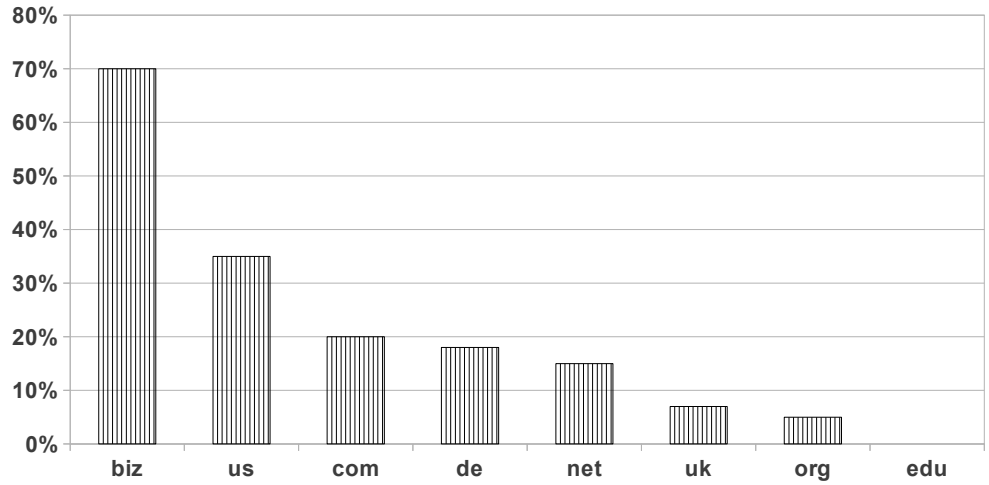


Figura II.4: Presencia de Web Spam en dominios de alto nivel

En ambos estudios queda patente el gran porcentaje de Web Spam existente frente al total de la Web, y por lo tanto, la importancia de una detección adecuada que evite su procesamiento por parte de un sistema de crawling. Por este motivo, todo crawler eficiente debe evitar este tipo de páginas porque, por una parte, empeoran la calidad de los contenidos procesados, y por otra parte, hacen que se dediquen recursos a ellas en lugar de aprovecharlos en alcanzar una mayor fracción del contenido válido de la Web.

Tipos de Web Spam

Existen diferentes tipos de Spam, y diferentes clasificaciones para ellos. Sin embargo, las clasificaciones más relevantes del estado del arte, llevadas a cabo por Gyongyi y García-Molina [GGM04], y por Najork [Naj09], apuntan que las principales clases de Web Spam son: content Spam, cloaking y redirection Spam, click Spam y link Spam. A continuación se describen cada uno de ellos:

- Content Spam: es una técnica de Spam utilizada para intentar mejorar el ranking de una página, y por lo tanto la probabilidad de que sea devuelta como resultado a una búsqueda (en las primeras posiciones, las que normalmente revisa un humano). Para lograrlo, la idea de este tipo de Spam es añadir al contenido de la página palabras clave de la búsqueda ante la que interesa que sea devuelta. Estas palabras clave suelen ser uno o varios términos de consultas.

  Una parte de este tipo de Spam es ineficaz, ya que los motores de búsqueda usan técnicas de análisis de lenguaje para detectar contenido aleatorio y no relacionado, como serían páginas web llenas de palabras

clave. Sin embargo, existe otro tipo de content Spam más sofisticado que únicamente contiene un conjunto pequeño de palabras clave, y el resto del contenido es generado aleatoriamente, copiando el contenido de uno o varios sitios web legítimos, lo que dificulta su detección.

- Cloaking y redirection Spam: hace referencia a la técnica de Spam que permite mostrar contenidos diferentes al usuario y al motor de búsqueda haciendo uso de scripts. Debido al hecho de que las direcciones IP y los User-Agent de los crawlers son públicas, los spammers pueden detectar si es un crawler el que está realizando la petición o se trata de un usuario.

- Click Spam: esta técnica consiste en ejecutar consultas contra los buscadores y realizar clicks sobre determinadas páginas con el fin de simular un interés real por parte del usuario. Los motores de búsqueda usan ciertos algoritmos para analizar los logs de clicks y detectar ciertas anomalías.

- Link Spam: es aquel tipo de Spam basado en la incrustación de enlaces malintencionados entre páginas y la creación de nuevas páginas con esa finalidad. De esta forma, trata de mejorar la relevancia de dicho sitio web. Los enlaces entre páginas y sitios web se hacen o bien desde páginas controladas por los spammers, o desde páginas, como foros o blogs, cuyos propietarios desconocen dichos enlaces.

  En otras ocasiones, los spammers compran dominios con gran cantidad de tráfico (usualmente porque sus nombres son relevantes o similares a otros de importantes sitios web), para generar enlaces desde ellos y lograr subir la relevancia de otros sitios web de Spam, construyendo lo que se denomina una "link farm".

Por último, es interesante destacar el estudio realizado por Castillo y Davison [CD11], donde realizan un extenso análisis sobre los diferentes tipos de Spam y las posibilidades existentes para su detección.

Una vez presentados los diferentes tipos de Spam existentes en la Web, en las siguientes secciones se comentan los principales trabajos existentes en la literatura que abordan la detección del mismo. Se distinguen 4 aproximaciones diferentes: basado en contenido, en análisis de enlaces, en análisis de de scripts o en análisis de la sesión HTTP.

Detección basada en análisis del contenido

Este conjunto de técnicas basan su detección en el análisis del contenido de las páginas web. Entre los artículos que siguen esta aproximación, destaca el presentado por Fetterly *et al.* [FMN04] en el año 2004. El estudio de Fetterly *et al.*, trata la detección de Web Spam basándose en la idea de que la mayor parte de las páginas de Spam están generadas de forman automática. Por ello, las características de contenido de una página legítima y una página de Spam serán diferentes. Como heurísticas de detección los autores proponen el estudio de: a) la longitud del nombre de los hostnames simbólicos, b) el número de hostnames diferentes que apuntan a una misma IP, c) la distribución de los enlaces de salida y de entrada, d) la variación del número de palabras

de una página web de un único servidor ante diferentes peticiones (ya que son contenidos autogenerados, usualmente el contenido es diferente, pero el número de palabras es similar) y e) la evolución del número medio de modificaciones realizadas en las páginas web de un mismo servidor. En 2005, Fetterly *et al.* en [FMN05], presentan un trabajo sobre la detección de Spam en base al análisis del uso de la técnica "cut and paste".

Otros artículos orientados a la detección de este tipo de Spam, son los presentados por Ntoulas *et al.* [NM06] y Gonzalez *et al.* [GJC09]. Las técnicas presentadas en ambos estudios serán analizadas en detalle en la parte final de esta sección.

Otra variante de las técnicas de detección basadas en contenido, es la detección de Spam por medio del análisis de las características lingüísticas del contenido. Un estudio que sigue dicha aproximación, es el presentado por Piskorski *et al.* [PSW08]. Las principales características que los autores proponen para ayudar a la detección de Web Spam son: relación de nombres usados frente al contenido total, relación de verbos utilizados y contenido total, relación de pronombre y contenido total, relación de adjetivos y adverbios frente al contenido total, ratio de verbos modales y verbos normales, relación del uso de la primera persona frente a otro tipo pronombres personales y relación del número de palabras que comienzan con mayúsculas frente al número total de palabras.

Como último comentario sobre los artículos relacionados con la aproximación basada en contenido, destacar que técnicas similares a las comentadas sobre la detección de este tipo de Spam, son también usadas para la detección de Email Spam unidas al uso de clasificadores (por ejemplo C4.5), como propusieron Hidalgo [Hid02] y Sahami *et al.* [SDHH98].

De los citados estudios, los mejores resultados han sido obtenidos por Ntoulas *et al.* [NM06] y Gonzalez *et al.* [GJC09]. A continuación se detallan las heurísticas presentadas por Ntoulas *et al.* y Gonzalez *et al.* para la detección de páginas Web Spam en base al contenido de las mismas:

- Número de palabras en una página: la técnica más extendida para hacer Web Spam se conoce con el nombre de "keyword stuffing" y consiste en incluir en el contenido de la página palabras populares. En muchas ocasiones el contenido de palabras populares supera, o constituye la totalidad del contenido de la página. Como método para su detección proponen medir el número de palabras, excluyendo etiquetas HTML, del contenido de una página.

- Número de palabras en el título: partiendo de la idea expuesta previamente, una técnica similar sería hacer "keyword stuffing" en el título de la página. En este caso, esta heurística relacionará el número de palabras en el título con la probabilidad de ser Web Spam. A mayor número de palabras mayor probabilidad de tratarse de una página de Spam.

- Longitud de las palabras: esta técnica mide la longitud media de las palabras del contenido, ya que se ha observado una nueva técnica de "keyword stuffing" que se basa en unir palabras pequeñas para formar

palabras de mayor longitud ("freepictures", "freemp3", etc.). Este método, lógicamente, no tiene en cuenta la longitud de las etiquetas HTML del código.

- Palabras en el texto de los enlaces: técnica de Web Spam que se basa en la idea de que para muchos motores de búsqueda el contenido del "anchor" describe a la página web destino. Debido a este hecho los spammers han usado el "anchor" para describir enlaces que apuntaban a páginas que en realidad era Web Spam. La técnica propone medir la relación entre el número total de palabras de una página (excluyendo etiquetas HTML) y el número total de palabras incluidas en el "anchor".

- Relación del contenido visible: una forma diferente de hacer "keyword stuffing" es agregar esas keywords en zonas que no serán mostradas al usuario en su explorador, como por ejemplo comentarios dentro del código HTML, en elementos Alt, Meta, etc. Un método que permita medir el contenido que realmente ve el usuario, es analizar la relación entre el contenido total, sin etiquetas HTML, y el contenido total de la página web. Otro método similar consiste en plantear la misma relación, pero considerar exclusivamente como contenido oculto aquel que está incluido en el atributo Alt, dentro de la etiqueta Img, y aquel que aparece como "hidden" en la etiqueta Input.

- Ratio de compresión: técnica de detección basada en la idea de que los spammers tratan de mejorar el ranking de sus páginas repitiendo decenas de veces palabras, párrafos o contenidos completos. La técnica consiste en medir el nivel de redundancia del contenido de una página de Spam. Para ello, se analiza el ratio de comprensión, es decir el tamaño, en bytes, de la página web normal frente al tamaño de la página web comprimida. En las páginas web de Spam el ratio de compresión es mayor ya que su nivel de redundancia también es mayor.

- Número de palabras populares: técnica que agrega palabras populares al contenido, ya sean de un lenguaje específico (diccionario) o de queries típicas, con la idea de ser respuesta al mayor número posible de consultas.

  La idea para detectar esta técnica de Spam, es calcular la relación entre las N palabras más comunes y el número total de palabras populares consideradas.

- Fracción de palabras populares: técnica de detección muy similar a la anterior, pero que evita que una página con una única palabra sea considerada Web Spam si esa única palabra es popular. Esta nueva propuesta calcula la relación entre apariciones diferentes de palabras populares y el número total de palabras populares consideradas. Obviamente, tanto este método como el anterior dependen del lenguaje, por lo que deberán ser modificados dependiendo de la situación.

- Probabilidad n-gram independiente: técnica basada en el análisis gramatical y semántico del contenido de la página web. Debido a que las páginas web de Spam son: a) automáticamente generadas o b) parten

de contenido de otras páginas web; su contenido contiene un mayor número de errores gramaticales y semánticos que el contenido de una página web legítima. Sin embargo, las técnicas basadas en análisis gramatical y semántico tienen como problema su alto coste computacional. Para solucionar esto, se han propuesto métodos que tratan de realizar este análisis gramatical y semántico, pero reduciendo el coste su coste computacional asociado. Esta técnica propone dividir el contenido de cada página en n-grams de n palabras consecutivas. Para reducir el coste computacional se supone que cada n-gram es independiente de los demás. La probabilidad de n-gram w_{i+1} ... w_{i+n} comenzando en la palabra i+1 es:

$$P(w_{i+1} \dots w_{i+n}) = \frac{\text{número de ocurrencias de } n\text{-gram}}{\text{número total de } n\text{-grams}}$$

De esta forma la probabilidad de un documento con k n-grams, bajo la asunción de independencia de los n-grams, viene dada por el producto de las probabilidades individuales. Tras normalizar por la longitud del documento tenemos:

$$IndepLH = -\frac{1}{k}\sum_{i=1}^{k-1} logP(w_{i+1} \dots w_{i+n})$$

La relación entre los resultados conseguidos y las páginas con Web Spam, ayudan a la detección de Web Spam.

- Probabilidad n-gram dependiente: Una mejora de la técnica de detección anterior, es no considerar independientes los n-grams. De esta forma la probabilidad de un n-gram condicionada a las $(n-1)$-palabras anteriores es:

$$P(w_n|w_{i+1} \dots w_{i+n-1}) = \frac{P(w_{i+1} \dots w_{i+n})}{P(w_{i+1} \dots w_{i+n-1})}$$

Normalizando del mismo modo que en la sección anterior se obtiene:

$$CondLH = -\frac{1}{k}\sum_{i=1}^{k-1} logP(w_{i+1} \dots w_{i+n-1})$$

Detección basada en el análisis de los enlaces y sus relaciones

Este conjunto de técnicas están orientadas a la detección de un tipo especial de Spam: “Link Spam”.

Wu y Davison [WD05b], proponen un algoritmo basado en los enlaces de salida y entrada comunes entre páginas web. Su aproximación se basa en el hecho de que los sitios web que utilizan link Spam, están altamente conectados con otras páginas y sitios web, las cuales en muchas ocasiones están conectadas a su vez con dichos sitios y páginas web. El algoritmo que proponen parte de un conjunto de páginas web que forman parte de una granja de enlaces. Cada nueva página que aparezca es probable que forme parte de dicha granja de enlaces si diversos enlaces de entrada y salida están conectados con el resto de páginas de la granja de enlaces. En este caso, la página pasará a formar parte del conjunto de páginas que forman la granja de enlaces, y el proceso continuará analizando los enlaces existentes en dicha página web. El proceso terminará en el momento en que no se agreguen nuevas páginas al conjunto.

Amitay *et al.* [ACD+03], presentan un sistema que permite clasificar automáticamente los sitios web en 8 clases definidas. Las clases definidas por los autores son las siguientes: corporación, contenido, búsqueda, directorio, portal, tienda, virtual host y universidad. Para que el sistema propuesto pueda clasificar correctamente un sitio web, debe analizar un conjunto de heurísticas previamente definidas. Las heurísticas definidas por los autores son las siguientes: 1) profundidad media de las páginas, 2) porcentaje de páginas en los niveles más populares, 3) relación del número de páginas en el segundo nivel y en el primer nivel, 4) enlaces de entrada por página, 5) enlaces de salida por página, 6) relación entre enlaces de entrada y enlaces de salida, 7) fracción de enlaces de entrada que provienen de páginas de alto nivel, 8) enlaces de salida por página web hoja, 9) nivel de profundidad medio de donde provienen los enlaces de entrada, 10) nivel de profundidad medio de las páginas destino de los enlaces de salida, 11) diferencia entre 10) y 9), 12) porcentaje del nivel de entrada para el nivel más popular, 13) porcentaje de enlaces de salida en cada uno de los niveles existentes, 14) enlaces cruzados por página, 15) relación entre los enlaces de navegación y los enlaces semánticos y 16) número de enlaces de entrada en cada una de las páginas del primer nivel, en relación al número total de páginas del sitio web.

Amitay *et al.* proponen el uso de su aproximación para la detección de sitios web que formen parte de una granja de enlaces.

Gyongyi *et al.* [GGMP04], proponen una técnica para lograr diferenciar entre páginas de Spam y páginas normales. El sistema propuesto, denominado TrustRank, parte de un conjunto de página legítimas, previamente seleccionadas por un grupo de expertos. Analizando el grafo de enlaces entre las páginas de dicho conjunto, permite descubrir páginas web que probablemente sean legítimas.

Por último, Becchetti *et al.* [BCD+06], presentaron un sistema que combina algunas de las ideas anteriormente explicadas. En el estudio presentaron más de 100 características con las que se propone la detección de Spam. Las citadas heurísticas se basan en: a) grado de correlación, b) número de vecinos, c) propagación del ranking a través de los enlaces, d) TrustRank, etc. Todas estas métricas se combinaron utilizando diferentes algoritmos de clasificación.

En resumen, se ha observado que a pesar existir diferentes aproximaciones para la detección de Spam

basándose en el análisis de la estructura de la Web, estas aproximaciones tienen ciertas desventajas. Muchas de ellas son supervisadas y requieren el uso de expertos para crear grupos de páginas legítimas. Además, este tipo de sistemas necesitan analizar el grafo previamente a decidir si es Spam, o no. Esto es un gran inconveniente, ya que el crawler ya habrá procesado dicha página con el consecuente gasto de recursos que esto supone. Por último, el uso de este tipo de sistema requiere un alto coste computacional, ya que requiere un análisis de las relaciones entre las páginas y entre sus características.

Detección basada en el análisis de scripts

Este conjunto de técnicas están orientadas a la detección de cloaking. A diferencia de los casos anteriores, en los cuales los diferentes estudios partían de una misma idea, existen diversas aproximaciones. Gyöngyi y García-Molina en [GGM04] explican las técnicas de cloaking existentes.

Una de las aproximaciones existentes para la detección de este tipo de páginas es la propuesta por Wu y Davison [WD05a], quienes proponen detectar páginas con cloaking, analizando las diferencias (términos y enlaces) entre tres copias descargadas de una misma página.

Otra de las técnicas existentes para la detección de cloaking es la patentada por Najork [Naj]. En ella, el autor propone el uso de una aplicación en el navegador del usuario, que firme el contenido de las páginas web que el usuario visita partiendo de los resultados de los motores de búsqueda. El inconveniente de esta aproximación, es que actualmente el contenido de las páginas web cambia demasiado rápido.

Por último, queremos destacar la aproximación propuesta por Chellapilla y Maykov [CM07]. En su estudio explican las diferentes técnicas usadas por los spammers para lograr realizar cloaking, dificultando el proceso de detección por parte de los motores de búsqueda. Entre las técnicas que describen se encuentran: redirecciones JavaScript en texto plano, manipulación de cadenas de texto y uso de la función Eval, uso de la función Unescape, función Decode, inyección de scripts, inyección de HTML y formularios, e inyección de eventos. En base a los resultados que obtienen, proponen el uso de motores y parsers de JavaScript que permitan la detección de dicho tipo de técnicas.

Esta aproximación tiene como inconveniente principal su alto coste computacional (por ejemplo, motores y parsers de JavaScript), lo que provoca que en sistemas de alto rendimiento como son los crawlers, no puedan ser usados.

Detección basada en el análisis de la sesión HTTP

Este conjunto de técnicas de detección de Spam se basa en el análisis de las cabeceras HTTP. Esta técnica fue presentada por Webb *et al.* en [WCP07] y [WCP08].

En dichos artículos los autores demuestran que las cabeceras HTTP de la mayor parte de las respuestas de

páginas y sitios web de Spam son idénticas o muy similares. Entre las cabeceras HTTP que analizan están: Content-Type, Server, Connection, Link, Expires, Pragma, Refresh, etc. La técnicas se dividen en dos procesos: a) generación de "features" y b) uso de clasificadores.

Para la generación de las "features" a partir de las cabeceras de la sesión HTTP, los autores generan tres tipos de representación, frases, n-gramas y tokens. Por una parte generan una frase uniendo todos los valores de las cabeceras HTTP. Para la creación de los n-gramas, se separan los valores de las cabeceras usando signos de puntuación y espacios en blanco, creando bigramas y trigramas. Finalmente, agregan el nombre de la cabecera a cada uno de los valores generados para evitar posibles confusiones con valores idénticos de diferentes cabeceras.

La gran desventaja de este método es que es fácilmente evitable por parte de los spammers. Únicamente deberán aleatorizar, o copiar de páginas legítimas, los valores de las cabeceras HTTP desde su servidor web. Además, muchas de las páginas y sitios web de Spam están alojadas en servidores de empresas de hosting, por lo que los valores de sus cabeceras HTTP serán idénticos a otros sitios web. Esto implica que muchas de las páginas de Spam detectadas y muchos de sus valores provocarán falsos positivos de otras páginas legítimas alojadas en los mismos servidores web.

Conclusiones sobre las técnicas de Web Spam existentes

Los estudios sobre técnicas de detección de Web Spam existentes en el estado del arte presentan múltiples problemas y desventajas. Estos inconvenientes aparecen tanto en la técnica de detección, como en su aplicación dentro de un sistema de crawling y se enumeran a continuación:

- Las técnicas presentadas se centran en un tipo de Web Spam en particular, sin tratar de abordar una detección completa del Web Spam.
- Las técnicas basadas en análisis de enlaces necesitan tener el grafo de los enlaces de una página previamente a decidir si es Spam, o no. Esto es un gran inconveniente, ya que el crawler ya habrá procesado dicha página con el consecuente gasto de recursos que esto supone.
- Para la detección de cloaking proponen el uso de sistemas como motores y parsers de JavaScript lo cual supone un alto coste computacional como para ser utilizado en un sistema de alto rendimiento como son los crawlers.
- Las técnicas de detección basadas en el análisis de las cabeceras HTTP tiene la desventaja que son fácilmente evitables.
- Otras de las aproximaciones presentadas son supervisadas y requieren el uso de expertos para crear grupos de páginas legítimas.

- Por último, los estudios presentados no han sido realizados para su aplicación a sistemas de crawling, por lo que no han tenido en cuenta cuestiones de rendimiento. De igual forma, ninguno de ellos describe cómo podrían ser adaptados a una arquitectura genérica de crawling.

Todas estas deficiencias dejan patente la necesidad de un nuevo método de detección que permita su utilización en sistemas de crawling.

II.3.2 Páginas Soft-404

Además de las páginas de Spam, existen otro tipo de páginas web, denominadas páginas Soft-404, cuyo contenido es inadecuado, y no debe ser procesado por los sistemas de crawling. Dichas páginas son enviadas por algunos servidores web, cuando estos reciben una petición a un recurso web desconocido, y realizan una redirección hacia otro contenido sin enviar el código de error HTTP correspondiente. De la misma forma que el Web Spam, como se demostrará en el capítulo VI, la presencia de este tipo de páginas es significativa en la Web.

En el conocido libro "Modern Information Retrieval" de Baeza-Yates y Ribeiro-Neto [BY11], analizan y destacan la importancia de este tipo de páginas en la Web, y el inconveniente que presentan para los crawlers. Por otro lado, en el estudio realizado por Bar-Yossef *et al.* [BYBKT04], indican que el 29% de los enlaces rotos de la Web apuntan a este tipo de páginas. Estos datos demuestran que las páginas Soft-404 constituyen un problema significativo para la Web que debe ser considerado por los crawlers.

Sin embargo, a pesar de ser un problema importante y significativo en la Web, no existen muchos trabajos en la literatura, que traten su análisis y detección. Concretamente, existen dos artículos que estudian este tipo de páginas, el realizado por Bar-Yossef *et al.* [BYBKT04] y el llevado a cabo por Lee *et al.* [LKK+09]. A continuación se analizan en detalle cada uno de ellos.

Detección basada en peticiones HTTP

Esta aproximación fue presentada en el año 2009 por Bar-Yossef *et al.* en [BYBKT04]. El artículo trataba de estudiar la decadencia de la Web. Para ello, deciden medir el porcentaje de enlaces rotos ("dead links"). Sin embargo, se encontraron con que gran parte de esos enlaces son enlaces a páginas Soft-404, lo cual dificultaba la detección de páginas que ya no existen. Bar-Yossef *et al.* definen un algoritmo para la detección de páginas Soft-404, basado en la realización de dos peticiones HTTP, una a la página que desea ser analizada y otra a una página aleatoria del mismo dominio. El pseudocódigo del algoritmo es el que se muestra a continuación:

```
EsSoft-404( URL u )

URLFinal_u, contenido_u, numRedirecciones_u, error <- Recuperar( u )
SI ( error ) {
    No-Soft-404
}

r <- DirectorioPadre( u ) + TextoAleatorio(25)
URLFinal_r, contenido_r, numRedirecciones_r, error <- Recuperar( r )
SI ( error ) {
    No-Soft-404
}

SI ( u ES RAÍZ de u.host ) {
    No-Soft-404
}
SI ( numRedirecciones_u != numRedirecciones_r ) {
    No-Soft-404
}
SI ( URLFinal_u = URLFinal_r ) {
    Soft-404
}
SI ( CasiIdenticos( contenido_u, contenido_r ) {
    Soft-404
}
SINO No-Soft-404
```

La función *Recuperar*, es la responsable de descargar la página web solicitada y devolver la URL final, el contenido de dicha URL, el número de redirecciones realizadas hasta llegar a la URL final, o un error en caso de que éste se produzca. La función *TextoAleatorio* genera una cadena alfanumérica de 25 caracteres generados de forma aleatoria. De esta forma los autores se aseguran que la segunda página solicitada es altamente probable que no exista. Por último, la función *CasiIdenticos*, se encarga de comparar el contenido de dos páginas web.

El algoritmo descrito, comienza ejecutando la función Recuperar sobre la página web que se desea conocer si es Soft-404 (u). En caso de error se considera que dicha página no es Soft-404. Posteriormente, se hace lo mismo con una página, r, generada aleatoriamente. En caso de que la página u sea raíz del $u.host$, o que el

número de redirecciones realizadas al recuperar u y r, sea diferente, se considera que la página web u, no es Soft-404. Tras esto, si la página final recuperada es la misma, o su contenido es casi idéntico, se considera que la página es Soft-404. En cualquier otro caso, se considera que la página no es Soft-404.

Una vez analizada la solución propuesta por Bar-Yossef *et al.*, se presentan a continuación el conjunto de problemas que tiene el uso de dicho método:

- El funcionamiento del algoritmo necesita que el servidor web realice redirecciones HTTP. En cualquier otro escenario, el algoritmo no detectará nunca páginas Soft-404.
- La suposición de que las páginas raíz de un sitio web nunca redirigen a páginas Soft-404 es muy arriesgada, ya que como se verá en los experimentos realizados en el capítulo VI, esta situación se da en multitud de ocasiones.
- El algoritmo mostrado, no detecta páginas de parking[2].
- En multitud de ocasiones, sobre todo en páginas de Redirect Spam, el número de redirecciones realizadas en cada petición difiere.
- La comparación de contenido de páginas web no es fiable, y su realización requiere un alto uso de recursos.
- Por último, el algoritmo presentado, tiene un problema de rendimiento, ya que tiene que realizar dos peticiones HTTP en la mayoría de los casos. Por ello, el crawler necesitará usar más recursos y el servidor web que aloja el sitio web tendrá que atender el doble de peticiones, lo que supone un mayor consumo de recursos.

Detección basada en análisis de redirecciones

La segunda aproximación presente en el estado del arte es la presentada por Lee *et al.* en [LKK+09]. En dicho artículo, los autores se centran en la detección de este tipo de página porque consideran que el 25% de los considerados como enlaces muertos ("dead links") están formados por este tipo de páginas. Debido a que son conscientes de que el trabajo previo tiene ciertas limitaciones, proponen una nueva medida para detectar este tipo de páginas web. El método propuesto consiste en detectar páginas Soft-404, analizando los logs de redirecciones durante el proceso de crawling.

Para poder explicar las heurísticas que se proponen en el artículo vamos a realizar las siguientes definiciones:

- Sea U el conjunto de URLs original, y V el conjunto de URLs destino.

[2]Páginas web que forman parte de dominios registrados (usualmente con nombres relevantes), cuyo único propósito es colocar enlaces patrocinados que generan ingresos por publicidad.

- Sea la función $\not f : U \rightarrow V$, donde una redirección desde $u \epsilon U$ hacia $v \epsilon V$, es denotada por $v = \not f(u)$.
- Sea L el log de redirecciones del crawler donde L es el conjunto de pares $\{(u, \not f(u))\, u \epsilon U\}$, recuperados por el sistema de crawling.

Las heurísticas definidas por los autores son las siguientes:

- Lee *et al.* proponen que es común que en un host exista una página v destino, utilizada para realizar muchas de las redirecciones en ese host. Para una página destino $v \epsilon V$, sea I_v el número de redirecciones en L cuyo destino es v, la probabilidad de que una página web sea una página Soft-404, aumenta cuanto mayor es I_v.

  Sin embargo, esta heurística tiene un problema importante, ya que existen multitud de páginas $v \epsilon V$, con un alto I_v, que no son páginas Soft-404. Un claro contraejemplo es la página web de Google (http://www.google.com).

- Para un host h, sea N_h el número de URLs originales en h, y M_h el número de URLs destino desde h. Entonces, N_h / M_h es el número medio de URLs originales en h que comparten la misma redirección destino. Es decir, una redirección de un host h, con un alto N_h / M_h, es probable que sea una página Soft-404.

  La base tomada para definir esta heurística es arriesgada, ya que supone que si un host h tiene muchas redirecciones y pocas URLs originales, es probable que una redirección v vaya a una página Soft-404. Existen multitud de páginas con pocas URLs que redirigen a otras páginas de otro host cuyo contenido es real y legítimo, y que en ningún caso se trata de una página Soft-404.

- Por último, los autores han detectado que servidores de publicidad generan un gran número de redirecciones desde hosts distintos. Debido a esto, la primera heurística podría fallar. Para ello, definen H_h, como el número de hosts destino alcanzados por las redirecciones del host h.

  Es decir, $H_h = \mid \{host(y) : (x, y)\, \epsilon L,\, host(x) = h\} \mid$.

  De esta forma, una redirección desde el host h será legitima si su H_h es alto. La limitación de la heurística propuesta es que es demasiado sencilla y existen multitud de casos en los que fallará. El hecho de que un host tenga muchas redirecciones a diferentes hosts, no implica que cualquier otra redirección tenga como destino una página Soft-404.

En base a las heurísticas definidas previamente, Lee *et al.* proponen la siguiente fórmula:

$$\mu(u, v) \triangleq k_1 log_{10} I_v + k_2 log_{10} \frac{N_{host(u)}}{M_{host(u)}} + k_3 log_{10} \frac{1}{H_{host(u)}}$$

Cuanto mayor sea la puntuación obtenida mayor será la probabilidad de que dicha página sea Soft-404. Las constantes k_1, k_2 y k_3, son pesos, no negativos, que varían la importancia de cada heurística. Como se ha indicado previamente, las heurísticas propuestas contienen una serie de problemas que con el uso de los pesos, podrían ser parcialmente mitigados. Sin embargo, los autores han usado $k = 1$ por lo que los problemas de cada una de las heurísticas siguen presentes y no se compensa el error de una de ellas con el acierto de las otras.

Por otro lado, el trabajo de Lee *et al.*, no realiza la detección de las páginas Soft-404 en tiempo real. Esto provoca que el sistema de crawling procese todas esas páginas, consumiendo gran cantidad de recursos, y posteriormente tratará de detectarlas. Esto es debido a que el sistema de Lee *et al.* realiza un proceso de crawling de los hosts que desea analizar como paso previo a decidir si es o no una página Soft-404. Este problema es un gran inconveniente desde el punto de vista del rendimiento del sistema.

Otra limitación importante del sistema es que necesita que las redirecciones sean HTTP. Si el servidor web las hace de forma transparente al crawler, el método propuesto nunca las detectará. Por último, las heurísticas propuestas son excesivamente simples, por lo que existen multitud de escenarios en los que pueden fallar.

En resumen, se observa que el estudio presentado por Bar-Yossef *et al.* en [BYBKT04], y el realizado por Lee *et al.* [LKK+09], contienen diversas limitaciones y desventajas, tanto en eficacia como en eficiencia. Un sistema de crawling eficiente deberá tratar de mejorar dichas limitaciones. Por ello, en este trabajo (capítulo VI), se proponen un conjunto de novedosas técnicas para la detección de las páginas Soft-404.

II.4 Actualización de Contenido en Crawlers

Debido a que el contenido de la Web está cambiando continuamente, cuando un crawler "concluye" un recorrido de la Web, debe comenzar el siguiente con el objetivo de mantener lo más actualizados posibles los repositorios. La mejora de la actualización de los repositorios es crítica, ya que de ellos dependerá la calidad de los resultados mostrados a los usuarios. En este punto se presenta otro de los grandes desafíos con los que debe lidiar un sistema de crawling: no todas las páginas cambian con la misma frecuencia. En algunos casos puede ser suficiente un crawling total periódico para mantener los repositorios actualizados, pero para otros contenidos altamente variables el tiempo que puede llevar completar un recorrido completo para comenzar con el siguiente puede no ser adecuado para garantizar una calidad en las búsquedas de los usuarios.

En esta sección se presenta el estado del arte relacionado con la mejora del proceso de actualización por parte de los sistemas de crawling. Por un lado, se discutirán los mecanismos y técnicas existentes, que ayudan a los crawlers en este proceso, y por otro se realiza un análisis de los artículos más relevantes presentes en la literatura.

II.4.1 Mecanismos de actualización de contenidos

Existen en la Web un conjunto de mecanismos creados para diferentes fines que pueden ser usados para conocer cuándo un documento o recurso ha sido modificado. A continuación se explican cada uno de ellos, discutiendo sus ventajas e inconvenientes:

- El protocolo HTTP, proporciona la cabecera Last-Modified, que indica cuándo ha sido la última modificación de un recurso. Sin embargo, en muchos casos la información de esta cabecera no es fiable debido a que el servidor web no ha implementado correctamente el protocolo HTTP, o porque el servidor web hace uso de un administrador de contenidos que genera dinámicamente las páginas web en el momento en el que son solicitadas. Además, el sistema de crawling requiere al menos un nuevo acceso al recurso para detectar si la página web ha cambiado.

- Algunos servidores proporcionan feeds RSS [3], para notificar cuándo ha habido modificaciones en las páginas web. Sin embargo, la mayor parte de los servidores web no los proporcionan y su implantación no es sencilla, pues requiere un coste relativamente alto para los creadores de los sitios.

- Otros servidores web proporcionan información sobre la frecuencia de cambio de sus recursos mediante el uso del protocolo de exclusión de robots ("robots.txt" y "sitemap.xml"). La utilización de este protocolo plantea dos inconvenientes importantes que hacen que tampoco se haya extendido su utilización para facilitar la detección de cambios por parte de los buscadores. Por un lado, es necesario tener acceso al directorio raíz del servidor para que el usuario pueda añadir o modificar información sobre la frecuencia de cambio de sus páginas, en los ficheros "robots.txt" y "sitemap.xml", sin embargo, gran parte de los usuarios no lo tiene. Además, es costoso y difícil de mantener, ya que es necesario conocer previamente la frecuencia de cambio de cada página web para poder publicarla de forma correcta [SZG07].

- Por último, las grandes compañías de motores de búsqueda, tienen acuerdos con importantes empresas de creación de contenidos, que les permiten conocer cuándo una página web ha sido modificada. Sin embargo, la mayor parte de los crawlers y de los sitios web no poseen dichos acuerdos, ya que sólo son posibles en los casos en los que exista un interés mútuo entre el sitio web y el sistema de crawling, como es el caso de grandes multinacionales (por ejemplo Amazon).

Como ya se ha comentado, estas aproximaciones presentan diferentes problemas que no las hacen viables para su utilización por los sistemas de crawling de forma exclusiva para permitir saber cuándo es necesario refrescar el contenido de páginas web. Es por ello, que han surgido múltiples estudios que tratan de solucionar estas deficiencias, y ayudar a los sistemas de crawling a mitigar la desactualización de sus contenidos basados

[3] http://www.rssboard.org/rss-specification

en crawling incremental, detección basada en modelos estadísticos o en análisis de contenido. En las siguientes secciones se comentan estas otras aproximaciones en detalle.

II.4.2 Actualización mediante crawling incremental

De entre los artículos que estudian cómo mejorar el proceso de actualización de los contenidos, destaca el presentado por Cho y García-Molina [CGM00a]. En él los autores presentan una arquitectura de crawling incremental, como medio para mejorar la actualización de los contenidos, en lugar de realizar un nuevo crawling completo. Los autores proponen que el proceso de crawling sea capaz de: a) estimar cuándo una página va a cambiar, en base a su frecuencia de cambio, y b) centrarse en aquellas páginas con una mayor calidad. El algoritmo en el que se basa la arquitectura propuesta es el siguiente:

```
crawlingIncremental( conjuntoTotalURLs, conjuntoLocalURLs )

MIENTRAS ( cierto ) {
    url <- seleccionarURL( conjuntoTotalURLs )
    contenido <- descargar( url )
    SI (url ϵ conjuntoLocalURLs ) {
       actualizar(contenido)
   }
    SI NO {
       URLtemporal <- seleccionarDescartar( conjuntoLocalURLs )
       descartar( URLtemporal )
       guardar( url, contenido)
       conjuntoLocalURLs <- ( conjuntoLocalURLs - URLtemporal) ∪ url
    }
    conjuntoNuevasURLs <- extraerURLs( contenido )
    conjuntoTotalURLs <- conjuntoTotalURLs ∪ conjuntoNuevasURLs
}
```

El uso de crawling incremental es algo totalmente aceptado para su uso en sistemas de crawling general. Sin embargo, el sistema propuesto, tiene el inconveniente de que realmente nunca sabe cuándo ha cambiado una página. Debido a ello, este sistema podría perder gran cantidad de cambios en páginas web, y de recursos, volviendo a visitar páginas web que no han sido modificadas.

II.4.3 Actualización basada en la detección de cambios usando modelos estadísticos

Debido a la necesidad de conocer la frecuencia de cambio de un recurso para poder realizar un crawling incremental eficiente, han surgido diversos estudios en los que tratan de demostrar que la distribución de Poisson puede ser usada para esta estimación de forma bastante fiable [TK98] y [Win72]. Sin embargo, dichos estudios son teóricos, y no contemplan las diferentes circunstancias presentes en la Web.

Cho y García-Molina en [CGM03b], presentan un trabajo donde tratan de estimar la frecuencia de cambio de las páginas web. A los ya comentados inconvenientes que tiene basar la decisión de volver a procesar un recurso en datos estadísticos, los propios autores enumeran tres nuevas desventajas: 1) los crawlers no tienen el historial completo de los cambios de cada página web, 2) los accesos realizados a cada página web suelen ser irregulares, y 3) cada página y sitio web ofrecen un nivel de información diferente lo cual dificulta la elección de los parámetros adecuados para la distribución Poisson. En este artículo los autores definen un estimador para aquellos casos en los que el histórico de cambios de una página web esté incompleto.

Otros estudios relevantes en esta temática realizados por Cho y García-Molina son [CGM00b] y [CGM03a], donde los autores presentan un análisis de las políticas de refresco existentes, y proponen una nueva, que considera la frecuencia de cambios y la importancia de las diferentes páginas web.

Otros artículo relevante, orientado a estimar cuándo se producen los cambios en las páginas web, es el realizado por Fetterly *et al.* [FMNW03]. Tras un extenso análisis, los autores obtienen un conjunto de heurísticas que, en base a sus resultados, permiten mejorar el procesamiento de un crawling incremental. Las heurísticas que proponen son las siguientes:

- Relación entre los dominios de alto nivel y la frecuencia de cambios.
- El tamaño de los documentos es un indicador importante de cambio.
- Los documentos grandes cambian más rápidamente y en mayor extensión.
- Los cambios pasados de una página son importantes para predecir cambios futuros.

Nuevamente, a pesar de que en base a los resultados experimentales demuestran que dichas heurísticas son efectivas, la decisión de volver a procesar una página recae en datos estadísticos, por lo que el crawler no conoce verdaderamente que página o sitio web ha cambiado. Por otro lado, existen estudios en los que se demuestra que las políticas basadas en frecuencia de cambios no son las más adecuadas. Un ejemplo es el artículo de Cho y Ntoulas [CN02], en el cual realizan un estudio sobre el uso de diferentes políticas de actualización basadas en muestreo. Estas técnicas, antes de decidir si volver a procesar un sitio web, se descargan un pequeño conjunto de páginas web del sitio para estimar si éste ha sido modificado. En el estudio se consideran las siguientes políticas:

- Proporcional: el número de modificaciones detectadas en los documentos descargados como muestra es proporcional a los posibles cambios existentes en el sitio web.
- Agresiva: se centra en aquellos sitios web cuyas muestras descargadas tienen más cambios. Primero se descargan todas las páginas del sitio web con más cambios y luego se continua por orden con los siguientes.
- Adaptativa: en base a la evolución de los cambios detectados en los sucesivos procesamientos del sitio web, y de los cambios detectados en las muestras descargadas, se modifica la política para procesar primero un sitio web u otro.

En dicho estudio demuestran que los mejores resultados se obtienen aplicando una política agresiva, y que su complejidad es similar o incluso mejor que la basada exclusivamente en la frecuencia de cambio [CGM03b] [JLW98]. Además, los autores concluyen que el uso de políticas basadas en frecuencia de cambios, no son adecuadas, ya que transcurre un largo período de tiempo hasta que se tiene información suficiente, como para poder estimar cuándo una página web va a cambiar.

II.4.4 Actualización basada en el análisis del contenido

Una aproximación diferente es la propuesta por Kharazmi *et al.* [KNA09], que trata de mejorar la actualización de los contenidos utilizando técnicas de data mining. La arquitectura de crawling que proponen contendría un componente encargado de recuperar información adicional durante el proceso de crawling. El sistema no comenzaría con una política predefinida, sino que aprendería la política más adecuada durante el proceso de crawling.

El primer paso que los autores proponen para determinar cuándo una página debe ser actualizada, es definir a qué clase de página pertenece. A pesar de existir múltiples aproximaciones para determinar la clase de una página web, los autores proponen extraer las palabras del contenido (excluyendo etiquetas HTML, stop words, etc.), y compararlas con las extraídas de otras páginas. Aquellas páginas con mayor número de palabras en común, pertenecerán a la misma clase. El siguiente paso es determinar cuándo se debe actualizar cada clase. Para ello, el sistema recuperará información estadística (clase de la página web, fecha, número de enlaces, tipos de enlaces, etc.), durante el proceso de crawling. Esta información será usada por algoritmos de data mining que ayudarán al sistema a obtener reglas y modelos que ayuden a estimar la frecuencia de cambio de cada clase.

Olston y Pandey [OP08], proponen analizar el contenido para poder distinguir entre contenido efímero y persistente para, de esta forma mejorar, la actualización de los contenidos indexados. La política de refresco que describen trata de maximizar el valor de $informacion\ x\ tiempo$. De esta forma la información almacenada en el sistema será más actual, y el crawler tendrá que usar un menor número de recursos, ya que los recursos

a actualizar son menos. Para ello, proponen comparar los contenidos entre diversas copias de cada página web con el fin de detectar aquellos recursos que cambian más que otros. La desventaja de este sistema es doble: por un lado es evidente que lo que a día de hoy es persistente, mañana puede dejar de serlo; por otro, el sistema necesitaría gran cantidad de recursos para permitir almacenar múltiples versiones de una misma página, y de esta forma poder determinar fielmente qué partes y qué páginas son persistentes.

En el estudio presentado por Baeza-Yates y Castillo [ByC02], proponen una arquitectura de crawling eficiente analizando el contenido procesado. Para ello, los autores proponen balancear el volumen total de datos recuperados, para mejorar la calidad y actualización de los documentos. La política que usa dicho crawler se basa en una fórmula que asigna una puntuación a cada página, entre 0 y 1. Cuanto mayor sea la puntuación obtenida mejor será la página. La fórmula propuesta está basada en las siguientes características de la página:

- Análisis de enlaces: PageRank [PBMW98], concentradores y autoridades [Kle99], popularidad.
- Análisis del contenido: comparación entre el contenido de la página y la query o conjunto de queries con las que dicha página puede ser recuperada.
- Patrones de acceso: analizar la importancia de dicha página para los usuarios.
- Análisis del sitio web: analizar las características anteriores para el sitio web que contiene la página.

Tras haber analizado las diferentes aproximaciones existentes para la detección de cambios en páginas web y mantener actualizados sus contenidos, se han detectado numerosas deficiencias que dificultan su utilización de forma satisfactoria en sistemas de crawling, y se enumeran continuación:

- El crawling incremental es una buena base de partida, pero las aproximaciones existentes carecen de un método que permita conocer cuándo una página web ha sido modificada. Debido a ello, los sistemas actuales son ineficientes ya que consumen recursos procesando páginas que no han sido modificadas.
- Las aproximaciones basadas en datos estadísticos, como la frecuencia de cambio o la longevidad del contenido, para crear modelos que permiten detectar cambios en páginas web, no son sistemas fiables. Estos sistemas no indican cuándo ha sido modificado un documento web, sino cuándo se estima que ha podido cambiar.
- Muchos de los sistemas existentes requieren un estudio previo, analizando durante bastante tiempo los datos y contenidos de una página web para poder estimar con cierta exactitud cuándo va a ser modificada.

Por último, destacar el artículo presentado por Lewandowski [Lew08], en el cual estudia la situación actual de los principales crawlers y buscadores, en relación a sus procesos de actualización de contenidos.

El estudio, realizado durante tres años consecutivos (2005, 2006 y 2007), concluye que: a) existen páginas que son actualizadas diariamente y páginas que se actualizan raramente, y b) los motores de búsqueda no parecen tener una política definida para saber cuándo deben volver a visitar una página.

Debido a estas razones y a los problemas identificados en los métodos existentes, es necesaria la creación de un método que permita la detección de cambios en páginas web más preciso. Idealmente, en tiempo real, para evitar el procesamiento de páginas que no han cambiado, a la vez que se minimice el tiempo que los contenidos están desactualizados. Conociendo el momento exacto de modificación de una página y su relevancia, el sistema de crawling puede calcular el momento adecuado para refrescar su contenido, minimizando el uso de recursos.

II.5 Discusión y Conclusiones

En este capítulo se han estudiado los diferentes trabajos sobre la Web, sus desafíos, y las soluciones existentes para lograr un procesamiento eficiente de la Web. Los principales objetivos del presente estudio del estado del arte son:

- Caracterizar la Web y su estructura a partir de los resultados obtenidos por los estudios existentes hasta el momento.
- Mostrar las soluciones existentes en la literatura para lograr construir un sistema de crawling eficiente.
- Analizar las aproximaciones presentes en el estado del arte para detectar páginas de Web Spam.
- Estudiar las técnicas y métodos existentes en la literatura relacionada para caracterizar y detectar páginas Soft-404.
- Conocer las técnicas y políticas definidas en el estado del arte para el proceso de actualización de los contenidos ya procesados por los crawlers.

En lo que respecta al primer objetivo, se ha definido la estructura de la Web y se han presentado multitud de estudios que la analizan. Los estudios presentados permiten conocer ciertas características puntuales de la Web, principalmente a nivel nacional. Sin embargo, dichos estudios no son suficientes para la presente tesis, ya que no se centran en las características adecuadas, y no lo hacen durante períodos de tiempo consecutivos, que permitan estudiar su evolución y por lo tanto extraer conclusiones. Por ello el presente trabajo incluye en el capítulo III, dos novedosos estudios. Por un lado, se ha realizado un estudio sobre el tratamiento de los sistemas de crawling de la Web Oculta del lado cliente. Por otro, se ha realizado un extenso estudio sobre la Web Global y Española, desde 2009 hasta 2011, analizando aquellas características que son relevantes desde el punto de vista de los buscadores y crawlers. Este estudio ha permitido proponer una serie de políticas de

actuación para mejorar el rendimiento de los sistemas de crawling y buscadores web. El conocimiento de la Web es la base para la realización de un crawling eficiente. Por ello, los estudios realizados, han sido la base para la creación de la arquitectura de crawling que se propone en la presente tesis.

El segundo objetivo se ha analizado en la sección II.2. En esta sección se ha descrito a alto nivel el proceso de crawling de la Web, sus desafíos y los trabajos existentes que estudian el procesamiento eficiente de la Web. Tal y como se explicó anteriormente, existen multitud de desafíos, sin embargo, se pueden resumir en uno: un sistema de crawling debe procesar el mayor y mejor número de contenidos posibles. Los trabajos existentes hasta la fecha se han centrado en otros desafíos dejando de lado éste. El trabajo que se presenta, se centra en una arquitectura de crawling que pretende evitar el procesamiento de recursos irrelevantes e intentar refrescar contenidos en el momento más adecuado para minimizar los refrescos innecesarios a la vez que su nivel de obsolescencia. Para ello incorpora un módulo de detección de páginas "basura" (Spam y páginas Soft-404) y un módulo de detección de cambios basado en los accesos de los usuarios, utilizando para ello una arquitectura colaborativa. De esta forma, evitando procesar la "basura" web y evitando que el sistema regrese a páginas que no han cambiado, ayudará a reducir el número de recursos usados, lo que permitirá utilizar esos recursos en procesar un mayor número de páginas. La arquitectura propuesta se describe y analiza en el capítulo IV.

La sección II.3.1 se centra en el estudio de los diferentes tipos de Web Spam y las técnicas existentes para su detección. Sin embargo, las técnicas y sistemas existentes abordan un único tipo de Spam. Además, los resultados que muestran deben ser mejorados, y adaptados a las necesidades de los sistemas de crawling. Debido a estas deficiencias y limitaciones, se han definido un conjunto de novedosas heurísticas que permiten la detección de Web Spam, mejorando la eficacia y eficiencia de los métodos existentes. En el capítulo V se discuten las citadas heurísticas, y se comparan con los sistemas existentes.

Durante el estudio de las técnicas existentes para la detección de páginas Soft-404, se ha detectado que presentan diferentes limitaciones. Los sistemas existentes no detectan cierto tipo de páginas Soft-404, y existen múltiples escenarios que no han sido contemplados. Otro punto negativo de las técnicas existentes, es que son poco eficientes para su aplicación en sistemas de alto rendimiento, como son los sistemas de crawling. Por estas razones, esta tesis, presenta un conjunto de técnicas para la detección de este tipo de páginas, que mejoran tanto la eficacia como la eficiencia de los sistemas presentes en la literatura. Por un lado, mejoran los resultados y los tipos detectados, y por otro el rendimiento del sistema. En el capítulo VI, se presentan un conjunto de técnicas de detección de páginas Soft-404 y el correspondiente método para combinarlas, junto con una comparativa con los resultados obtenidos por el sistema propuesto y por los sistemas presentes en la literatura.

El último de los objetivos del análisis del estado del arte, hace referencia al estudio de las técnicas existentes para el proceso de actualización de los contenidos previamente procesados. En la sección II.4 se analizan, por un lado, los mecanismos existentes para ayudar a los sistemas de crawling en su proceso de actualización, y por otro, las técnicas que han sido presentadas en diferentes artículos de la literatura relacionada.

Los mecanismos actuales para tratar de ayudar a los crawlers a detectar cambios realizados en páginas web, tienen múltiples deficiencias. El uso de la cabecera Last-Modified obliga a depender de la implementación que cada servidor web haga del protocolo HTTP. Por otro lado, los sistemas RSS no los usan todos los sitios web y su uso requiere un gran trabajo por parte de sus creadores. Proporcionar información sobre la frecuencia de cambios de sus recursos mediante el uso del protocolo de exclusión de robots, tiene una desventaja doble. En primer lugar, el crawler debe tener acceso al directorio raíz para acceder a dichos ficheros, y en segundo lugar, es un mecanismo complejo y costoso de mantener, debido a que es necesario conocer previamente la frecuencia de cambio de cada recurso [SZG07]. Por último, los acuerdos entre sitios web y sistemas de crawling para notificar las modificaciones realizadas, se realizan principalmente entre sitios web de gran relevancia y con crawlers de los motores de búsqueda más importantes. Por lo que esta posibilidad no puede ser usada por la mayoría de los crawlers y sitios web.

En los trabajos existentes, se observa que presentan una característica común: todos tratan de mantener actualizados los repositorios estimando el tiempo que puede tardar una página o sitio web en cambiar. El problema de decidir cuándo volver a procesar una página web basándose en datos estadísticos, es que es probable que esta decisión esté equivocada. En ese caso el crawler visitaría una página que no ha cambiado todavía, con la consecuente pérdida de recursos, o en otro caso, si dicha página hubiera sido cambiada hace demasiado tiempo y su información ya no fuese tan relevante, provocaría una pérdida de calidad en los contenidos indexados.

Para evitar los inconvenientes presentes, tanto en los mecanismos como en los artículos existentes, proponemos un sistema que permite detectar los cambios realizados en las páginas web casi en tiempo real. El funcionamiento de este sistema y su funcionamiento dentro del sistema de crawling, se describen en el capítulo VII.

Durante el estudio del estado del arte, se han detectado un conjunto de deficiencias y limitaciones, que han dado lugar a la creación de este trabajo. La tesis que se presenta, realiza, en primer lugar, un conjunto de estudios sobre la Web para tratar de caracterizarla y, de esta forma aprender a procesarla de forma mas eficiente. Por otro lado, este trabajo, propone novedosas técnicas para la detección de Web Spam y páginas Soft-404. Por último, se incluye un sistema que permite detectar, casi en tiempo real, las modificaciones realizadas en las páginas y sitios web. Estos avances (detección de Web Spam y páginas Soft-404, y detección de cambios en páginas web), son los tres módulos que forman parte de la arquitectura de crawling eficiente que se propone en esta tesis.

III

Estudios de Caracterización de la Web

En este capítulo se presentan los estudios realizados sobre la Web con el fin de caracterizarla y definir los desafíos que presenta para los crawlers. Estos estudios han sido el punto de partida para el posterior diseño e implementación de una arquitectura de crawling eficiente y para la definición de un conjunto de políticas de actuación de sistemas de crawling y buscadores web.

El capítulo está dividido en tres secciones. En la sección III.1, se realiza un análisis del nivel de procesamiento de la Web Oculta del lado cliente por parte de los crawlers. En la sección III.2, se presenta un estudio comparativo sobre la Web Global y Española, durante 3 años. Se analizan aspectos tales como la similitud de las páginas, su edad, las tecnologías utilizadas, los formatos más usados, etc. Por último, en la sección III.3 se discuten las conclusiones obtenidas a partir de los estudios anteriores, y las políticas de actuación propuestas que serán consideradas como base para el diseño de la arquitectura de un crawler eficiente, que será presentada en los siguientes capítulos.

III.1 Análisis de la Web Oculta del Lado Cliente

Existen numerosos trabajos en la literatura que analizan las complejidades del tratamiento de la Web Oculta del lado servidor, sin embargo, son menos los trabajos existentes encargados de caracterizar la Web Oculta del lado cliente. Algunos de los trabajos centrados en la Web Oculta del lado cliente proponen técnicas para dotar a los sistemas de crawling de algoritmos que permitan su tratamiento. Sin embargo, no existen artículos que analicen qué parte de la Web Oculta del lado cliente está siendo tratada actualmente por los sistemas de crawling, y por tanto cuánta información queda fuera.

El nivel de utilización de lenguajes de script en el diseño de sitios web ha variado a lo largo del tiempo. Tras el gran auge de sus comienzos, fue perdiendo protagonismo, en gran parte debido a las dificultades de los crawlers globales para acceder a su información. Actualmente, debido a la aparición de tecnologías como AJAX y a un nuevo modelo de diseño de sitios web en los que el lado cliente gana mucha importancia, las tecnologías de scripting han resurgido con fuerza. Por ello, desde el punto de vista de los sistemas de crawling sería útil, no sólo conocer el porcentaje de fuentes en Internet que utilizan este tipo de tecnologías en la actualidad, sino también para qué las están utilizando.

El objetivo de esta sección es definir una serie de niveles de utilización de tecnologías de script, estimando para cada uno de ellos el coste computacional que supondría ser tratado por un sistema de crawling. La escala tiene en cuenta aspectos tales como: sitios que requieren login/password, enlaces estáticos, enlaces con script en href, enlaces con script en algún manejador de evento, generación de código HTML utilizando la función document.write, etc. Mediante la escala propuesta se obtendrá: a) clasificación de los sistemas de crawling en base a su efectividad accediendo a la Web Oculta del lado cliente, y b) indicadores sobre el porcentaje de la Web oculta del lado cliente que está quedando fuera del alcance de los sistemas de crawling.

Para la creación de esta escala se ha realizado un análisis de las tecnologías del lado cliente y de las características de los enlaces en las páginas web. Como resultado de este análisis se obtuvieron una serie de escenarios que agrupados en base a características comunes nos han permitido definir la escala. El estudio también incluye un conjunto de métodos que permiten evaluar los sistemas de crawling de acuerdo a los niveles de la escala. Para la obtención de los resultados de los crawlers se ha desarrollado un sitio web que contiene los diferentes tipos de enlaces de cada uno de los niveles de la escala.

Esta sección está estructurada de la siguiente forma. En la sección III.1.1 se introducen las tecnologías del lado cliente y su uso para la construcción de sitios web. En la sección III.1.2 se realiza un listado de las dificultades con las que se encuentran los sistemas de crawling para procesar la Web. Estas dificultades han sido clasificadas para poder crear una primera versión de la escala. La sección III.1.3 muestra la escala definida, teniendo en cuenta el uso de las diferentes tecnologías en la Web, la complejidad computacional de cada escenario y qué crawlers han conseguido procesar cada uno de ellos. En la siguiente sección se presentan un conjunto de métodos que permiten evaluar la eficacia de los crawler procesando la Web Oculta del lado cliente. En la sección III.1.5 se describe el sitio web creado para la realización de experimentos y se discuten los resultados obtenidos para los diferentes crawlers en el sitio web y los resultados obtenidos en la escala propuesta.

III.1.1 Tecnologías Web del Lado Cliente

El primer paso para la creación de la escala es entender el impacto de las diferentes tecnologías del lado cliente en la Web (Crawling Web Coverage). Dichas tecnologías son utilizadas, entre otras, para realizar las siguientes acciones:

- Creación de aplicaciones interactivas.
- Generación de contenido dinámicamente.
- Contenidos gráficos vectoriales, sonido, flujo de vídeo y audio, etc.
- Mejora de la velocidad, interactividad, y usabilidad de las páginas web.

La navegación dinámica del usuario, generada en base a determinadas acciones o datos del usuario, provoca que la complejidad de extracción de los enlaces sea mucho mayor que si estos fueran estáticos.

Para el estudio de estas tecnologías, se ha utilizado el dataset "The Stanford WebBase Project" [1], que forma parte del proyecto de la Universidad de Stanford "Stanford Digital Libraries Project" [2].

[1] http://dbpubs.stanford.edu:8091/~testbed/doc2/WebBase/
[2] http://diglib.stanford.edu:8091/

Este dataset está formado por más de 260 TB de datos, organizados en diferentes subconjuntos dependiendo de la temática que traten (temática generalista, desastres naturales, gubernamentales, etc.). Para no sesgar el contenido del dataset estudiado, se han utilizado páginas web incluidas dentro del conjunto de datos de temática generalista. Para el análisis se creó un subconjunto de 120 millones de páginas web del dataset del año 2011.

A continuación se explican en detalle cada una de las tecnologías estudiadas:

- JavaScript [Fla98], es un lenguaje orientado a objetos, imperativo, débilmente tipado y dinámico. JavaScript se diseñó con una sintaxis similar al C, aunque adopta nombres y convenciones del lenguaje de programación Java.

  Surge como una variación del estándar ECMAScript, especificación de lenguaje de programación publicada por ECMA International. El desarrollo empezó en 1996 y fue propuesto como estándar por Netscape Communications Corporation. Actualmente está aceptado como el estándar ISO 16262.

  Su uso por parte de los navegadores permite mejoras en la interfaz de usuario y la creación de páginas web dinámicas, en base a los eventos generados por el usuario, a través del acceso al árbol DOM [w3d11]. El árbol DOM ("Modelo de Objetos del Documento" o "Modelo en Objetos para la Representación de Documentos") es un conjunto estándar de objetos para representar documentos HTML y XML. Ofrece un modelo sobre cómo deben combinarse dichos objetos, y una interfaz que permite acceder a ellos y manipularlos. Actualmente, el responsable del DOM es el World Wide Web Consortium (W3C).

- VBScript [KHKHR07] (abreviatura de Visual Basic Script), lenguaje interpretado creado por Microsoft como variante del lenguaje Visual Basic. Su funcionalidad es similar a la que aporta JavaScript.

  Actualmente es usado principalmente por los administradores de Windows como herramienta de automatización, ya que permite más margen de actuación y flexibilidad que el lenguaje batch desarrollado a finales de los años 1970 para el MS-DOS. Además, es parte fundamental de la ejecución de aplicaciones de servidor programadas en ASP (Active Server Pages) [Mic11]

- Flash [BWN07], aplicación que permite la creación y manipulación de gráficos vectoriales, con posibilidades de gestión de código mediante un lenguaje de script denominado ActionScript Flash. Utiliza gráficos vectoriales y gráficos rasterizados, sonido, código de programa y flujo de vídeo y audio bidireccional.

  ActionScript es un lenguaje de programación de características parecidas a JavaScript y VbScript. Fue desarrollado con la idea de ofrecer a los desarrolladores una forma de programación más interactiva. La versión usada actualmente, ActionScript 3.0, se aproxima más fielmente al paradigma de programación orientada a objetos al ajustarse mejor al estándar ECMA-262. Además incluye nuevas características como el uso de expresiones regulares y nuevas formas de empaquetar las clases.

- Python [DDH00], es un lenguaje de programación interpretado y multiplataforma que usa tipado dinámico. Fue creado a finales de los 80 por Guido van Rossum en el Centro para las Matemáticas y la Informática (CWI, Centrum Wiskunde & Informática), en los Países Bajos, como un sucesor del lenguaje de programación ABC. Su filosofía hace hincapié en una sintaxis limpia, lo que favorecerá la creación de código legible.

  Python es un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Actualmente está administrado por la Python Software Foundation.

- Tcl [Ous94], del acrónimo inglés Tool Command Language, es un lenguaje de script creado por John Ousterhout. Se creó con el propósito de usar una sintaxis sencilla que facilitase su aprendizaje, sin detrimento de la funcionalidad y expresividad. Los scripts Tcl son más compactos y legibles que los programas funcionalmente equivalentes en otros lenguajes de programación.

  Es un lenguaje multiplataforma, cuyo código puede ser creado y modificado dinámicamente. Una de sus características más significativas es su extensibilidad. En el caso de que una aplicación Tcl requiera una funcionalidad no ofrecida por Tcl estándar, dichas funcionalidades pueden ser creadas usando C, C++ o Java.

- Applet [Wik11], componente Java de una aplicación que se ejecuta en el cliente web. Usualmente se utiliza para proporcionar información gráfica e interactuar con el usuario. Por cuestiones de seguridad, no mantiene estado en el servidor y el conjunto de operaciones que puede realizar es muy restringido.

  Un applet permite tener acceso casi completo a la máquina, con velocidades similares a la de lenguajes compilados, lo que permite crear soluciones más escalables al número de usuarios.

- AJAX [Hol08] acrónimo de Asynchronous JavaScript And XML, tecnología basada en JavaScript, que permite la creación de aplicaciones interactivas. Dichas aplicaciones se ejecutan en el lado cliente, y permiten una comunicación asíncrona con el servidor, lo cual permite realizar y obtener cambios en las páginas web sin necesidad de volver a cargarlas. Esto provocará una mejora de la experiencia del usuario, ya que logra mejorar la interactividad, velocidad y usabilidad en las aplicaciones.

  AJAX es multiplataforma y puede ser utilizado en infinidad de sistemas operativos y navegadores basados en estándares abiertos como JavaScript y Document Object Model (DOM). AJAX está basado en cuatro tecnologías:

  - XHTML o HTML y hojas de estilo en cascada (CSS).
  - DOM.
  - XMLHttpRequest, un objeto que permite el intercambio asíncrono de información con el servidor.

TECNOLOGÍA	MES							TOTAL
	Enero '11	Febrero '11	Marzo '11	Abril '11	Mayo '11	Junio '11	Julio '11	
Páginas	8519300	20057638	18700000	1230000	12677481	24505000	33453991	**119143410**
JavaScript	5791148	10436415	12672553	902780	6960075	16959179	18013475	**71735625**
	67.98%	**52.03%**	**67.77%**	**73.40%**	**54.90%**	**69.21%**	**53.85%**	**60.30%**
VBScript	7871	38243	19361	213	6253	20700	32529	**125170**
	0.09%	0.19%	0.10%	0.02%	0.05%	0.08%	0.10%	**0.11%**
Flash	257134	430772	584661	29376	273882	829380	669211	**3074416**
	3.02%	**2.15%**	**3.13%**	**2.39%**	**2.16%**	**3.38%**	**2.00%**	**2.58%**
Python	9	27	31	0	15	33	44	**159**
	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	**0.00%**
Tcl	228	42	228	3	26	1821	5083	**7219**
	0.00%	0.00%	0.00%	0.00%	0.00%	0.01%	0.02%	**0.01%**
Applets	139	2832	1049	5	2142	1176	2671	**10014**
	0.00%	0.01%	0.01%	0.00%	0.02%	0.00%	0.01%	**0.01%**

Tabla III.1: Análisis de las tecnologías Web del lado cliente

– XML, formato usado normalmente para el intercambio de datos entre cliente y servidor.

La detección de las diferentes tecnologías estudiadas se ha realizado mediante el uso de un parser XML que permitió detectar la presencia de las etiquetas "script" y "applet". Posteriormente se utilizaron dos aproximaciones, una de ellas analizaba los atributos "type" y "content" de dichas etiquetas para determinar el tipo de lenguaje utilizado o si era un applet o un contenido Flash. En caso de que esta aproximación fallara, debido a que en ciertas ocasiones el contenido de los atributos no era correcto, y fuera un script o applet remoto, se analizó la extensión del fichero correspondiente.

La Tabla III.1 muestra los resultados obtenidos para cada tecnología (fila) en cada mes (columna). De acuerdo con estos resultados, el 60.30% de las páginas web contienen código JavaScript, mientras que solamente el 2.58% de ellas contienen ActionScript (Flash). El grado de ocurrencia de VBScript, Python y TCL es meramente simbólico.

Los resultados obtenidos confirman la importante presencia de las tecnologías de lado cliente en la Web. Este dato junto con la patente complejidad de extracción de enlaces de este tipo de páginas web, implican que los crawlers deben mejorar el procesamiento de estas tecnologías. En cualquier otro caso, como indican los resultados mostrados, se estará perdiendo una gran parte de la Web.

III.1.2 Análisis de la utilización de las Tecnologías del Lado Cliente

Para definir los niveles básicos de la escala se han tenido que identificar previamente los mecanismos más usados para la generación de enlaces. Para ello, se realizó un análisis de las tecnologías del lado cliente más extendidas (sección III.1.1) y un análisis manual de un subconjunto de 1000 páginas aleatoriamente seleccionadas del dataset previamente descrito. También se han considerado aquellos casos de generación de enlaces contemplados en la literatura [Dan02] [KB03].

Tras estos estudios se decidió considerar la generación de enlaces basándose en 4 características: tecnología utilizada, método de construcción de la cadena de texto del enlace, localización, tipo de URL, y en sus diferentes combinaciones. Según la tecnología utilizada en el enlace, existen los siguientes tipo:

- Enlaces de texto, constituyen el nivel más bajo de la escala.

  ```
  <a href="a_110001001000000000_test...html">Luis
  Ramirez Lucena</a>
  ```

- Navegación sencilla generada con JavaScript, VBScript o ActionScript. Esto incluye enlaces generados por medio de la función "document.write()" o funciones similares que permiten al desarrollador añadir enlaces de forma dinámica.

  ```
  <a href="JavaScript:  ...">Paolo Boi</a>
  ```

- Navegaciones generadas por medio de un Applet.
- Navegaciones generadas por medio de Flash.
- En el caso de Applets y Flash, se han dividido en dos tipos de enlaces: aquellos que son pasados como argumentos a la aplicación, y aquellos que son creados como una cadena de texto dentro del código.
- Navegaciones generadas por medio de AJAX.
- Menús Pop-up, generados por medio de un script al cual se le asocia un evento.
- Enlaces definidos como cadenas de texto en ficheros .java, .class u otro tipo de fichero binario o de texto.
- Navegaciones generadas en funciones script. Pueden estar escritas en cualquier lenguaje de Script, y pueden estar alojadas dentro del propio HTML o en un fichero externo.
- Navegaciones generadas por medio de diversos tipos de redirecciones:

- Redirecciones especificadas en la etiqueta `<meta>` del documento HTML.
- Redirecciones generadas en el evento "onLoad" de la etiqueta HTML `<body>`.
- Redirecciones generadas por scripts cuando se dispara un evento de la página (por ejemplo el evento "onClick").
- Otras redirecciones en bloques de script que son ejecutadas en el momento en que se carga la página web.
- Redirecciones ejecutadas en un applet cuando la páginas web es cargada.
- Redirecciones Flash que se ejecutaran cuando la página y su correspondiente fichero Flash sean procesados.

El método de construcción de la cadena de texto es otra de las características que deben de considerarse a la hora de analizar un enlace. Entre los diferentes métodos existentes están los que se describen a continuación:

- Una cadena de texto estática dentro del Script.

```
menu_static_embedded_relative(){
    document.location="a_10010101100000000...html";
}
```

- Concatenación de una cadena de texto.

```
function menu_concatenated_embedded_relative(){
    var out="";
    out="a_10010010100000000000_test_menu" +
    "_concatenated_embedded_relative.html";
    document.location=out;
}
```

- Ejecución de una función que construye una URL en diversos pasos.

```
function menu_special_function_embedded_relative(){
    var a1="win",a2="dow",
    a3=".location.",="replace",a5;
    a5="('a_1001000110000000000_test_menu_special_function";
    var a6="_embedded_relative.html')"; var i,url="";
    for(i=1;i<=6;i++){
       url+=eval("a"+i);
    }
    eval(url);
}
```

Por último, además de la tecnología utilizada y el método de generación de la cadena de texto, se han contemplado dos características adicionales. El tipo de URL, dependiendo de si la ruta del enlace es relativa o absoluta, y la localización del código script utilizada para la generación del enlace, para la cual existen dos opciones, embebido en el propio HTML o en un fichero externo.

Es importante indicar que los métodos descritos pueden ser combinados entre ellos (ciertos sitios web generan menús Pop-up dinámicamente por medio de llamadas a la función "document.write()"). Sin embargo, su combinación no proporcionaría mayor información sobre las tecnologías del lado cliente en la Web o sobre como los crawlers las procesan. Por ejemplo, no es necesario tener en cuenta la combinación de menús JavaScript y la utilización de la función "document.write()" para generarlos, puesto que se puede saber si un crawler es capaz de procesar ese tipo de menús analizando los dos casos base por separado.

Teniendo en cuenta todas las variantes de generación de enlaces descritas, se han identificado 70 escenarios que deben ser considerados en la definición de la escala. Los diferentes escenarios y su explicación se muestran en la Tabla III.2. Las primeras 4 columnas muestran las 4 características consideradas a la hora de generar un enlace: tipo de URL, tecnología utilizada, localización del enlace y método utilizado para la creación de la cadena de texto del enlace. La última columna indica el número de test asignado a cada escenario: combinación de tipo de URL, tecnología, localización y tipo de cadena utilizados.

III.1.3 Escala para Crawlers Web

Para crear la escala a partir de los 70 escenarios propuestos, se ha implementado un proceso de agrupamiento en base a su dificultad:

- Dificultad teórica del procesamiento de cada escenario.
- Dificultad práctica, para los sistemas de crawling existentes para tratar cada escenario.

Tipo de URL	Tecnología	Localización	Tipo de cadena		Test
Relative / Absolute	Text	Embedded	Static		1 - 2
	JavaScript	Embedded / External	Static / Concatenated / Special Function		3 - 38
	Document.write()	Embedded / External	Static / Concatenated / Special Function		3 - 38
	Menu JavaScript	Embedded / External	Static / Concatenated / Special Function		3 - 38
	Link in .java	External	Static		39 - 40
	Link in .class	External	Static		41 - 42
	Applet-Link HTML	Embedded	Static		43 - 44
	Applet-Link Class	External	Static		45 - 46
	Flash-Link HTML	Embedded	Static		47 - 48
	Flash-Link SWF	External	Static		49 - 50
	AJAX	Embedded	Static		61 - 62
	JavaScript with #	Embedded	Static / Special Function		63 - 66
	VBScript	Embedded	Static / Special Function		67 - 70
Relative / Absolute	Redirect	External	Static	Tag Meta	51 - 52
				Tag body	53 - 54
				JavaScript	55 - 56
				Applet	57 - 58
				Flash	59 - 60

Tabla III.2: Combinación de los tipos de enlaces

Con respecto a la primera cuestión, se ha realizado un análisis teórico y empírico de los escenarios. El análisis teórico consiste en definir qué módulos necesitaría un crawler para procesar cada escenario (columna "Proceso Extracción" de la Figura III.1). Por ejemplo, para el procesamiento de enlaces de texto (nivel 1), sólo se necesitaría un crawler y un extractor de URLs básico. Sin embargo, para extraer URLs embebidas en código JavaScript (nivel 2), el sistema necesitaría un intérprete de JavaScript, lo cual incrementa notablemente su complejidad. Tras esto, para comprobar que el análisis teórico realizado era correcto, se ha realizado un análisis empírico de los escenarios utilizando herramientas como intérpretes de JavaScript, decompiladores Flash, etc. Dependiendo del número y complejidad de módulos necesarios para la extracción, se ha asignado un valor, que se muestra en la columna "Complejidad" de la Figura III.1.

Los valores obtenidos dependen de la dificultad de desarrollo de los módulos y del coste computational de su ejecución. El agrupamiento realizado se basa en la asunción de que los escenarios que requieren un mayor número de módulos tienen mayor dificultad para ser procesados. Por ejemplo, el coste de desarrollo de una aplicación Flash que muestra los enlaces en un menu desplegable (escenarios 47, 48, 49 y 50) es mucho mayor que un enlace generado por una llamada a un código JavaScript embebido en el HTML (escenarios 3 y 4). De igual modo, el coste de extracción de este tipo de enlaces es mucho mayor que un enlace de texto embebido en el propio código HTML.

Para el análisis de la dificultad práctica de cada escenario, es necesario conocer cómo un sistema de crawling procesa los diferentes escenarios propuestos. Para ello, se ha implementado un sitio web (sección III.1.5) que contiene todos los escenarios de la Tabla III.2. La Figura III.4 y la Tabla III.5 (sección III.1.5) muestran los resultados obtenidos por los diferentes crawlers procesando el sitio web de ejemplo. Estos resultados han permitido generar un nuevo agrupamiento desde un punto de vista práctico. Este agrupamiento difiere ligeramente del construido en base a la dificultad teórica de cada escenario.

Para la construcción de la escala se han utilizado como base los agrupamientos de los diferentes escenarios en base a su complejidad teórica y práctica, obteniéndose la clasificación que se muestra en la Figura III.1. Como se puede ver en en la Figura III.4 y en la Tabla III.5, en ciertas ocasiones la dificultad teórica no es acorde con la dificultad práctica. Ciertos escenarios sencillos desde el punto de vista teórico no han sido procesados por diversos crawlers, y en otros casos escenarios complejos desde el punto de vista teórico sí han sido procesados. En dichas situaciones se optó por dar mayor importancia al análisis teórico que al práctico. Finalmente, tras este proceso de refinamiento, se han unido los escenarios descritos creando una escala con 8 niveles. La Figura III.1 muestra la escala propuesta, junto con una breve descripción, los escenarios que forman ese nivel y la complejidad del mismo.

Como se explicó, este estudio tiene dos objetivos, por un lado clasificar los sistemas de crawling en base a su capacidad de procesamiento de la Web Oculta del lado cliente, y por otro, determinar la parte de la Web que no está siendo tratada. La escala propuesta, hasta el momento, es válida para el primero de los objetivos, ya que permite evaluar la capacidad de un crawler. Sin embargo, aún conociendo qué tipos de enlaces no tratan los crawlers, no se puede determinar qué parte de la Web se está perdiendo. Para ello, es necesario conocer la frecuencia de aparición en la Web de cada una de las tecnologías utilizadas en cada escenario.

Esta frecuencia se ha obtenido a partir de los resultados mostrados en la sección III.1.1. Este dato se muestra en la columna “F” de la Figura III.1. Por ejemplo, el porcentaje de uso de JavaScript es del 60.30%, por lo que, normalizando a 1, los escenarios basados en JavaScript tendrán una frecuencia de 0.6. En el caso de los enlaces de texto se ha asumido que todas las páginas web tienen al menos un enlace de texto, por lo que su frecuencia de aparición es 1. Esta información complementa la escala presentada y permite determinar con exactitud qué proporción de la Web no está siendo tratada por los sistemas de crawling, así como el nivel máximo de complejidad que son capaces de procesar.

En la siguiente sección se proponen diferentes métodos de evaluación para medir la eficacia de los sistemas de crawling de acuerdo a la escala presentada.

III.1.4 Métodos de Evaluación sobre la Escala

Además de la escala ya explicada, se han propuesto un conjunto de métodos que permiten clasificar los crawlers de acuerdo al nivel de complejidad de los enlaces que son capaces de procesar.

Nivel	Descripción	Escenarios	F	Complejidad	Proceso Extracción
1	Text link	1,2	1	Muy Bajo	Modulo of Crawling - Extractor of URLs
2	JavaScript/Document.Write/Menu - Static String - Embedded	3,4,15,16,27,28	0.6	Bajo	Modulo de Crawling - Interprete JS
	JavaScript - Concatenated String - Embedded	6	0.6		Analizador de redirecciones - Extractor de URLs
3	HTML/onBody/JavaScript Redirect	51,52,53,54,55,56	1	Bajo	Modulo de Crawling - Interprete JS
	JavaScript # - Static String - Embedded	63,64	0.6		Extractor de URLs
4	VBScript - Static String - Embedded	67,68	0.01	Medio	Modulo de Crawling - Interprete VBS
	VBScript - Special Function - Embedded	70	0.01		Extractor de URLs
5	JavaScript/Document.Write - Static String - External/Embedded	9,10,18	0.6	Medio /	Modulo de Crawling - Interprete VBS
	Document.Write/Menu - Static String - External	21,22,33,34	0.6	Alto	Extractor de URLs
	Menu - Concatenated String - Embedded	30	0.6		
6	JavaScript/Document.Write/Menu-Concatenated String-External	12,24,36	0.6	Medio /	Modulo de Crawling - Interprete JS
	Applet - Static String in HTML	43,44	0.03	Alto	Extractor Avanzado de URLs
7	JavaScript - Concatenated String - External/Embedded - Relative	5,11	0.6	Alto	Modulo de Crawling
	JavaScript - Special Function - External/Embedded - Relative	7,8,13,14	0.6		Interprete Avanzado JS
	Document.Write - Concatenated String - External/Embedded	17,23	0.6		Decompilador Java
	Document.Write - Special Function - External/Embedded	19,20,25,26	0.6		Analizador ficheros externos
	Menu - Concatenated String - External/Embedded - Relative	29,35	0.6		Extractor Avanzado de URLs
	Menu - Special Function - External/Embedded	31,32,37,38	0.6		
	Link in .java	39,40	0.03		
	AJAX Link - Absolute	62	0.6		
8	Link in .class	41,42	0.03	Muy Alto	Modulo de Crawling
	Applet - Static String in .class	45,46	0.03		Interprete Avanzado JS
	Flash - Static String in HTML/SWF	47,48,49,50	0.3		Decompilador JS - Decompilador Flash
	Applet/Flash Redirect	57,58,59,60	1		Interprete Avanzado VBS
	AJAX Link - Relative	61	0.6		Analizador ficheros externos
	JavaScript with # - Special Function - Embedded	65,66	0.6		Analizador de redirecciones
	VBScript - Special Function - Embedded	69	0.01		Extractor Avanzado de URLs

Figura III.1: Escala y clasificación de los enlaces en base a la dificultad

Como paso previo para describir formalmente los métodos, es necesario definir algunos conceptos:

- Sea $S = \{Full\ set\ of\ scenarios\}$, siendo $|S|$ el número total de escenarios y S_i un valor binario que indica si el escenario $i-th$ ha sido procesado, "1", o "0" en otro caso.
- Sea $N = \{Full\ set\ of\ levels\ that\ the\ crawler\ processes\ successfully\}$ siendo $|N|$ el número total de niveles y $|N_i|$ cada uno de los elementos de N. Por ejemplo, el crawler Web Copier Pro [web11b] (ver sección III.1.5) ha sido capaz de procesar los niveles 1, 2, 5 y 7, por lo que $N = \{1, 2, 5, 7\}$, y el número total de niveles procesados, $|N|$, es de 4. Con estos resultados el $|N_0|$ sería el nivel 1, el $|N_1|$ el 2, y así sucesivamente.
- Sea $C = \{Full\ set\ of\ crawlers\}$, siendo $|C|$ el número total de crawlers, $C_i = \{Set\ of\ crawlers\ that\ achieved\ the\ scenario\ i\}$.

  Los valores de C y C_i son constantes para la escala propuesta, y han sido calculados evaluando los crawlers contra el sitio web que contiene los escenarios y niveles de la escala definida. De esta forma, cualquier otro crawler que desee ser clasificado en base a la escala y métodos propuestos, deberá usar los valores de C_i mostrados en la Tabla III.3. En ella, el conjunto de los 70 escenarios se muestran en forma de matriz, donde el identificador de cada escenario se construye tomando el valor de la fila superior (unidades), con el valor correspondiente de la columna de la izquierda (decenas). Por ejemplo, el escenario 35 formado con el número 3 en las decenas y el 5 de las unidades (resaltados en negrita en la Tabla III.3), ha sido procesado por 2 de los crawlers estudiados.

Escenarios	Unidades									
		0	1	2	3	4	**5**	6	7	8	9
D	0	7	7	6	6	1	6	1	1	3	3
e	1	1	2	1	1	6	6	1	3	1	1
c	2	3	3	1	2	1	1	6	6	1	3
e	**3**	1	1	3	3	1	**2**	1	1	0	0
n	4	1	1	2	2	0	0	0	0	0	0
a	5	5	5	5	5	5	5	0	0	0	0
s	6	0	1	5	5	0	0	5	5	0	4

Tabla III.3: Valores de C_i

- Sea f_i la frecuencia (Figura III.1) de la ocurrencia de i.

Como se observa en la definición de los conceptos previos, para aplicar los métodos de evaluación propuestos sobre un crawler nuevo, es necesaria la obtención de S y N. Para ello, dicho crawler debe ser evaluado previamente contra un sitio web que implemente la escala definida. Un ejemplo de sitio web fue él creado para este estudio (ver sección III.1.5).

A continuación se describen cada uno de los métodos que se proponen:

- Simple Average: trata a cada uno de los escenarios definidos de la misma forma, sin tener en cuenta su dificultad. Este método valorará mejor a aquellos crawlers que procesen un mayor número de escenarios, y consecuentemente a aquellos que presten más atención a la Web Oculta en general.

$$SA = \left(\frac{1}{|S|}\right) \sum_{i=1}^{|S|} S_i$$

- Maximum Level: ordena los crawlers de acuerdo al nivel más alto de dificultad que es capaz de procesar. Un crawler que obtenga una puntuación i, quiere decir que dicho crawler tiene capacidad para procesar cualquier escenario de dicho nivel o de un nivel inferior. Ciertos crawlers han procesado algunos niveles y no los niveles inferiores. Esto podría ser debido a problemas con el sitio web y no a que dicho crawler no es capaz de procesar los enlaces de dichos niveles inferiores. Este método de evaluación asume que si un crawler tiene capacidad para procesar el nivel i, también la tendrá para procesar los niveles inferiores.

$$ML = i \mid \forall\, i, j \in 1 \ldots |N|\, N_i, N_j = 1 \Lambda\, i \geq j$$

- Weighted Average: este método depende del número de crawlers que previamente hayan sido capaces de procesar cada escenario (C_i), y si el nuevo crawler a evaluar ha sido capaz o no. Asumiendo que todos los crawlers pueden procesar los escenarios más sencillos, este método considera mejor a aquellos crawlers que procesen un alto número de enlaces de alta dificultad, ya que ellos serán los únicos que puntuarán en los escenarios difíciles.

$$WA = \left(\frac{1}{|C|+1}\right) * \sum_{i=1}^{|S|} ((C_i + S_i) * S_i)$$

- Eight Levels: en este método a cada nivel de la escala se le asigna el valor 1. Si un crawler procesa todos los escenarios de un nivel dicho crawler obtiene 1 punto. Para cada escenario que el crawler procese adecuadamente, obtendrá $1/n$ puntos, donde n es el total de escenarios que contiene cada nivel.

  Sea $L = \{L_1 \dots L_8\}$ el conjunto de los escenarios que representan los 8 niveles previamente definidos. Entonces L_{ij} es $j-th$ escenario del $i-th$ nivel. El número de elementos de cada conjunto puede ser diferente de los otros.

$$\begin{array}{ccccc} L_{11} & \dots & L_{1p} & \rightarrow & L_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ L_{81} & \dots & L_{8w} & \rightarrow & L_8 \end{array}$$

$$EL = \sum_{i=1}^{8} \sum_{j=1}^{|L_i|} \frac{L_{ij}}{|L_i|}$$

Los métodos presentados permiten conocer la dificultad de los enlaces que un crawler es capaz de tratar. Sin embargo, para poder obtener un dato más concluyente sobre cómo procesan los crawler la Web Oculta del lado cliente, es importante contextualizar las tecnologías según su grado de uso en la Web. Un crawler no será "mejor" que otro por el simple hecho de procesar niveles superiores de la escala, cuando los tipos de enlaces de dicho nivel tienen poca presencia en la Web.

Para solucionar este problema, se ha creado la métrica denominada Crawling Web Coverage, que tiene en cuenta el número de escenarios que un crawler procesa, pero ponderando cada tecnología por su frecuencia de aparición en la Web. Tras esto se han extendido los métodos Weighted Average y Eight Levels, para crear dos nuevos métodos llamados Ranked Weighted Average y Ranked Eight Levels, que sí consideran el grado de ocurrencia de cada uno de los escenarios en la Web.

Para ello, se han comparado los resultados obtenidos en nuestro estudio sobre las tecnologías web (sección III.1.1) con los presentados en W3techs [w3W11] y builtwith [web11a]. Dichos estudios presentan resultados

similares a los nuestros, a excepción de la presencia de JavaScript en la Web. De acuerdo con dichos estudios, el 90% de las páginas web usan JavaScript, cifra que baja hasta el 60% en nuestro estudio. Finalmente se optó por usar el resultado obtenido en el estudio realizado, ya que los datos presentados por W3techs [w3W11] y builtwith [web11a], se basaban en el análisis de páginas web incluidas en el top 1 millón (en base al ranking del buscador Alexa [3]) o que tenían más de 10000 visitas. El hecho de considerar solo ese tipo de páginas para sus estudios sesga los resultados, ya que páginas web con algo de tráfico y uso suelen contener tecnologías innovadoras que usualmente se basan en AJAX.

La columna "F" (Frecuencia) de la Figura III.1 muestra la frecuencia de ocurrencia en la Web de la tecnología utilizada en cada uno de los niveles. De esta forma, un crawler que procese un nivel con frecuencia 1 será más valorado que un crawler que procesa un nivel de baja frecuencia, ya que la probabilidad de que aparezcan en la Web el tipo de enlaces del primero es mucho mayor al segundo. Por otro lado, un crawler que procese enlaces con una frecuencia de 0.01, obtendrá un bajo resultado ya que dicho tipo de enlaces son muy poco utilizados en la Web.

- Ranked Weighted Average

$$RWA = \left(\frac{1}{|C|+1}\right) * \sum_{i=1}^{|S|} ((C_i + S_i) * S_i * f_i)$$

- Ranked Eight Levels

$$REL = \sum_{i=1}^{8} \sum_{j=1}^{|L_i|} f_{i,j} * \frac{L_{i,j}}{|L_i|}$$

Los métodos Simple Average y Weighted Average para poder ser comparados con mayor facilidad, deben de ser normalizados para obtener valores entre 0 y 8.

Cada uno de los métodos presentados proporciona un punto de vista diferente sobre las capacidades de un crawler. Por lo tanto, el conjunto de los métodos es lo que proporciona una visión Global de cada crawler, de sus capacidades de procesamiento de las tecnologías del lado cliente y del porcentaje de Web que no está pudiendo tratar.

III.1.5 Resultados Experimentales

En esta sección se describen los resultados de los experimentos realizados para la creación de la escala y su aplicación en la clasificación de crawlers Open-Source y comerciales. Como paso preliminar se describirá el sitio web creado para la realización de los tests.

[3] http://www.alexa.com/

Sitio Web de prueba

Para poder evaluar como los diferentes sistemas de crawling procesan los diferentes escenarios, se ha creado un sitio web[4], sobre el que realizar los experimentos. El sitio web contiene los 70 niveles definidos por la escala (Tabla III.2), así como los enlaces usados y la combinación de las diferentes tecnologías previamente explicadas. Con la finalidad de incentivar la indexación del sitio Web se han realizado las siguientes acciones: poner un enlace desde la página principal del departamento, elaborar el contenido en inglés y añadir a cada nivel la biografía de un ajedrecista. Esto último se ha hecho para evitar que, debido a que carecían de contenido y el sitio web contenía multitud de enlaces de difícil procesamiento, los motores de búsqueda lo filtraran al considerarlo Web Spam.

En la Figura III.2 se muestra la página principal del prototipo de sitio web. Se identifican las siguientes partes:

- En la parte superior, de izquierda a derecha, aparecen los enlaces en Menús JavaScript, los enlaces generados en el Applet y finalmente los enlaces en Flash.
- En el centro de la página aparece una tabla dividida en 4 columnas, mostrando primero el número que identifica a cada test, a continuación una pequeña descripción del test y finalmente el enlace relativo y absoluto.
- En la parte inferior, tras la tabla, aparece el contenido.

Por otro lado se ha creado una página de resultado para cada una de las pruebas, de tal modo que si el crawler es capaz de acceder a dicho contenido significa que ha sido capaz de procesar el enlace. En la Figura III.3 se muestra una página de resultado, formada por los siguientes elementos:

- En la parte superior, en el centro, se muestra el número y nombre del test.
- En la parte superior izquierda aparece el código del test, representado mediante una máscara binaria que representa numéricamente las características que tiene o no tiene la prueba. Por ejemplo, la mascara binaria de la página web resultado mostrada en la Figura III.3 es: 10100010100000000000000001. Cada uno de los dígitos binarios indica la presencia, 1, o ausencia, 0, de las diferentes niveles teóricos mostrados en la Tabla III.2. En este caso, por ejemplo, podemos ver que el primer 1 indica que es de nivel general, el siguiente 0 que no se trata de un enlace de texto y el siguiente 1 que se trata de un enlace del tipo "href=javascript:" y así sucesivamente.
- En el lateral izquierdo se muestra una tabla que enumera las características de la prueba.

[4] http://www.tic.udc.es/~mad/resources/projects/jstestingsite/

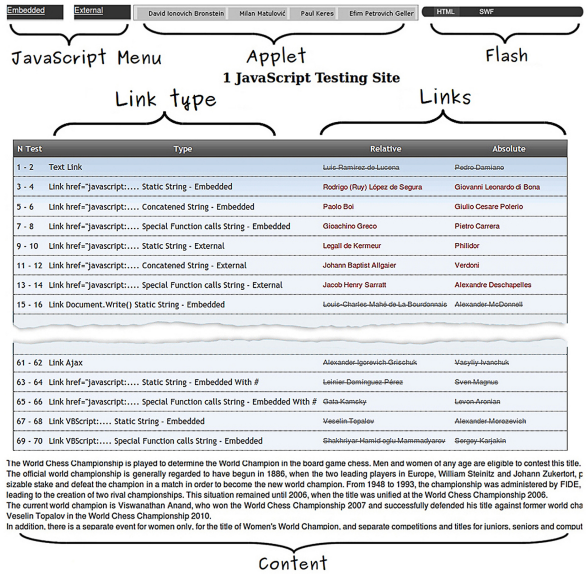


Figura III.2: Página principal del sitio web

- En la parte central se incluye la biografía de un maestro ajedrecista. Las diferentes biografías utilizadas se han descargado de la Wikipedia [5], cuyos contenidos están bajo licencia Creative Commons 3.0, con atribución para compartir y derivar nuevas obras a partir de la original.

Configuración de Experimentos

Para el desarrollo de la escala y para la clasificación de los sistemas de crawling de acuerdo con los métodos de evaluación propuestos, se han realizado dos tipos de pruebas. Estas pruebas permitirán comparar los resultados obtenidos por cada crawler en cada una de ellas.

Por un lado, se ha analizado el contenido indexado y almacenado por los crawlers en su repositorio. Esta prueba aportará una visión estricta de los escenarios que cada crawler ha procesado. Por otro lado, se han estudiado los logs de acceso del servidor web donde está alojado el sitio web. Con este estudio lo que se pretende es conocer en más detalle cómo los crawlers tratan de procesar cada nivel, y si han logrado extraer ciertos escenarios que por otros motivos no han sido indexados o almacenados en sus repositorios.

[5] http://es.wikipedia.org

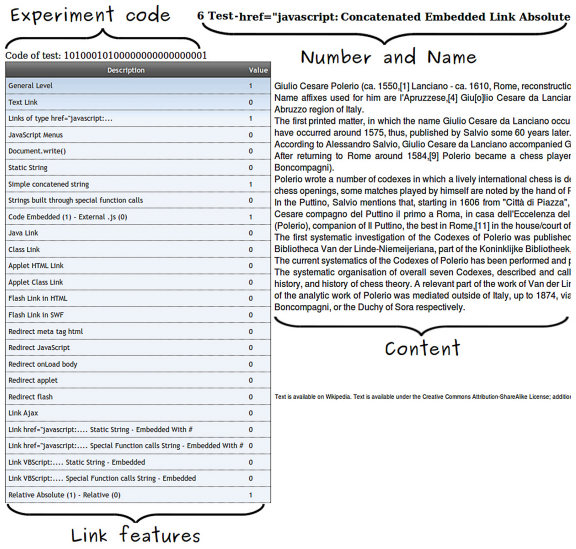


Figura III.3: Página web de resultado, asociada a un enlace de un escenario

Para identificar correctamente cada uno de los sistemas de crawling estudiados, se ha utilizado el User-Agent junto con su correspondiente IP o conjunto de IP's (obtenidos de la página web de robots [6]). Este par de datos (User-Agent y IP) evita problemas a la hora de identificar que ha procesado cada crawler, ya que es habitual encontrarse crawlers que usan el User-Agent de GoogleBot (u otros motores de busqueda conocidos) para evitar ser filtrados por el servidor web. Además, con la idea de acelerar la indexación del sitio web por parte de los buscadores, se ha registrado el sitio web en los principales motores de búsqueda: Google[7], Bing[8], Yahoo![9], PicSearch[10] y Gigablast[11].

Tanto para los datos de los logs como para los de los repositorios, se han analizado los resultados para los principales motores de búsqueda y para un conjunto de crawlers Open-Source y comerciales. Primero se ha estudiado cada tipo de crawler de forma independiente y posteriormente se han comparado los resultados en base al tipo de enlace y al método de construcción de la cadena del enlace.

Los resultados obtenidos en el sitio web han sido utilizados para evaluar cada uno de los sistemas de crawling

[6] http://www.robotstxt.org/
[7] http://www.google.es/addurl/
[8] http://www.bing.com/webmaster/SubmitSitePage.aspx
[9] http://siteexplorer.search.yahoo.com/submit
[10] http://www.picsearch.com/menu.cgi?item=FAQ
[11] http://www.gigablast.com/addurl

con los métodos propuestos y así poder obtener una clasificación de como los crawlers procesan la Web Oculta.

Crawlers Open-Source y Comerciales

En el estudio se han analizado 23 crawlers Open-Source y comerciales. La Tabla III.4 muestra sus características más relevantes desde el punto de vista de las tecnologías del lado cliente.

Entre las características que comparten, pero que no se muestran en el estudio, se pueden destacar: opciones en el uso de proxy, limitación del número de documentos descargados, uso de diferentes protocolos, inclusión/exclusión de ficheros, cookies, user-agent, opciones sobre dominios/directorios, logging, etc.

Crawler	Licencia	JavaScript	Flash	Formularios	Autenticación	Thread
Advanced Site Crawler	Free	Si	No	No	Si	Si
Essential Scanner	Free	No	No	No	Si	No
Gsite Crawler	Free	No	No	No	No	Si
Heritrix	Free	Si	No	No	Si	Si
Htdig	Free	No	No	No	Si	No
ItSucks	Free	No	No	No	Si	Si
Jcrawler	Free	No	No	No	No	No
Jspider	Free	No	No	No	Si	Si
Larbin	Free	No	No	No	Si	Si
MnogoSearch	Free	No	No	No	Si	No
Nutch	Free	No	Si	No	No	Si
Open Web Spider	Free	No	No	No	No	Si
Oss	Free	No	No	No	No	Si
Pavuk	Free	Si	No	Si	Si	Si
Php Crawler	Free	No	No	No	No	No
WebHTTrack	Free	Si	Si	No	Si	Si
JOC Web Spider	Shareware	No	No	No	Si	Si
MnogoSearch	Shareware	No	No	No	Si	Si
Teleport Pro	Shareware	Si	No	Si	Si	Si
Visual Web Spider	Shareware	No	No	No	Si	Si
Web Data Extractor	Shareware	No	No	No	Si	Si
Web2Disk	Shareware	Si	No	Si	Si	Si
Web Copier Pro	Shareware	Si	Si	Si	Si	Si

Tabla III.4: Crawlers Open-Source y comerciales

Tras examinar éstas y otras características, y teniendo en cuenta que se evaluará el tratamiento de las tecnologías del lado cliente, se han seleccionado 7 crawlers de entre los mostrados en la Tabla III.4.

Entre los crawlers Open-Source, se han seleccionado los siguientes: Nutch[12] [CC04], Heritrix[13] [MKSR04],

[12] http://nutch.apache.org/
[13] http://www.archive.org

Pavuk[14] y WebHTTrack[15]; y entre los crawlers comerciales: Teleport[16], Web2disk[17] y WebCopierPro[18].

III.1.6 Resultados de los Crawlers Open-Source y Comerciales

Los resultados han sido obtenidos analizando las páginas web indexadas en los repositorios de cada uno de los sistemas de crawling estudiados. La Figura III.4 resume dichos resultados, mostrando en el lado izquierdo se muestran los resultados obtenidos analizando los repositorios, y en el derecho los obtenidos analizando los logs de acceso.

Los mejores resultados los obtiene el crawler WebCopierPro (lado izquierdo de la Figura III.4), que procesa el 57.14% de los niveles, seguido por Heritrix con un 47.14% y por Web2Disk con 34.29% de los niveles. Únicamente un pequeño conjunto de los crawlers estudiados han obtenido más del 25% de los niveles procesados para ciertos tipos de enlaces.

Es importante observar que los peores resultados han sido obtenidos durante la ejecución de las redirecciones. En concreto WebCopierPro ha obtenido malos resultados en los enlaces con redirecciones, y sin embargo ha conseguido procesar el 100% de otros niveles con una dificultad superior. Ninguno de los crawlers ha obtenido un 100% en los niveles asociados a las redirecciones, ya que estos no están preparados para procesar redirecciones embebidas en Applets o Flash. Se ha observado que han descargado la página que contenía el Applet o Flash, pero no han ejecutado la redirección correspondiente.

Analizando los resultados obtenidos en base al tipo de enlace, se puede ver a que tipo de enlaces orientan su procesamiento la mayor parte de los crawlers. Dejando a un lado el 100% obtenido en la extracción de los enlaces de texto, se puede ver que los crawlers logran completar entre el 25% y el 40% de los escenarios de href="javascript..."; "document.write()"; menu links; links with "#" y VBScript. Sin embargo, únicamente han logrado procesar el 7% de los enlaces presentes en ficheros .class, .java y aquellos generados mediante AJAX, y ninguno de ellos ha logrado procesar los enlaces de Flash. Esto último puede ser debido a la escasa atención que los sistemas de crawling ponen en este tipo de enlaces.

En las últimas filas de la Figura III.4 se muestran los resultados obtenidos según el método de construcción del enlace. Realizando la media de los porcentajes mostrados se obtiene que los crawlers han sido capaces de tratar el 42.52% de los enlaces estáticos, el 27.38% de los generados mediante la concatenación de dos cadenas y el 15.18% de los enlaces generados por medio de funciones especiales. Estos resultados son indicativos de que la mayor parte de los crawlers utilizan expresiones regulares para la extracción de enlaces.

[14] http://www.pavuk.org
[15] http://www.httrack.com
[16] http://www.tenmax.com/teleport/pro/home.htm
[17] http://www.inspyder.com/ products/Web2Disk/Default.aspx
[18] http://www.maximumsoft.com/products/wc_pro/overview.html

Repositorios (Extraídos — %)															Logs (Extraídos — %)			
	Heritrix		Nutch		Pavuk		Teleport		Web2Disk		WebCopierPro		WebHTTrack		Teleport		WebHTTrack	
Text link	2	100.00%	2	100.00%	2	100.00%	2	100.00%	2	100.00%	2	100.00%	2	100.00%	2	100.00%	2	100.00%
Href=”JavaScript link	6	50.00%	3	25.00%	0	0.00%	3	25.00%	5	41.67%	12	100.00%	3	25.00%	4	33.33%	3	25.00%
Document.write link	6	50.00%	3	25.00%	0	0.00%	2	16.67%	4	33.33%	12	100.00%	2	16.67%	3	25.00%	2	16.67%
Menu link	6	50.00%	3	25.00%	0	0.00%	2	16.67%	4	33.33%	12	100.00%	2	16.67%	3	25.00%	2	16.67%
Flash link	0	0.00%	0	0.00%	0	0.00%	0	0.00%	0	0.00%	0	0.00%	0	0.00%	0	0.00%	2	50.00%
Applet link	2	50.00%	0	0.00%	0	0.00%	0	0.00%	0	0.00%	0	0.00%	2	50.00%	0	0.00%	2	50.00%
Redirects	6	60.00%	6	60.00%	2	20.00%	6	60.00%	4	33.33%	0	0.00%	6	60.00%	6	60.00%	6	60.00%
Class or Java link	0	0.00%	0	0.00%	0	0.00%	0	0.00%	0	0.00%	2	50.00%	0	0.00%	0	0.00%	0	0.00%
AJAX link	0	0.00%	1	50.00%	0	0.00%	0	0.00%	0	0.00%	0	0.00%	0	0.00%	0	0.00%	0	0.00%
Links with #	2	50.00%	2	50.00%	0	0.00%	2	50.00%	2	50.00%	0	0.00%	2	50.00%	3	75.00%	2	50.00%
VBScript link	3	75.00%	3	75.00%	0	0.00%	3	75.00%	3	75.00%	0	0.00%	2	50.00%	3	75.00%	2	50.00%
Enlace - Estático	26	61.90%	19	45.24%	4	9.52%	18	42.86%	22	52.38%	16	38.10%	20	47.62%	18	42.86%	22	52.38%
Enlace - Concatenación	6	50.00%	2	16.67%	0	0.00%	1	8.33%	1	8.33%	12	100.00%	1	8.33%	1	8.33%	1	8.33%
Enlace - Función	1	6.25%	2	12.50%	0	0.00%	1	6.25%	1	6.25%	12	75.00%	0	0.00%	5	31.24%	0	0.00%
Tests procesados	33	47.14%	23	32.86%	4	5.71%	20	28.57%	40	57.14%	40	57.14%	21	30.00%	24	34.29%	23	32.86%

Figura III.4: Resumen de los resultados obtenidos por los crawlers Open-Source y Comerciales. En el lado izquierdo se muestran los resultados obtenidos analizando los repositorios, y en el derecho los obtenidos analizando los logs de acceso

Con los resultados mostrados se puede concluir que la probabilidad de extraer un enlace con éxito utilizando expresiones regulares es inversamente proporcional a la dificultad de generación del enlace. Los datos mostrados en la Figura III.4, muestran que solamente una tercera parte de los enlaces que son generados con las tecnologías del lado cliente pueden ser procesados con crawlers Open-Source o comerciales.

Como complemento a los resultados mostrados, se han analizado los logs de accesos de los crawlers. Los resultados obtenidos son idénticos a los obtenidos analizando los repositorios excepto para dos crawlers, que se muestran en el lado derecho de la Figura III.4.

Estas diferencias son debidas a:

- WebHTTrack ha tratado de acceder a los niveles de Flash en que las direcciones están en el código HTML para ser pasadas a Flash mediante parámetros. No tuvo en cuenta que la estructura para pasar parámetros a Flash es ”parametro1=datos1&parametro2=datos2&....” por lo que considera toda la cadena como un único URL cuando no lo es.

  ```
  "GET mad/resources/projects/jstestingsite/1_test/link_relative=a_10000100100001000000000000
  _test_flash_link_html_relative.html&link_absolute=http://www.tic.udc.es/~mad/resources/projects/
  jstestingsite/1_test/a_100001001000010000000000001_test_flash_link_html_absolute.html"
  ```

- Teleport realizó intentos de acceso a URLs absolutos generados mediante una función. Detectó “http...” y buscó el correspondiente “.html” que indica el final de la dirección. No obstante, no detecta que por el medio se han definido variables, y por tanto el URL generado es incorrecto.

```
"GET /~mad/resources/projects/jstestingsite/1_test/";var%20a6="a_10001001100000000000000001
_test_document_write_function_special_embedded_absolute.html"
```

III.1.7 Resultados de los Crawlers de los principales Motores de Búsqueda

Para la realización de los experimentos sobre los crawlers de los principales buscadores se registró el sitio web el 23 de Mayo del 2011 en los diferentes motores de búsqueda. Una vez el sitio web estaba disponible se esperaron 70 días antes de obtener los resultados. Tras esto se comprobaron los logs de acceso para conocer qué crawlers habían visitado el sitio web. La Tabla III.5 contiene los resultados obtenidos En esta tabla no se muestran los resultados de los crawlers Alexa, Ask, Bing, Gigablast y PicSearch, ya que dichos buscadores no habían indexado en sus repositorios el sitio web. Únicamente Yahoo! y Google indexaron el sitio web.

A continuación se identifican los factores que pueden haber provocado que el sitio web no haya sido indexado:

- Bajo PageRank [PBMW98].
- Alojamiento del sitio web en un directorio de demasiada profundidad en relación con el PageRank del sitio web.
- Alto contenido de código, lo que implica que en base a una heurística un crawler determine la posibilidad de que el sitio web contenga Malware o bien determine que los enlaces tienen una alta dificultad para ser procesados.

Google consiguió procesar un 44.29% de los enlaces, lo cual indica que no ha sido por el diseño, contenido o demás variables por las que el sitio no ha sido indexado por los demás. El crawler de Yahoo! también ha procesado algún enlace, pero de escasa dificultad.

Google ha logrado procesar un 50% de los niveles propuestos. Sin embargo, no ha sido capaz de extraer los enlaces de la mitad de los escenario en los que el código del script estaba almacenado en un fichero externo. Si Google hubiera procesado dichos escenarios, hubiera obtenido los mejores resultados. Entre los enlaces que Google no ha procesado se encuentran los incluidos en Flash, Applets, AJAX o en ficheros .class y .java.

A diferencia de los resultados obtenidos para los crawlers Open-Source y comerciales, los resultados de los crawlers de los principales buscadores eran idénticos analizando los logs de acceso del servidor web y los repositorios de los buscadores.

	Google (Extraídos — %)		Yahoo! (Extraídos — %)	
Text link	2	100.00%	1	50.00%
Href=”javaScript link	**6**	**50.00%**	0	0.00%
Document.write link	**6**	**50.00%**	0	0.00%
Menu link	**6**	**50.00%**	0	0.00%
Flash link	0	0.00%	0	0.00%
Applet link	0	0.00%	0	0.00%
Redirects	**6**	**50.00%**	2	20.00%
Class or java link	0	0.00%	0	0.00%
AJAX link	0	0.00%	0	0.00%
Links with #	**4**	**100.00%**	0	0.00%
VBScript link	**1**	**25.00%**	0	0.00%
Enlace - Estático	**17**	**40.48%**	3	7.14%
Enlace - Concatenación	**6**	**50.00%**	0	0.00%
Enlace - Función	**8**	**50.00%**	0	0.00%
Tests procesados	**31**	**44.29%**	3	4.29%

Tabla III.5: Resultados obtenidos para los crawlers de los principales motores de búsqueda

III.1.8 Comparación de Resultados

Los resultados mostrados en la Figura III.4 y en la Tabla III.5, indican que en general los crawlers Open-Source y comerciales obtienen mejores resultados que los crawlers de los motores de búsqueda. Únicamente Google consigue obtener unos resultados similares (34 escenarios procesados de un total de 70).

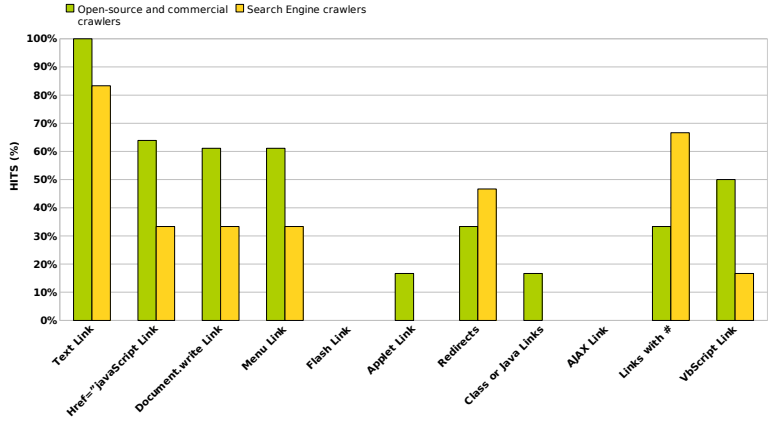


Figura III.5: Comparación de los crawlers según los tipos de enlaces procesados

Esto puede ser debido al hecho de que los crawlers Open-Source y comerciales han sido configurados por el usuario para seguir cierto tipos de enlaces que probablemente por rendimiento o seguridad los crawlers de los buscadores no procesen.

La Figura III.5 compara los resultados obtenidos en base a la tecnología utilizada en cada enlace. Una vez más los crawlers Open-Source obtienen una mayor eficacia. Observando la curva que dibuja cada grupo de crawlers, puede verse que es similar. Este hecho indica que, a pesar de que cada crawler consigue procesar un número diferente de enlaces, en general centran sus recursos en procesar el mismo tipo de enlaces, o lo que es lo mismo, la misma tecnología.

III.1.9 Clasificación de los Crawler en base a la Escala

Tras haber obtenido los resultados de cada crawler, se utilizarán dichos resultados para evaluar cada crawler con los métodos propuestos en la escala. La Figura III.6 muestra los resultados obtenidos para Simple Average, Maximum Level, Weighted Average y Eight Levels.

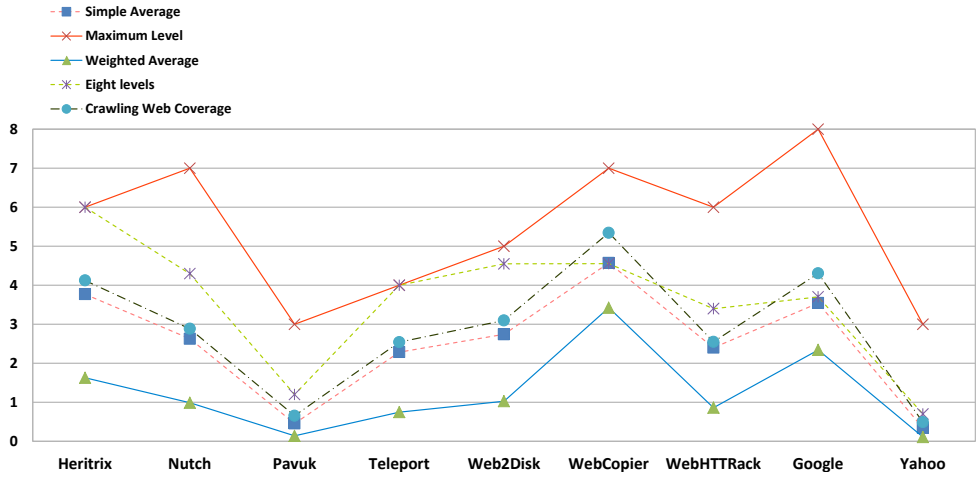


Figura III.6: Clasificación de los crawlers de acuerdo a la escala propuesta. Metodos: Simple Average, Maximum Level, Weighted Average y Eight Levels

- Para el método de Simple Average, WebCopier obtiene los mejores resultados, por delante de Heritrix y Google.
- En el método Maximum Level, Google se desmarca del resto consiguiendo procesar enlaces de nivel 8, seguido por WebCopier que obtiene un 7 y Heritrix un 6. El hecho de que Google haya alcanzado el

nivel máximo según este modelo y no en otros, indica que es probable que tenga capacidad para tratar cualquiera de los escenarios considerados, pero no lo hace debido a políticas internas.

- Para el método Weighted Average, nuevamente WebCopier, seguido de Google, Heritrix y Nutch presentan los mejores resultados.

- En Eight Levels, los primeros puestos son Heritrix, Web2Disk y WebCopier, seguido en cuarto lugar por Google. Esto indica que estos tres crawlers o bien han tratado gran parte de los escenarios de cada nivel, o bien han atravesado enlaces que formaban parte de un grupo con pocos enlaces.

Debido a que la frecuencia de uso de cada una de las tecnologías es un factor importante, también se muestra la métrica Crawling Web Coverage y los resultados obtenidos en los métodos Ranked Weighted Average y Ranked Eight Levels, que están basados en la citada métrica. WebCopier, Google y Heritrix obtienen los mejores resultados, por lo que son capaces de procesar la mayor parte de los enlaces que frecuentemente aparecen en la Web.

En la Figura III.7 se muestran los resultados obtenidos para Ranked Weighted Average y Ranked Eight Levels.

- Para el método Ranked Weighted Average, los crawlers obtienen resultados inferiores pero muy similares siendo nuevamente WebCopier y Google los primeros con cierta diferencia sobre los siguientes.

- En el método Ranked Eight Levels los resultados para casi todos los crawlers descienden entre 1 y 2 puntos.

El hecho de que algún crawler consiga buenas puntuaciones en los métodos previos y peores en estos, indica que dicho crawler tiene capacidad para procesar enlaces con una alta dificultad, pero que la frecuencia de uso de estos enlaces es muy baja en la Web.

Los resultados ponderados no sólo indican que los crawlers tienen capacidad para analizar las tecnologías del lado cliente en los sitios web, sino que esas capacidades son adecuadas para tratar los tipos de enlaces más presentes en la Web. Se puede concluir que los crawlers que más y mejor tratan la Web Oculta del lado cliente son Google y WebCopier, seguidos por Heritrix, Nutch o Web2Disk. Es importante resaltar los resultados obtenidos para GoogleBot, por tratarse de un sistema de crawling orientado a toda la Web, con grandes requisitos de rendimiento y seguridad, y que no por ello ha descuidado el tratamiento de este tipo de tecnologías.

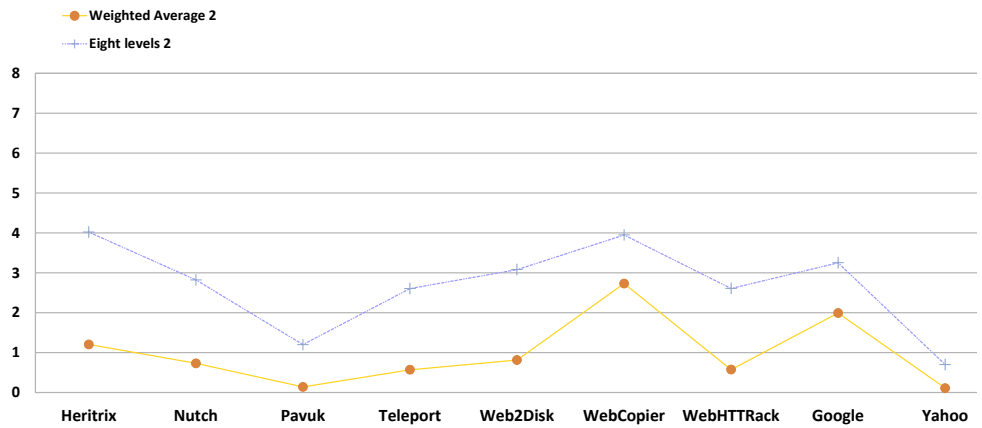


Figura III.7: Clasificación de los crawlers de acuerdo a la escala propuesta. Metodos: Ranked Weighted Average y Ranked Eight Levels

III.2 Análisis General sobre la Web Española y Global

En la presente sección se realiza un estudio acerca de la evolución de la Web Global y Española a lo largo de 3 años (2009, 2010 y 2011). Se han analizado las principales características de la Web, agrupadas en diferentes niveles y en tres períodos de tiempo distintos. El objetivo es obtener las tendencias de cambio de la Web Global y Española a lo largo de los años, comparando y analizando sus resultados. Todo ello prestando especial atención a aspectos como: el grado de similitud de la Web, la evolución de la edad de las páginas web o las tecnologías web más usadas.

La sección se estructura de la siguiente forma. En la sección III.2.1 se describe la metodología utilizada para el análisis de la Web. La sección III.2.2 explica los datasets utilizados para la realización del estudio. La sección III.2.3 contiene los resultados de la Web Global y Española a lo largo de los 3 años de estudio. Los resultados se agrupan en base a diferentes niveles de granularidad las características de la Web.

III.2.1 Metodología

El análisis de la Web puede realizarse a diversos niveles de granularidad. La Figura III.8 muestra la clasificación de niveles definida en [BI04]. De mayor a menor granularidad los niveles considerados son (incluidos en negrita en la Figura III.8):

- Palabra: El estudio de este nivel permite obtener datos sobre el vocabulario utilizado en la Web, identificar

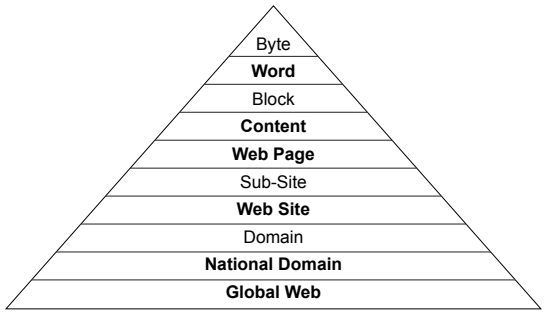


Figura III.8: Niveles de granularidad para analizar la Web [BI04]

las *stopwords* y las etiquetas HTML.

- Contenido: El análisis de este nivel permite obtener datos sobre la evolución del tamaño del contenido de las páginas web y su relación con su contenido útil real. También sirve para conocer la evolución de los lenguajes y de los formatos de los ficheros multimedia usados en la Web.

- Página web: En este nivel analizamos las características de una página web completa, como la longitud de los URLs y el nivel de compresión de las páginas. Además haremos especial énfasis en la edad y la similitud de las páginas. Estos datos nos permitirán conocer como un crawler debe manejar los URLs en memoria y en disco, cuál es la política de almacenamiento de un buscador en base al nivel de compresión, cuál es la fecha aproximada en la que una página debe ser re-crawleada y qué nivel de similitud tiene la Web.

- Sitio web: En este nivel se discutirán las características relevantes de los sitios web, definidos como colecciones de páginas web relacionadas y comunes a un dominio o subdominio. Comenzaremos analizando el número de enlaces (entrantes/salientes y estáticos/dinámicos) de la Web. Esto ayudará a conocer cómo está creciendo la Web y su estructura, y cómo esto afecta a los algoritmos de búsqueda. También estudiaremos lo que se conoce como la Web Oculta del lado servidor [RGM01], el conjunto de páginas web que no se encuentran directamente accesibles a través de enlaces. En su lugar, para acceder a ellas es necesario realizar una consulta sobre un formulario web. Esta parte de la Web contiene gran cantidad de información, pero su tratamiento presenta desafíos adicionales.

- Web nacional: El análisis de este nivel permite conocer el software usado por los servidores web y algo más relevante para los crawlers y buscadores, la evolución de los dominios en la Web Española (número de dominios de nueva creación y dominios eliminados). Este dato determinará el crecimiento de la Web

.es y ayudará a diseñar arquitecturas más adecuadas a su tamaño y evolución.

- Web global: Al igual que en el nivel de Web nacional, mostraremos un análisis sobre el uso y evolución de los servidores Web.

En todos los niveles estudiados se discuten los resultados obtenidos tanto para la Web Global como para la Española, comparando y analizando las similitudes que existen entre ellos.

Los demás niveles de la Figura III.8 (byte, word, block, sub-site, domain) quedan fuera del alcance de este artículo, ya que las conclusiones de estos niveles pueden ser obtenidas con el estudio de los niveles considerados.

III.2.2 Datasets

En esta sección se describen los datasets utilizados para la realización del estudio, y cómo ha sido obtenido cada uno de ellos.

El dataset de la Web Española considera sitios web alojados bajo el dominio .es, independientemente del lenguaje usado en estos o del país donde estuviera alojada la IP del servidor web. El estudio se orienta a conocer la Web Española y Global con independencia del lenguaje usado. Es importante resaltar que el estudio no incluye a la social media Web (Vimeo, Flickr, etc.), ya que aunque éstas están en continuo crecimiento, no representan la realidad de la Web de un país. Esto es debido a que plataformas como Vimeo o Flickr utilizan los mismos formatos o tecnologías independientemente del país.

Para generar el dataset de la Web Española, se comenzó con un conjunto de dominios como semilla y se configuró el sistema para que no permitiera procesar páginas externas al dominio .es. Este proceso se repitió desde el 2009 una vez al año, obteniendo 3 datasets: a) 2009 con 577542 documentos, b) 2010 con 785000 documentos y c) 2011 con 1050000 documentos. Tras esto conseguimos un dataset total de aproximadamente 2,5 millones de páginas web.

Para el análisis de la Web Global se utilizaron los datos que proporciona "The Stanford WebBase Project" [19], que forma parte del proyecto "Stanford Digital Libraries Project" [20]. El dataset completo contiene más de 260 TB de datos, organizado en subconjuntos de diferentes temáticas (general, desastres naturales, gobiernos, etc.). Para nuestro estudio únicamente se considerarán los datasets de temática generalista de los 12 meses correspondientes a 2009, 2010 y 2011. De los más de 700 millones de páginas obtenidas, se realizó un muestreo aleatorio para obtener un subconjunto de 10 millones de páginas representativo de cada año analizado. Finalmente, el dataset Global analizado contiene aproximadamente 30 millones de páginas web.

[19] http://dbpubs.stanford.edu:8091/~ testbed/doc2/WebBase/
[20] http://diglib.stanford.edu:8091/

	Etiquetas HTML
2009	body, head, html, root, title, meta, script, img, div, table, tr, td, p, link, br, claris, tagview, window, span, style, object, param, embed, h1, font, form, strong, input, li
2010	body, head, html, root, title, meta, script, img, div, p, link, br, table, tr, td, span, style, li, ul, h1, form, strong, object, param, input, embed, h2, b, font
2011	body, head, html, root, title, meta, script, img, div, p, link, br, table, tr, td, span, style, li, h1, ul, form, strong, object, param, input, embed, h2, b, font

Tabla III.6: Etiquetas HTML más utilizados

III.2.3 Resultados de la Evolución de la Web

Partiendo de los datasets definidos, en los siguientes apartados se discutirán los resultados obtenidos tras el análisis de las diferentes características contempladas en el estudio, agrupados en base a su nivel de granularidad.

Características a Nivel de Palabra

En el ámbito de las páginas web, una palabra puede ser usada principalmente como término o como etiqueta de HTML. En esta sección se analizan las palabras de una página desde esos dos puntos de vista.

- Vocabulario: Para el estudio del vocabulario usado en la Web, se ha considerado una palabra como cualquier secuencia alfanumérica de longitud mayor a 1 carácter (ignorando las etiquetas HTML). Tanto en los resultados obtenidos en la Web Global como los resultados de la Web Española, se ha observado que la cantidad de páginas que contienen términos cuya frecuencia de uso es baja, está aumentando.

  Esto indica que ha crecido el número de palabras comunes en las páginas web. Desde el punto de vista de los buscadores esto hace que haya aumentado el número de términos que no sirven para representar el contenido de una página web sobre las demás, dificultando la selección de documentos relevantes ante un conjunto de términos.

- Uso de etiquetas HTML: Analizando la Web Global y la Española, se ha detectado que las 50 etiquetas HTML más usadas son comunes en los 3 años (a excepción de pequeños cambios). En la Tabla III.6 se muestan las etiquetas HTML más habituales durante los años estudiados. Los resultados indican que se han producido escasos cambios a la hora de desarrollar páginas web. Por otro lado, se han encontrado gran cantidad de etiquetas HTML incorrectas que dificultarán la tarea al explorador provocando que probablemente no se muestre correctamente la página.

Características a Nivel de Contenido

En este apartado se analiza la evolución del tamaño del contenido total/útil de las páginas, los charset usados, los idiomas que utilizan las páginas, un resumen de los formatos de archivos de audio, vídeo e imágenes más extendidos, y un análisis sobre el uso de ciertos atributos de la etiqueta Meta.

- Tamaño del contenido total/útil: Un dato importante para los buscadores y crawlers es el tamaño del contenido que descargan y almacenan, y su relación con el contenido útil de cada página. El contenido útil de una página web es el contenido principal, donde está la información relevante tras eliminar las etiquetas HTML, imágenes, etc. El contenido útil es aquel que usan los motores de búsqueda para localizar documentos relevantes ante una búsqueda de un usuario. El proceso de extracción de contenido útil de una página web es complejo. En nuestro caso, para obtener el contenido útil hemos seguido la aproximación de Donghua *et al.* [DP08]. Dicha aproximación supone que el contenido útil suele encontrarse centrado en una página y presenta una estructura jerárquica. Para diferenciar aquellos nodos DOM [w3d11] con contenido útil de los demás nodos, los autores proponen un algoritmo que evalúa el contenido de cada nodo en base a 4 parámetros: densidad de enlaces en el texto, cantidad de enlaces, densidad de enlaces en cada nodo y longitud del texto de cada nodo. Tras esto, se determinarán una serie de umbrales que permitirán diferenciar un nodo con contenido útil de los demás. Cuándo la densidad de enlaces en un nodo sea muy alta en relación al texto que contiene y a la cantidad de enlaces totales de la página y de los demás nodos, ese nodo se considerará que no tiene contenido útil. En otro caso, el contenido del nodo será considerado útil.

  La Figura III.9 muestra los resultados obtenidos. Para el proceso de descarga de las páginas se ha considerado el contenido completo de la página (a diferencia de otros estudios existentes que truncan las páginas a un cierto tamaño [BYC04]).

  En la Web Global (Figura III.9 a, b, c) en el año 2009 el contenido total medio de las páginas era de 28.18 KB. Esta cifra ha disminuido en 2010 hasta 24.1 KB y el 2011 hasta 21.4 KB. Esto supone una reducción en 3 años de 24.06%. Analizando los resultados del tamaño del contenido útil, vemos que en 2009 el contenido útil medio era de 6.49 KB, cifra que no cambió demasiado en los sucesivos años, siendo de 6.03 KB en 2011. Esto indica que el tamaño del contenido útil de las páginas no ha variado a lo largo de los años y que la relación entre tamaños de contenido total y útil representa aproximadamente un 28% del total.

  La Web Española (Figura III.9 d, e, f) tenía un tamaño medio de páginas de 9.98 KB en 2009, en 2010 creció hasta los 11.8 KB y en 2011 hasta los 13,4 KB. En los 3 años se observa que la mayor parte de las páginas tienen un tamaño de 10 a 500 KB, y que existen algunos casos de páginas que llegan hasta los 5 MB. También hemos estudiado el contenido útil de las páginas y su relación con el contenido total. En lo que respecta al contenido útil, en 2009 vemos que el tamaño medio del contenido útil de las páginas

es de 5.52 KB, en 2010 creció hasta los 6.27 KB y en 2011 hasta los 6.31 KB. En 2011 el contenido útil frente al total representaba un 47.08%.

Estos datos indican que el contenido total de las páginas web españolas tiene menos de la mitad de contenido que las páginas del resto de la Web. Se ha observado que el contenido total de la Web Global se está reduciendo frente al de la Web Española, que está aumentando.

Como discute Downey [Dow01], las diferencias observadas en la Web Global y Española entre el contenido total y el útil, pueden ser debido a las diferencias en el uso del HTML para codificar las páginas web, y a la tendencia a que las páginas web sean más complejas.

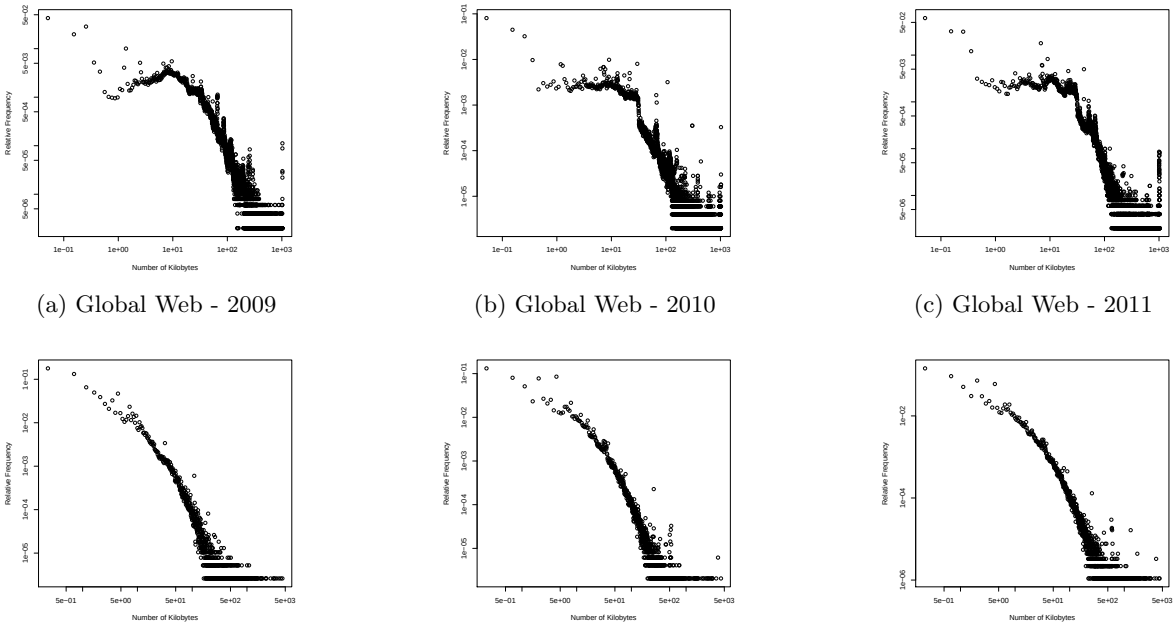
Comparando los resultados obtenidos con los de otros trabajos existentes en la literatura, se puede decir que las páginas web, tanto la Web Global como la Española, tienen más contenido útil que las páginas web de Portugal [GS05a]. Concretamente, en 2009 la Web Española y Global tenían un 5.52 KB y 6.49 KB de contenido útil, respectivamente, frente a los 2.8 KB de la Web Portuguesa [GS05a], es decir, aproximadamente la mitad.

En resumen, se ha observado que aunque el tamaño de las páginas web a nivel global está decreciendo, y las de la Web Española está aumentando, en 2011 la media de tamaño en la Web Global era aproximadamente el doble que en la Web Española. Centrándose en el contenido útil, se puede decir que no existen diferencias entre el tamaño del contenido útil de las Webs estudiadas. Sin embargo, la relación entre el contenido útil y el contenido total es menor en la Web Global que en la Web Española, en la cual el contenido útil representa el 50% del contenido total. Este hecho puede ser debido a que las páginas de la Web Global incluyen gran cantidad de código HTML, lo cual ocurre probablemente por el uso de tecnologías del lado cliente que mejoran la experiencia del usuario. Desde el punto de vista de los motores de búsqueda, prefieren páginas web con una alta relación entre contenido útil y contenido total, ya que, de esta forma los sistemas de crawling necesitan procesar menor cantidad de código HTML para obtener el mismo contenido útil.

De los resultados obtenidos, se puede concluir que la relación entre el contenido total y el contenido útil de las páginas web españolas tiene significantes diferencias con las páginas de la Web Global. Este hecho es importante a la hora de optimizar sistemas de almacenamiento a gran escala, tales como los usados por los sistema de crawling o los motores de búsqueda. Adaptando las políticas de almacenamiento de los motores de búsqueda a las necesidades y evolución de cada país se lograría ahorrar gran cantidad de recursos.

- Lenguaje: Para identificar el lenguaje usado en cada página se ha usado la librería "language detector" [BYC10] que utiliza filtros Bayesianos para determinar el lenguaje de un texto en base a un conjunto de reglas. Esta librería tiene una precisión del 0.99 en los 53 lenguajes que detecta.

  Los resultados obtenidos a nivel global y de España, se muestran en la Figura III.10. A nivel global



(a) Global Web - 2009 (b) Global Web - 2010 (c) Global Web - 2011

(d) Spanish Web - 2009 (e) Spanish Web - 2010 (f) Spanish Web - 2011

Figura III.9: Tamaño del contenido de una página web

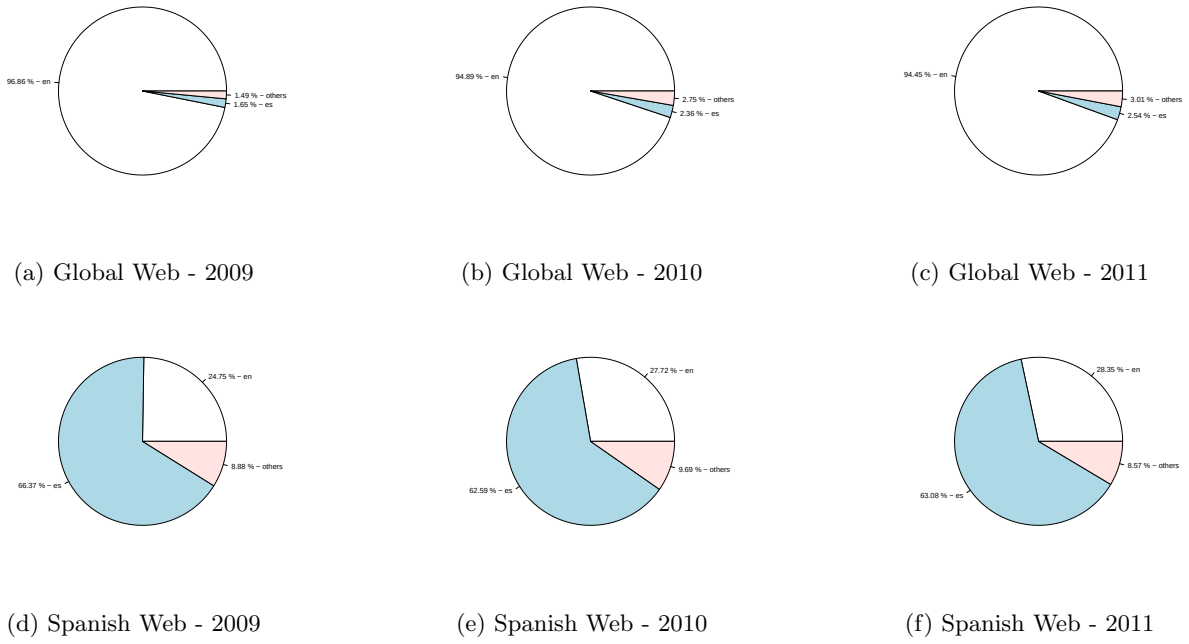
(Figura III.10 a, b, c) se observa que el idioma predominante es el inglés con un 96.86% de ocurrencias en 2009, reduciéndose en 2011 al 94%. En el estudio presentado por Grefenstette y Nioche [GN00] en el año 2000, los autores estiman que aproximadamente el 70% de la Web está escrita en inglés. De acuerdo con los datos obtenidos, esta cifra ha aumentado un 16.86%.

A nivel de la Web española, Figura III.10 (d, e, f), observamos que en 2009 el español representaba un 63.08% del total, y el inglés un 28.35%. No obstante se puede observar que desde 2009 a 2011 ha disminuido aproximadamente un 3% el uso del español frente al inglés. Esto puede ser debido a la apertura a nuevos mercados de la economía española, que ha hecho que muchos contenidos y sectores comiencen a trabajar y escribir en inglés. El resto de los lenguajes presentes en la Web Española representan el 8.57%, entre los cuales aparecen los principales lenguajes europeos (italiano, alemán, francés, portugués, etc.).

En 2011 el porcentaje de páginas web escritas en español en la Web Española era de aproximadamente el 63%. Sin embargo, esta cifra es inferior a las obtenidas en otros países con respecto a su lengua oficial. En la Web Portuguesa aproximadamente el 73% de las páginas web están escritas en portugués [GS05a],

en la Web Brasileña esta cifra aumenta hasta el 75% [dSVG+99] y en la Web Chilena hasta el 90% las páginas escritas en castellano. Por otro lado, vemos en la Web Tailandesa [SPnK+00] que el 66% de las páginas web están escritas en inglés y que en diversos países de África [BBC+08] esa cifra aumenta hasta el 75%.

Basándose en estos datos, se puede observar que la Web Española está formada por páginas con mayor diversidad de lenguajes que las Webs de otros países. Esto puede ser debido a la internacionalización, hecho que es una ventaja, ya que usuarios de diversos países con diferente lenguaje podrán acceder igualmente al contenido. Los motores de búsqueda tienen en cuenta este hecho a la hora de valorar una página web.



(a) Global Web - 2009 (b) Global Web - 2010 (c) Global Web - 2011

(d) Spanish Web - 2009 (e) Spanish Web - 2010 (f) Spanish Web - 2011

Figura III.10: Distribución de los idiomas más usados en la Web Global y española

- Keywords: Las keywords son aquellas palabras contenidas en el atributo "keywords" de la etiqueta Meta de HTML, que describen el contenido tratado en la página. Hemos analizado el número medio de keywords en la Web Global y en la Española. En el primer caso hemos observado que en 2009 la media de keywords era de 8,37 por página en 2010 de 9,21 y en 2011 de 8,89. En los 3 años el número de keywords se

ha mantenido en torno a 9. En el caso de la Web Española, el número medio de keywords ha mostrada mínimas variaciones durante los años estudiados, permaneciendo entorno a 15 keywords por página web.

Comparando ambos resultados vemos que en las páginas web españolas está más extendido el uso de keywords, que en la Web Global. Esto puede ser debido al hecho de que ciertos lenguajes tienden a usar más palabras que otros.

En base a estudios realizados previamente por Prieto *et al.* [PALGC12b], el uso intensivo de keywords puede ser un indicador de una página web de Spam o Soft-404. Un número lógico de palabras para definir la temática de una página web debería de estar comprendido entre 7 y 15. Sin embargo, ciertos lenguajes usan un mayor número de palabras que otros. Por tanto, los motores de búsqueda debería conocer la media de keywords por país, y penalizar a aquellas páginas web que abusen de su uso.

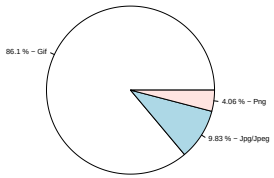
- Tipos de formatos de archivos de audio: Se han estudiado los formatos de archivos de audio más usados en la Web. En este tipo de ficheros hemos observado que durante los 3 años analizados casi no ha variado la presencia de los diferentes formatos en la Web. En 2011, tanto en el caso de la Web Global como de la Española, el formato más extendido es el MP3 con un 92% de ocurrencia y un 86.06%, respectivamente. En la Web Global y Española el siguiente formato de audio más usado es el WAV con un 4.39% y un 6.4% de aparición. Otros formatos que también hemos encontrado son WMA con un 5.71% en la Web Española y MIDI, ASF, con porcentajes meramente simbólicos. La poca evolución que se ha observado es debido a la buena calidad del formato MP3 en un tamaño relativamente pequeño, y a que no ha aparecido ningún nuevo formato en los últimos años que haya hecho disminuir el dominio del MP3.

- Tipos de formatos de imagen: La Figura III.11 muestra los cambios en el uso de los diferentes formatos de archivos de imágenes.

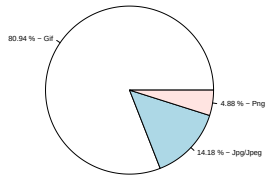
  A nivel global se observa que hay un predominio del formato GIF con un 74.95% de ocurrencias en 2011, frente a los demás formatos. Tras GIF los formatos de imagen más usados en 2011 son JPG con un 14.36% y PNG con un 10.7%. En la Web Española la presencia de diferentes formatos de imagen está más distribuido. A pesar de esto, el formato GIF sigue siendo el más usado con un 45.42%, aunque tanto JPG como PNG han ido ganando terreno. En 2009 el 29.06% de las páginas usaba JPG frente al 34.5% actual. Lo mismo ha pasado con PNG, pero más significativamente, ya que ha aumentado de un 7.65% a un 20.02% en 2011.

- Tipos de formatos de archivos de vídeo: La Figura III.12 muestra los formatos de vídeo más utilizados. En la Web Global (a, b, c) se observa que en 2009 el formato predominante era WMV con un 59.5%, seguido de MOV con un 37.36% y de AVI con un 2.74%. En 2010 WMV aumentó su presencia hasta el 71.46%. En 2011 se igualó el uso de WMV y MOV, con un 54% y un 42.54%, respectivamente.

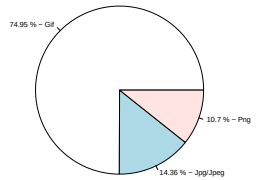
  En la Web Española (d, e, f) vemos que el formato predominante también es WMV con un 76.56% en 2009 y un 70.27% en 2011. Esa pérdida de porcentaje es debido al aumento de uso del formato MOV,



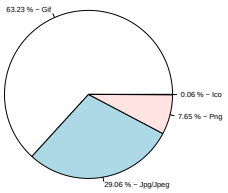
(a) Global Web - 2009



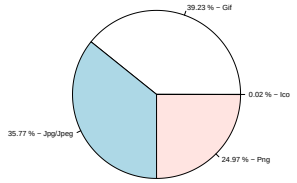
(b) Global Web - 2010



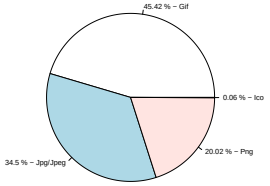
(c) Global Web - 2011



(d) Spanish Web - 2009



(e) Spanish Web - 2010



(f) Spanish Web - 2011

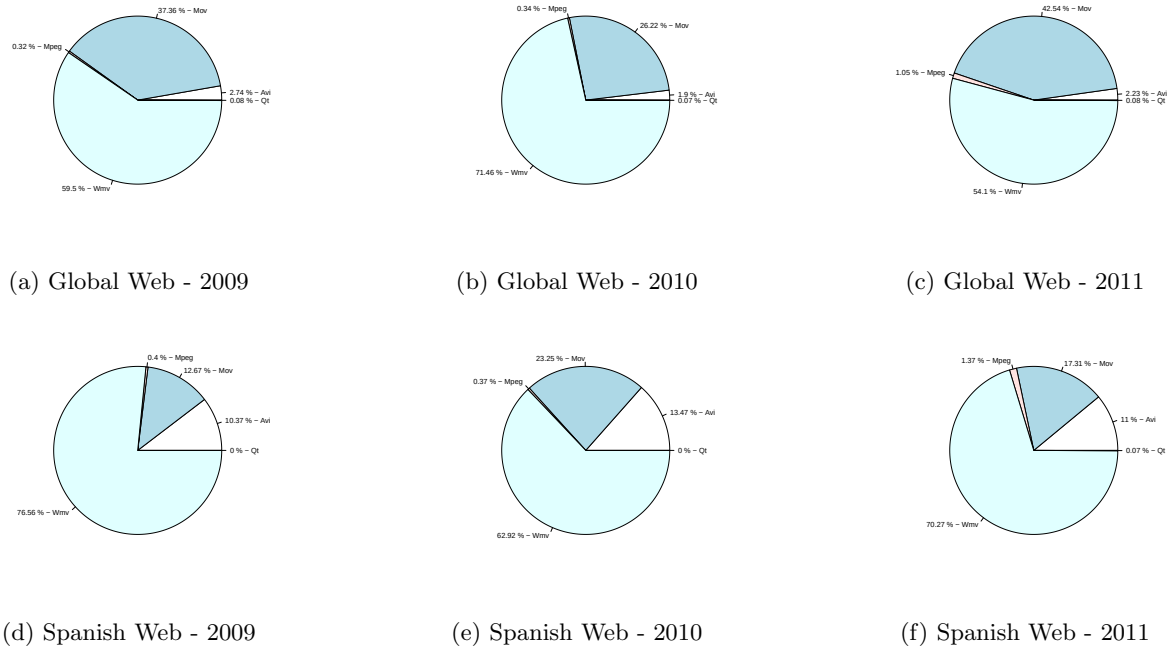
Figura III.11: Distribución de los formatos de ficheros de imágenes

que en 2011 se usaba en un 17.31% de los casos. El formato AVI vemos que a pesar de variar a lo largo de los 3 años, se mantiene por encima del 10%.

El formato AVI fue uno de los primeros formatos de vídeo creados. Se caracteriza por su buena calidad, pero esto provoca que sean muy pesados. En años anteriores era el formato más extendido en Internet. Hoy en día, formatos como WMV permiten obtener calidades similares a AVI con tamaños mucho menores. Debido a esto, el formato AVI está casi desapareciendo de las páginas web, en favor de formatos menos pesados.

En resumen, vemos que en la Web Global el uso de los diferentes formatos de vídeo está más distribuido que la Web Española, en la que principalmente se usa el formato WMV.

- Estilos: El hecho de que una palabra aparezca resaltada (negrita o cursiva) indica que es más relevante para esa página que otras. Para tener este hecho en cuenta, los buscadores en el proceso de indexación de los datos almacenarán también el estilo con el que se escribieron ciertos términos del contenido. Por ello se ha realizado un breve estudio sobre los estilos estándar comúnmente usados. Los resultados se



(a) Global Web - 2009 (b) Global Web - 2010 (c) Global Web - 2011

(d) Spanish Web - 2009 (e) Spanish Web - 2010 (f) Spanish Web - 2011

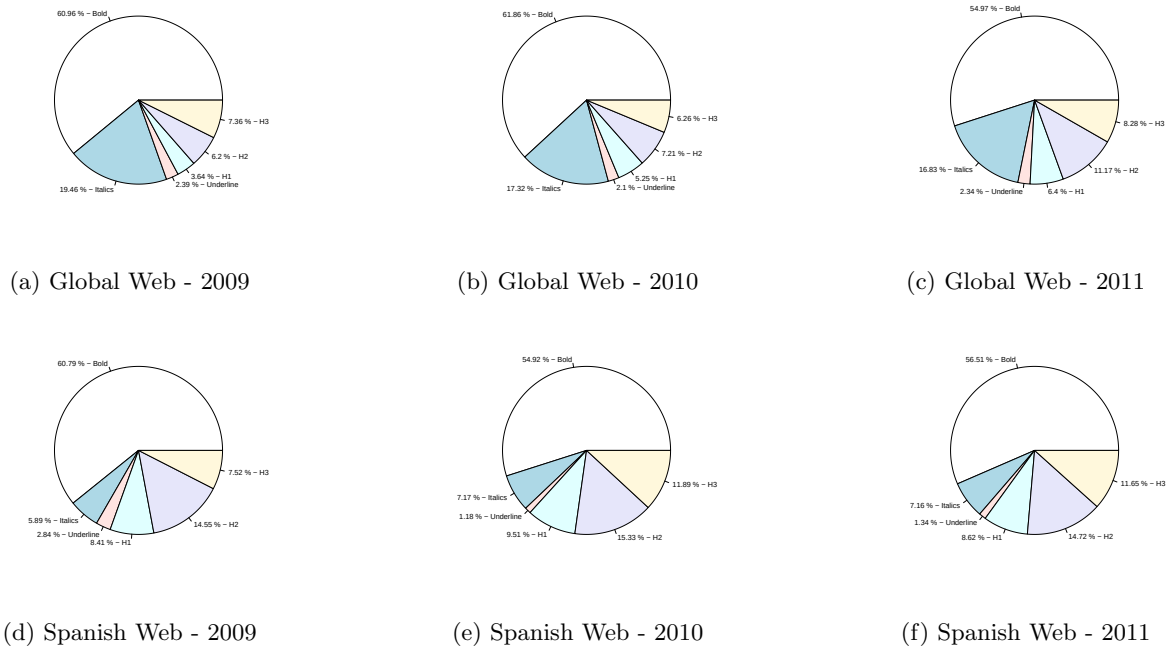
Figura III.12: Distribución de los tipos de ficheros de vídeo en la Web Global y española

muestran en la Figura III.13.

A nivel Global se observa que el estilo más usado es la negrita, con un 60.96% de uso en 2009, pero que ha caído aproximadamente un 5%. El uso de la cursiva también se ha reducido aproximadamente un 3% desde 2009, pasando de un 19.46% a un 16.83%. En base a los resultados vemos que el descenso de la negrita y la cursiva, es debido al aumento en el uso de los estilos H1, H2 y H3 desde 2009 a 2011.

Analizando los resultados referentes a la Web española, Figura III.13 (d, e, f), podemos observar que el estilo más usado es la negrita con un 56.51%, seguido de los estilos usados para encabezados H1 con 8.62%, H2 con 14.72% y H3 con un 11.65% del total de los estilos considerados.

Los resultados obtenidos tanto en la Web Global como en la Española, son lógicos ya que un texto puede tener muchas palabras en negrita, menos en H1, H2 y H3. Por otro lado, se observa que tanto en la Web Global como en la Española, el estilo más utilizado es la negrita. En concreto, la cursiva se usa más del doble en la Web Global frente a la Española, pero en la Web Española se usan más los estilos H1, H2 y H3 referentes a los títulos de secciones.



(a) Global Web - 2009 (b) Global Web - 2010 (c) Global Web - 2011

(d) Spanish Web - 2009 (e) Spanish Web - 2010 (f) Spanish Web - 2011

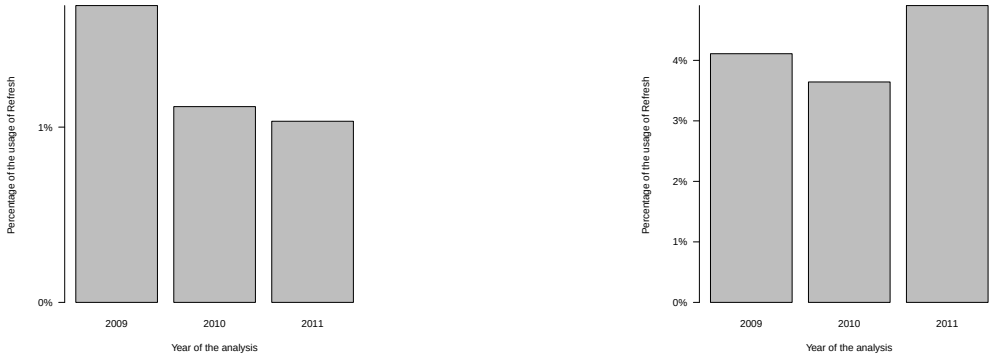
Figura III.13: Distribución de los tipos de estilos presentes Web Global y Española

- Atributos de la etiqueta Meta: Una parte importante de la información sobre el contenido de una página web se encuentra en la etiqueta HTML Meta. Estas etiquetas se sitúan al inicio del código HTML y ofrecen información sobre una página al usuario, al explorador y a los crawlers y buscadores. Existen diferentes atributos de la etiqueta Meta, no obstante, el estudio analiza únicamente 2, por considerarlos los más relevantes.

  - Refresh: Este atributo indica cuándo debe refrescarse el contenido de la página. En la Figura III.14 se muestran los resultados obtenidos en la Web Global y en la Española. En la Web Global (a) en el año 2009 un 1.6% de las páginas usaban el atributo refresh. En 2010 y 2011 descendió un 0.5% hasta casi el 1%.

    En la Web española se observa que el uso del atributo refresh es más habitual. En 2009 y 2010 el porcentaje de uso era similar, en torno al 4%. En el 2011 el valor subió hasta el 4.9%.

    A pesar del aumento de páginas dinámicas que obligan a un continuo refresco del contenido, el atributo refresh ha disminuido a nivel global y ha subido solo un 1% en la Web Española. Esto



(a) Global Web (b) Spanish Web

Figura III.14: Uso del atributo refresh en la Web Global y Española en 2009, 2010 y 2011

puede ser debido a que existen otros métodos para refrescar el contenido, total o parcial, de la página de forma transparente para el usuario, mediante llamadas JavaScript. Este tipo de métodos son muy utilizados en páginas con AJAX, que actualizan su contenido sin que el usuario lo perciba.

– Content-type: Atributo que indica el tipo de contenido y el conjunto de caracteres utilizado para su codificación.

  Los resultados obtenidos se muestran en la Tabla III.7. Tanto en la Web Global como en la española, se observa que en 2009 el charset más utilizado era ISO-8859-1. En 2009 el uso de ISO-8859-1 en páginas representaba un 57.74% en la Web Global y un 61.5% en la Española. En los años 2010 y 2011 el uso de ISO-8859-1 ha ido disminuyendo hasta llegar al 38.71% en la Web Global y el 41.07% en la Española. Esta disminución viene provocada por el aumento en el uso de UTF-8, que en 2009 en la Web Global se usaba un 34.68% y en la Española un 27.73%. En 2010 y 2011 ha aumentado su presencia hasta un 51.51% en la Web Global y un 51.53% en la Web Española.

  El aumento en el uso de UTF-8 y la disminución de ISO-8859-1, es debido a la necesidad de nuevos tipos de codificación que permitan soporte multilenguaje. El uso de UTF-8 en las páginas web permite que se visualicen correctamente independientemente del charset utilizado en cada ordenador. Sin embargo, este hecho tiene una desventaja, ya que el tamaño de las páginas web con UTF-8 es mayor que con codificación ISO, ya que la codificación ISO usa un byte por carácter y UTF-8 usa 2 bytes para ciertos caracteres especiales.

Web Global					
	ISO-8859-1	UTF-8	ISO-8859-15	windows-1252	Otros
2009	57.74%	34.68%	0.57%	5.51%	1.91%
2010	43.23%	43.75%	0.64%	7.52%	4.86%
2011	38.71%	51.51%	0.61%	7.15%	2.02%
Web Española					
	ISO-8859-1	UTF-8	ISO-8859-15	windows-1252	Otros
2009	61.5%	27.73%	0.53%	6.92%	3.32%
2010	31.82%	63.34%	0.60%	3.56%	0.68%
2011	41.07%	51.53%	1.25%	4.42%	1.73%

Tabla III.7: Charsets más utilizados en la Web Global y Española

Características a Nivel de Páginas Web

Esta sección se centra en las características propias de una página web, haciendo especial énfasis en el estudio de la longitud de los URL, la edad de las páginas y su similitud, puesto que son características muy importantes desde el punto de vista de los crawlers.

- Longitud de URLs: Conocer la longitud de los URLs es muy importante ya que de ese modo se podrán mejorar los esquemas de compresión utilizados para el caching e indexación de la Web. Por ello, este aspecto es muy relevante desde el punto de vista de los sistemas de crawling y los motores de búsqueda.

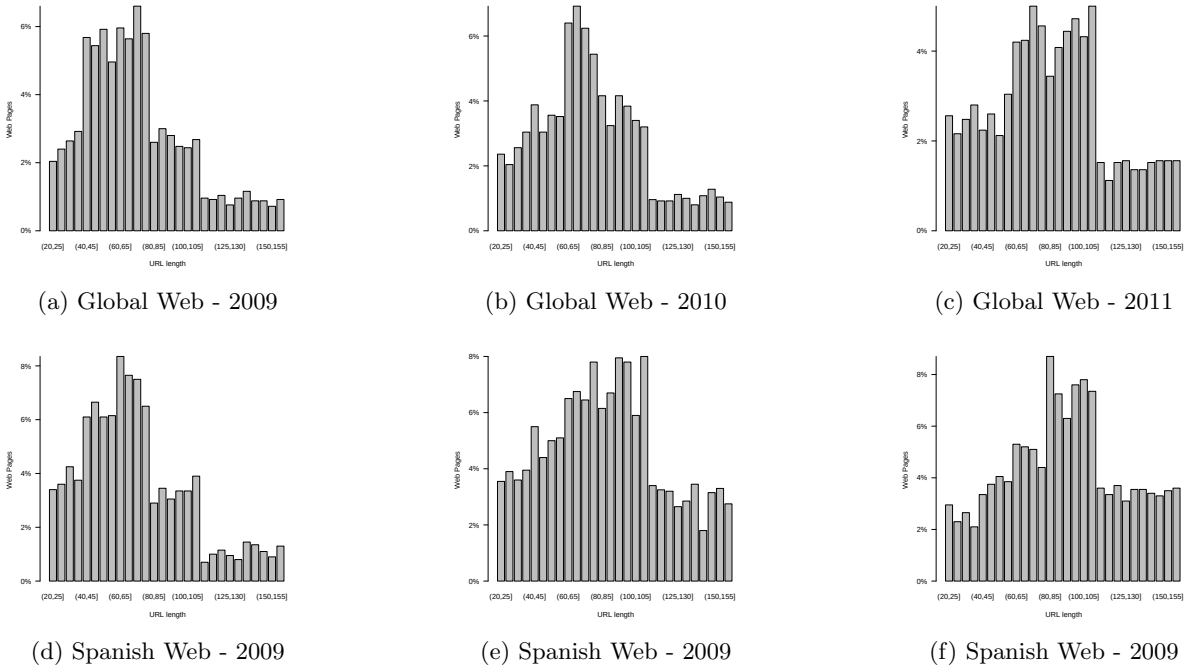
  La Figura III.15 (a, b, c) muestra la longitud de los URLs en bytes para la Web Global. En el año 2009 la longitud media era de 70.88 bytes. En los años 2010 y 2011, la longitud media creció hasta 77.2 y 84.66 bytes, respectivamente. Analizando la evolución se observa que la longitud de URLs creció y que el número de URLs con longitud entre 100 y 150 bytes aumenta de forma continuada.

  Los resultados relativos a la Web Española se pueden ver en la Figura III.15 (d, e, f). En el año 2009 gran parte de las páginas web contenían URLs de tamaño entre 45 y 75 bytes, y una pequeña parte tenía URLs de más de 100 bytes. En 2010 la mayor parte de las páginas tenían URLs con un tamaño de entre 65 y 100 bytes. Finalmente, en 2011, la mayor parte de las páginas tienen URLs entre 80 y 110 bytes y ha aumentado el grupo de páginas con URLs de entre 100 y 150 bytes.

  Tanto en la Web Global como en la Española, se observa que la longitud de las URLs ha crecido desde el año 2009. Esto es debido al crecimiento de las páginas dinámicas, nuevas tecnologías y la necesidad de enviar en muchas ocasiones parámetros dentro de la URL. Este cambio en las URLs implica e implicará cambios en el diseño de los sistemas de caché y almacenamiento de URLs de los crawlers, tanto para las

colas de URLs visitadas como para las colas de URL a visitar.

Comparando los resultados obtenidos con los resultados mostrados en otros estudios, se ha observado que la longitud de las URLs en la Web Española es similar a la observada en la Web Portuguesa, la cual tiene una longitud media de 62 caracteres. Por otro lado, comparando los resultados obtenidos de la Web Global con los presentados por Suel y Yuan [SY01], se puede ver que la longitud media de las URLs fue en 2001 50 caracteres menor que en 2011.



(a) Global Web - 2009 (b) Global Web - 2010 (c) Global Web - 2011

(d) Spanish Web - 2009 (e) Spanish Web - 2009 (f) Spanish Web - 2009

Figura III.15: Tamaño de la longitud de las URLs

- Título de las páginas web: El título es uno de los elementos más representativos en una página web. El uso de títulos descriptivos es importante para la usabilidad de la Web, ya que permitirá a los usuarios conocer de forma rápida la temática de la página web. Para el análisis de los títulos de las páginas web, hemos considerado su longitud media, para de esta forma poder determinar su importancia real a la hora de describir el contenido de una página.

  Analizando los resultados para la Web Global se observa que, en 2009, la media de la longitud del título de las páginas era de 7.26 palabras, en 2010 de 6.94 y en 2011 de 7.11. A grandes rasgos se puede decir

que el valor medio se ha mantenido entorno 7 palabras. En los resultados de la Web Española se observa que no ha habido cambios relevantes desde 2009 hasta 2011 ya que la curva que describen es muy similar en los 3 casos. La media de longitud es aproximadamente de 5 palabras, existiendo páginas con una sola palabra y otras con hasta 35 palabras.

Tanto en la Web Global como en la Española el número de palabras se ha mantenido relativamente constante a lo largo de los años estudiados. No obstante vemos que los títulos de las páginas de la Web Global son de mayor tamaño que las españolas.

Los motores de búsqueda usan las palabras del título para definir la relevancia de una página web en una temática particular. Los buscadores dan más relevancia a las palabras que aparecen en el título que las que aparecen en el texto. Analizando los resultados se ha observado que existe un conjunto de páginas web con gran cantidad de palabras en el título. En muchos casos esto es debido a páginas de Spam o Soft-404 que tratan de mejorar su ranking en los buscadores. Por ello, los buscadores deberían detectar y penalizar a aquellas páginas web que cuyo número de palabras en el título es bastante superior a la media.

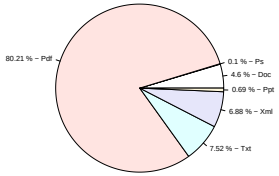
- Otro tipo de documentos: Un punto que caracteriza a las páginas web son los diferentes tipos de documentos que contiene. En la Figura III.16 se muestran los resultados obtenidos para los tipos de documentos más relevantes.

  Analizado los resultados, vemos que tanto en la Web Global como en la española, los documentos no HTML que más aparecen son los PDF (Portable Document Format). En 2011, a nivel global el 80.87% eran PDF y a nivel español era el 86.86%, un 5% mayor. A nivel global el siguiente tipo de documento más utilizado son los DOC, que creció desde 2009 con un 4.6% hasta un 7.91% en 2011. El uso de ficheros TXT en 2009 era de un 7.52%, pero ha disminuido a un 3.03% en 2011. Por último comentar la reducción del uso de los documentos XML en 2010, que pasó de un 6.98% en 2009 a un 2.91% en 2010, aunque en 2011 volvió a aumentar hasta el 5.16%.

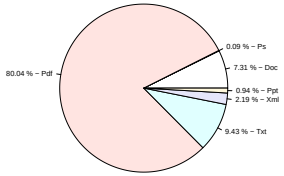
  En la Web Española, los documentos XML y DOC tienen menos presencia que los documentos en formato PDF, con una incidencia del 8.64% y del 3.23% en el año 2011. El hecho de que Microsoft Windows sea el sistema operativo más extendido [wik13], ayuda a incrementar el número de documentos Office tales como ficheros DOC.

  Los estudios realizados sobre la Web de otros países también muestran que el formato PDF es el más extendido. Concretamente, en la Web Brasileña [MPZ+05] los documentos PDF tienen una presencia del 48%, en la Web Chilena [BYC04] de aproximadamente el 63% y en la Web Portuguesa [GS05a] del 46%.

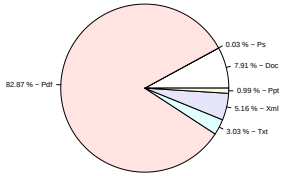
  En resumen, vemos que los resultados obtenidos son lógicos, ya que los documentos PDF pueden ser usados con independencia del sistema operativo utilizado.



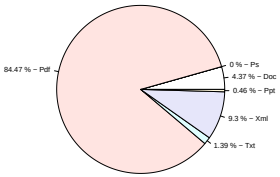
(a) Global Web - 2009



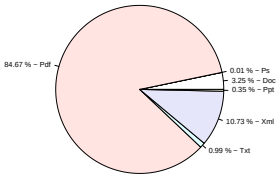
(b) Global Web - 2010



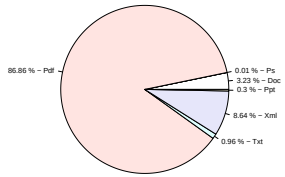
(c) Global Web - 2011



(d) Spanish Web - 2009



(e) Spanish Web - 2010



(f) Spanish Web - 2011

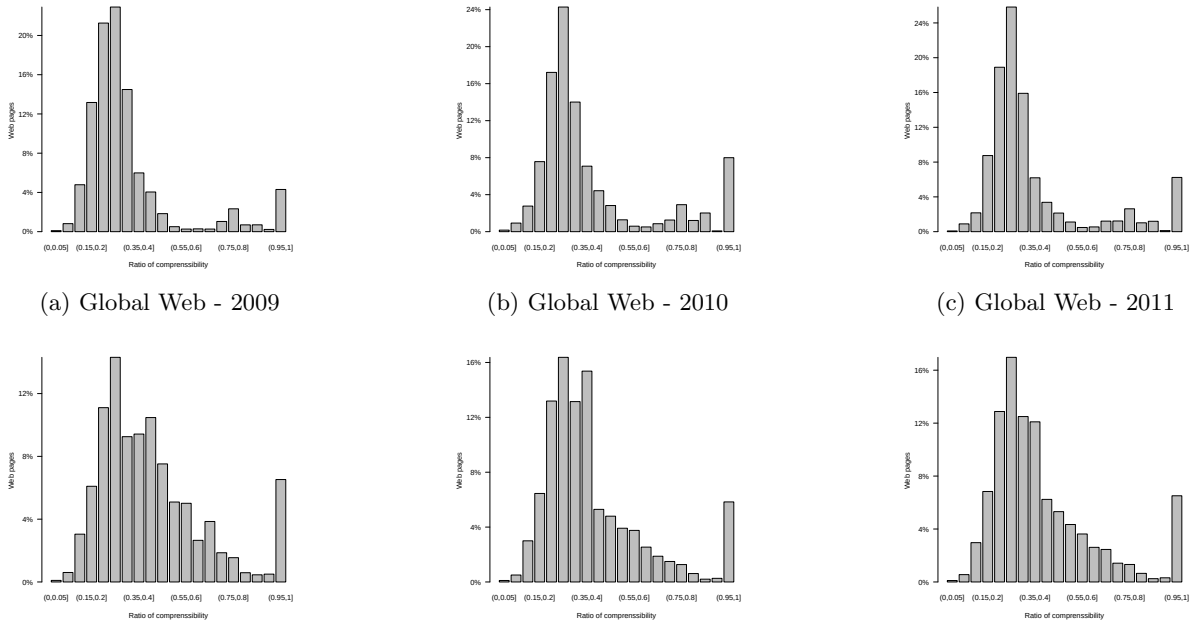
Figura III.16: Distribución de los tipos de documentos presentes Web Global y Española

- Compresión del contenido: Indica el nivel de similitud del contenido dentro de una misma página web, ya que contenidos repetidos provocarán un mayor nivel de compresión. Con un mayor nivel de comprensión el contenido será menos diferente y por lo tanto en muchas ocasiones tendrá una menor calidad. Su análisis permite a los buscadores definiar políticas de almacenamiento adecuadas y detectar contenidos y sitios web repetidos o de escasa calidad. La Figura III.17 muestra los resultados obtenidos.

  Analizando los resultados obtenidos, vemos que en la Web Global en 2009 el nivel de compresión era de 0.32, valor que creció en los sucesivos años hasta 0.38 en 2010 y 0.36 en 2011. En lo que respecta a la Web Española, podemos observar que en 2009 el nivel de comprensión medio era de 0.43, y que en los dos años sucesivos, 2010 y 2011, dichos niveles se redujeron a valores entre 0.38 y 0.21. Estos resultados indican que en la Web Global a pesar de aumentar y mejorar el contenido ha mejorado el nivel de comprensión, lo cual no ocurre en la Web Española, en donde en cada período se ha ido reduciendo el nivel de comprensión.

  En resumen, vemos que el nivel de compresión es mayor en la Web Global que en la Web Española. Esto

nos indica que los contenidos de las páginas de la Web Global son más repetitivos que las páginas de la Web Española. Este hecho también provoca que el coste de almacenamiento de los contenidos de la Web Global sea menor. Basándose en estos resultados, los motores de búsqueda deberían modificar sus políticas de almacenamiento para adaptarse a los requisitos particulares de cada país, consiguiendo de este modo dimensionar mejor los recursos requeridos y mejorar la calidad de sus contenidos almacenados.



(a) Global Web - 2009 (b) Global Web - 2010 (c) Global Web - 2011

(d) Spanish Web - 2009 (e) Spanish Web - 2010 (f) Spanish Web - 2011

Figura III.17: Nivel de compresión de los contenidos en cada uno de los períodos

- Edad: Representa el tiempo de validez de una página web. Para su calculo se ha usado la cabecera HTTP "Last-Modified", que permite conocer cuándo ha sido la última vez que una página web ha sido modificada, y por lo tanto conocer su edad.

  En la Figura III.18 (a, b, c) se muestran los resultados para la Web Global. Se puede observar que en 2009 aproximadamente el 25% de las páginas tienen una edad menor a 3 meses. En el 2010 y 2011 la edad de las páginas ha seguido disminuyendo. En 2011 el 45% de las páginas tiene una edad menor a un mes. En resumen, se observa una tendencia decreciente en la edad de las páginas, lo que implica que las páginas cambian y se actualizan cada vez con mayor frecuencia.

Los resultados obtenidos sobre la Web Española se muestran en la Figura III.18 (d, e, f). En 2009 aproximadamente el 20% de las páginas tenían menos de 3 meses, y un 13% tiene entre 12 y 15 meses. En 2010 y 2011 el número de páginas con menos de 3 meses de antigüedad aumentó al 35% y al 37%, respectivamente. Por otro lado, ha disminuido el número de páginas con edad superior a 6 meses tanto en 2010 como en 2011.

Se puede concluir que las páginas de la Web Global se actualizan con más frecuencia que las de la Web Española, es decir la edad media de las páginas de la Web Global es menor. En 2011, en la Web Española, el 37% de las páginas tenían una edad menor a 3 meses frente al 45% de la Web Global. A pesar de esta diferencia vemos que la edad de las páginas ha disminuido desde 2009 a 2011 y que probablemente esta tendencia continue en los años sucesivos. Esta evolución demuestra que el contenido de las páginas web se actualiza cada vez con mayor frecuencia. Esto provoca que las políticas de re-crawling de los crawlers y las de actualización de los índices de los buscadores, tengan que cambiar para adaptarse lo mejor posible a las frecuencias de modificación de los sitios web, ya que tener repositorios e indices actualizados es esencial para que el usuario pueda realizar búsquedas de calidad. Por ello, es necesario crear un sistema que permita a los crawlers conocer cuándo una página ha cambiado, mejorando el rendimiento de los buscadores y la experiencia del usuario.

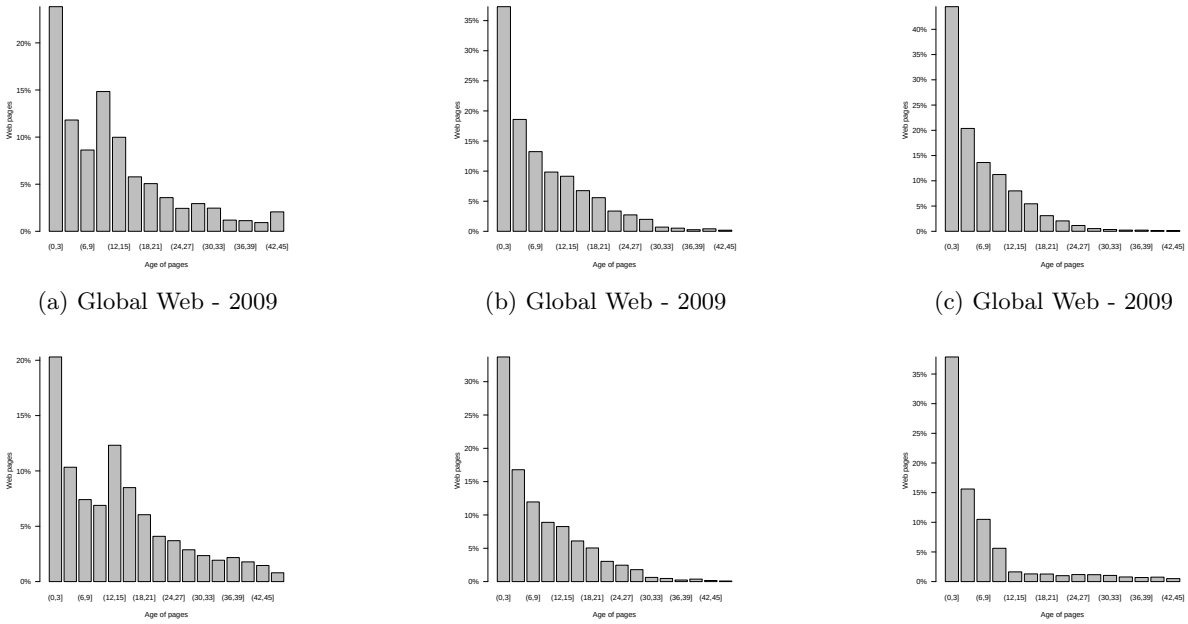
- Similitud: Indica el nivel de semejanza que tiene el contenido de dos páginas web. Concretamente, el elemento que se ha comparado ha sido el contenido útil de cada página web, usando la misma aproximación que la descrita en la sección III.2.3.

  Para la obtención de estos resultados se ha usado una herramienta implementada por Viliam Holub[21]. Esta herramienta divide cada documento en n tokens, asignándoles un peso. Después de esto, le asigna un hash a cada uno de ellos. Finalmente con el peso y el hash de cada token crea un hash del documento, que "resume" su contenido.

  Para este experimento se ha creado un dataset por año, y se ha dividido en 10 subconjuntos aleatorios con 10000 páginas web cada uno de ellos. Tras esto se ha generado el resumen (hash) de cada página web para cada uno de los años, y se ha calculado la distancia Hamming entre los hashes de las diferentes páginas web. El resultado final, mostrado en la Figura III.19, se obtuvo la media de los resultados de cada uno de los 10 subconjuntos para cada año.

  En la Figura III.19 (a, b, c) se muestran los resultados obtenidos sobre la Web Global. En 2009 el 24% de la Web Global tenía entre un 50% y un 60% de similitud, más de un 30% tenía una similitud comprendida entre un 60% y un 70% y un 24% de la web Global tenía una similitud entre 70% y 80%. En 2010 y 2011 disminuyó el número de páginas con similitud entre el 50% y el 60%, pero aumentó el número de páginas con similitud del 60%, 70% y superiores.

[21] http://d3s.mff.cuni.cz/∼holub/sw/shash/



(a) Global Web - 2009 (b) Global Web - 2009 (c) Global Web - 2009

(d) Spanish Web - 2009 (e) Spanish Web - 2010 (f) Spanish Web - 2011

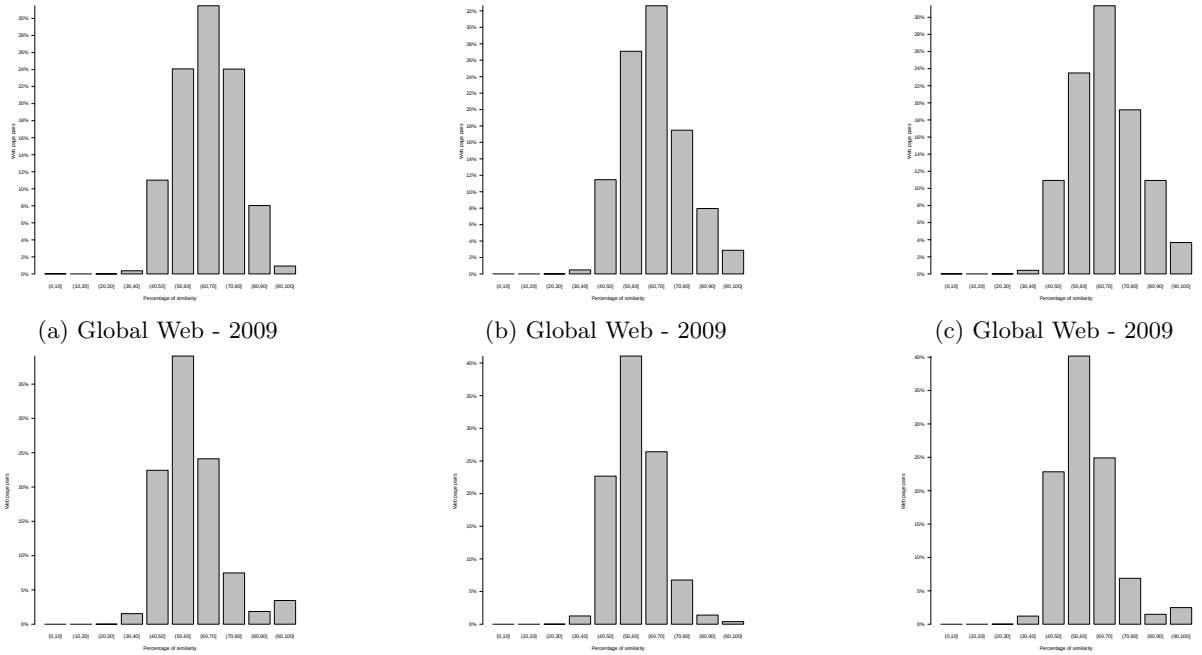
Figura III.18: Evolución de la edad de las páginas en la Web Global y Española

Los resultados sobre la similitud de la Web Española se muestran en la Figura III.19 (d, e, f). En resumen podemos decir que la similitud entre páginas se ha mantenido en los 3 años. En 2009 aproximadamente el 37% de la Web Española tenía entre un 50% y un 60% de similitud. En 2010 y 2011 este valor subió hasta el 40% de la Web Española. En los 3 años, un 22% de la Web Española tenía una similitud de entre el 60% y el 70%. Con valores superiores al 70% de similitud está el 10% de la Web Española.

Comparando los resultados obtenidos vemos que la Web Global tiene una mayor similitud que la Web Española. Centrándonos en los resultados de 2011, más del 30% de la Web Global tiene un nivel de similitud entre un 60% y un 70%, frente al 25% que obtiene la Web Española.

Estos resultados son consistentes con los obtenidos sobre el nivel de compresión de las páginas web, donde se observaba que en la Web Global era superior al de la Web Española. Por lo tanto, se puede decir que la Web Global tiene mayor cantidad de páginas con contenido similar que la Web Española.

Comparando los resultados obtenidos con los presentados por Cho *et al.* [CSGM00], vemos que la similitud ha aumentado desde el 20% o el 40%, mostrado en dicho estudio, hasta el 50% o el 60% obtenido 2011



(a) Global Web - 2009 (b) Global Web - 2009 (c) Global Web - 2009

(d) Spanish Web - 2009 (e) Spanish Web - 2010 (f) Spanish Web - 2011

Figura III.19: Similitud de las páginas en la Web Global y Española

en nuestro análisis. Este incremento es bastante significativo y es un indicador de que la calidad de los contenidos web está decayendo.

El alto grado de similitud tanto en la Web Global como en la Española, puede ser debido a:

- Los sitios web de Spam tratan de mejorar su ranking y relevancia en los motores de búsqueda creando gran cantidad de páginas web con contenido similar y con enlaces que apuntan a páginas web del mismo sitio web o de otros sitios web propios y de temática similar [PBMW98] [WD05b]. La presencia de Web Spam en la Web es un problema importante. En base a los estudios presentados por Ntoulas *et al.* [NM06], en el dominio .biz el 70% de sus páginas web son Spam, en el dominio .us el 35%, en el dominio .com el 20% y en el dominio .net el 15% de su contenido se considera Spam.
- Sitios web con contenido legítimo, pero que su contenido está repetido en otros sitios web (sitios web "espejo"). Ejemplos claros son la documentación de la API de Java, u ciertos manuales y guías de usuario que se encuentran repetidas en infinidad de sitios web.

En resumen, se puede decir que existe una pequeña parte de la Web que contiene contenidos de alta calidad y no repite su contenido, frente a una gran parte de la Web con contenidos repetidos y de mala calidad. Debido a esto, los motores de búsqueda deberían detectar aquellos sitios web que tiene multitud de contenido repetido, para tratar de penalizarlos. De esta forma se lograrían mejorar los contenidos indexados y ahorrar gran cantidad de recursos.

Características a Nivel de Sitio Web

Este nivel agrupa las características más relevantes de un sitio web, como son las tecnologías del lado cliente y servidor usadas, y el número de enlaces y sus tipos (estáticos, dinámicos, entrantes, salientes):

- Enlaces: El número de enlaces existentes entre páginas web es un indicador del nivel de interconectividad de la Web, por tanto su análisis es importante para los crawlers.

  Los resultados obtenidos sobre la Web Global indican que en el año 2009 un sitio web tenía de media 6502 enlaces. En el año 2010 y 2011 el número medio de enlaces bajó a 4639 y 5773, respectivamente. Analizando la Web Española estos valores se reducen. En el año 2009 un sitio web tenía de media 1831 enlaces. En 2010 aumentó en un 24.13%, hasta los 2273. En el último año también creció un 4.31% con respecto a 2010, hasta los 2371.

  En resumen vemos que el número medio de enlaces por sitio web es más del doble en la Web Global que en la Española. Esto indica que los sitios de la Web Global tienen mayor número de relaciones con otros sitios web que la Web Española. En el caso de la Web Española se observa un crecimiento del 29.49% desde 2009. Sin embargo, la Web Global ha bajado desde 2009 un 11.22%.

  En la Tabla III.8 se muestran los resultados sobre los enlaces entrantes, salientes, dinámicos, estáticos y el tipo de ruta absoluta o relativa. Los resultados obtenidos para la Web Global indican que los enlaces entrantes en 2009 suponían un 82.35% del total y los salientes un 17.65%, resultados que se mantuvieron similares a lo largo de 2010 y 2011.

  En base al tipo de ruta del enlace se observa que los enlaces relativos dominan frente a los absolutos. En 2009 el 60.88% de los enlaces eran relativos frente al 29.12% que eran absolutos. Este resultado se mantuvo similar en 2010 y 2011, donde el 63.22% y 36.78% eran enlaces relativos y absolutos, respectivamente.

  Analizando los resultados del número de enlaces dinámicos y estáticos vemos que los enlaces estáticos predominan sobre los dinámicos. En el año 2009 un 79.47% eran estáticos, frente un 20.53% dinámicos, porcentajes que fueron en aumento hasta llegar al 88.2% de enlaces estáticos y a 11.8% de enlaces dinámicos en el año 2011.

  Centrándonos en los resultados relativos a la Web Española se puede ver que el número de enlaces entrantes de los sitios web ha cambiado significativamente. En 2009, un 54.74% de los enlaces eran

entrantes y un 43.26% eran salientes. En 2011 este dato cambió significativamente, aumentando los enlaces entrantes hasta un 62.94% y disminuyendo los salientes hasta un 37.06%.

La relación entre enlaces absolutos y relativos se ha observado compensada y constante entre ambos tipos en aproximadamente un 50%.

Analizando los datos sobre enlaces estáticos y dinámicos, observamos que no ha variado el nivel de cada uno de los tipos en los 3 años. En 2009, el 71.97% de los enlaces eran estáticos y el 28.03% dinámicos. Esos datos cambiaron en 2011 aumentando un 5% los estáticos y disminuyendo la misma cantidad los dinámicos.

Este resultado va un poco en contra de la tendencia del uso de tecnologías que permiten acceso dinámico a los datos frente al uso de páginas estáticas. Este hecho puede ser debido a dos razones. Por un lado, a la forma en que se han detectado los enlaces dinámicos (en base a la extensión del enlace y al uso de ? y parametros en los URLs). Por otro lado, aunque las páginas dinámicas ofrecen una mejor experiencia al usuario que las estáticas, su procesamiento por parte de los crawlers es mucho más costoso y en ocasiones imposible por lo que en muchas ocasiones estos contenidos no son indexados. Esto provoca que muchos creadores de contenidos opten por sitios web estáticos.

También se ha observado que el número de enlaces entrantes es más de un 20% superior en páginas web de la Web Global que en la Web Española. Cuando un sitio web está enlazado desde multitud de sitios web indica que ese sitio web tiene cierta información relevante para todos los que lo enlazan. Por ello, se puede decir que el número de enlaces entrantes de un sitio web representa la calidad de dicho sitio web. Esta idea forma parte de muchos de los algoritmos de valoración utilizados por los buscadores [PBMW98].

Sin embargo, durante el estudio se observó que existían algunos grupos de sitios web con gran cantidad de enlaces entrantes y grupos de páginas con el mismo número de enlaces entrantes y salientes. Estos resultados muchas veces son causados por páginas de Spam, como ya ha observado Fetterly *et al.* [FMN04] y Thelwall *et al.* [TW03] en sus estudios. Este hecho ha provocado cambios en los algoritmos de ranking de páginas web.

En resumen, los motores de búsqueda deberían detectar y penalizar aquellos sitios web que traten de mejorar su relevancia (PageRank) incrementando artificialmente los enlaces de entrada de los sitios web.

- Formularios web: Como ya se explicó en la sección III.2.1, la Web Oculta del lado servidor es una parte importante de la Web. Estudiando el uso de los formularios web, se ha detectado que en la Web Global hay de media más de 1.1 formularios por sitio web. Esto indica que la mayor parte de sitios web usan formularios para dar acceso a cierta información, y que algunos de ellos tienen más de uno. En la Web Española este dato se reduce hasta los 0.4 formularios por sitio web, es decir aproximadamente uno de cada dos sitios web hace uso de formularios para el acceso a la información. En ambos casos no se han observado cambios significativos en el número de formularios usados durante estos 3 años.

Web Global						
	Entrante	Saliente	Estático	Dinámico	Relativo	Absoluto
2009	82.35%	17.65%	79.47%	20.53%	60.88%	39.12%
2010	84.68%	15.32%	85.45%	14.55%	60.99%	39.01%
2011	84.57%	15.43%	88.20%	11.08%	63.22%	36.78%
Web Española						
	Entrante	Saliente	Estático	Dinámico	Relativo	Absoluto
2009	56.74%	43.26%	71.97%	28.03%	46.98%	53.02%
2010	60.70%	39.30%	77.70%	22.30%	46.65%	53.55%
2011	62.94%	37.06%	76.97%	22.03%	49.14%	50.86%

Tabla III.8: Enlaces entrantes, salientes; estáticos y dinámicos; relativos y absolutos

Estos resultados indican que los crawlers deben estar preparados para el acceso a este tipo de información, ya sea mediante la ejecución automática de consultas en base a aprendizaje máquina, o bien estableciendo algún tipo de acuerdo con los creadores de la información, que les permita tener un acceso más sencillo a los datos.

- Tecnologías del lado servidor: Desde el punto de vista de un crawler es esencial conocer las tecnologías web usadas en el lado servidor, ya que de esta manera sabrá cual es el método más adecuado para su procesamiento (extracción de datos y enlaces).

  Los resultados de la Web Global se muestran en la Figura III.20 (a, b, c). En ellos se puede observar que en 2009 el uso de tecnologías del lado servidor estaba muy repartido, siendo dominante PHP con un 25.17%, seguido de JSP con 22% , CGI con 16.18%, SHTML con un 17.66% y ASP con 15.66%. En 2010 se produjo un cambio muy significativo. Por un lado, el uso de ASP aumentó hasta un 55.35%, y por otro el uso de PHP, pero sobre todo de CGI y PERL, se redujeron en gran medida. Esta tendencia se confirma en 2011 ya que ASP sigue aproximadamente con un 50% de uso, seguido de PHP con 30% y de JSP Y SHTML.

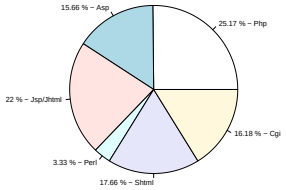
  Centrándonos en los resultados a nivel de la Web Española (Figura III.20 (d, e, f)), en 2009, 2010 y 2011 la tecnología más usada era PHP con aproximadamente un 70% de los sitios web, seguida de ASP con más de un 20%, y muy por debajo, con aproximadamente un 2%, JSP. En estos 3 años PHP se ha mantenido, ASP ha aumentado un 2% y JSP ha disminuído un 1%. Tecnologías como CGI o SHTML han disminuido su presencia desde 2009.

  En los resultados se aprecia claramente la diferencia de uso de este tipo de tecnologías entre España y la Web Global. En España se usa principalmente PHP, en cambio en la Web Global es ASP la tecnología

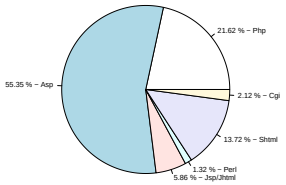
más usada seguida de PHP.

Las diferencias obtenidas en estos resultados también han aparecido en la Web de otros países. En la Web Brasileña el 73% de los sitios web usan PHP [MPZ+05] y en la Chilena el 78% [BYC04]. Sin embargo, como ocurre en la Web Global, en otros países ASP también es la tecnología más usada. Un ejemplo de esto es la Web Africana [BCSV02] en donde un 63% de los sitios Web usan ASP.

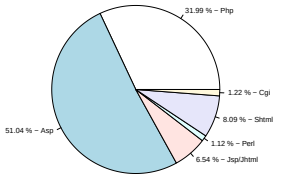
La explicación del aumento de uso de PHP, puede ser debido a que ASP es una tecnología cerrada y propietaria que únicamente funciona en un sistema operativo en particular, y PHP tiene disponible gran cantidad de interpretes en código abierto. La posibilidad en PHP de crear y modificar clases y métodos adaptados, junto con la diferencia de coste de ambas posibilidades pueden ser las razones de este continuo aumento.



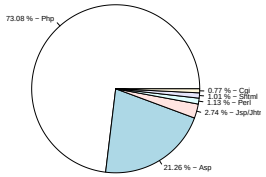
(a) Global Web - 2009



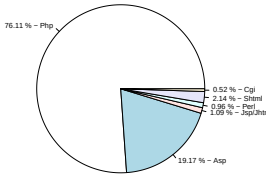
(b) Global Web - 2010



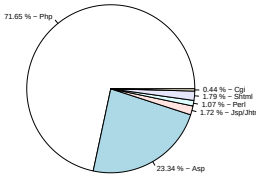
(c) Global Web - 2011



(d) Spanish Web - 2009



(e) Spanish Web - 2010



(f) Spanish Web - 2011

Figura III.20: Distribución de las tecnologías del lado servidor usadas en la Web Global y Española

- Tecnologías del lado cliente: Son aquellas tecnologías que permiten crear dinamismo en los sitios web y así mejorar la experiencia al usuario. Sin embargo, esto hace más complejo el proceso de crawling, creando lo que se conoce como Web Oculta del lado cliente.

En la Figura III.21 mostramos los resultados. A nivel global y nacional, se puede observar que las tecnologías más usadas están basadas en JavaScript. En la Web Global su uso ha aumentado desde 2009 con un 93.63% hasta un 97.15% en 2011. La siguiente tecnología más usada a nivel global es Flash, aunque ha reducido su presencia desde 2009 con un 6.28% hasta un 2.84%.

En la Web Española el lenguaje predominante también es JavaScript, apareciendo en 2009 en un 70.67% de los sitios que usaban tecnologías web, seguidos por Flash con un 28.91%. De igual forma que ocurre a nivel Global, el uso de JavaScript aumentó en 2011 hasta un 77.01% y el uso de Flash se redujo hasta el 22.17%.

También se puede observar que desde 2009 casi han desaparecido ciertos lenguajes como VbScript o Tcl. Estos resultados, son en gran parte, debido al uso extendido de tecnologías como AJAX basadas en JavaScript, y a la multitud de problemas de compatibilidad y seguridad que está teniendo Flash.

En base a los resultados, y al coste de procesamiento de estas tecnologías, se puede concluir, que los sistemas de crawling deberían centrarse en procesar JavaScript, ya que es la tecnología más extendida y esto le permitirá acceder a mayor cantidad de datos y nuevas páginas web.

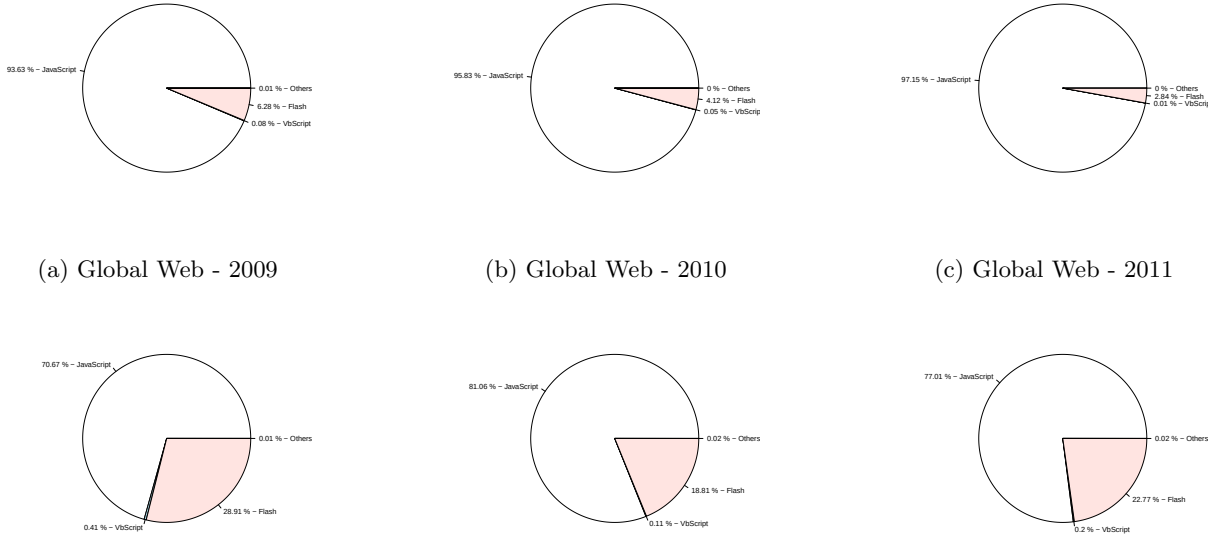
Características de la Web Nacional

En esta sección se estudiaran las características de la Web a nivel nacional. Se analizarán por un lado el software usado por los servidores web y por otro el número de dominios nuevos y eliminados a lo largo de los 3 años considerados.

- Servidor Web: Los resultados obtenidos indican que, el software más usado como servidor web es Apache, que en 2009 estaba presente en el 65.23% de los servidores y que en 2011 aumentó hasta el 70.13%. Tras el servidor Apache se encuentra el servidor de Microsoft IIS, que actualmente tiene una cuota de mercado del 26.94%. Otros servidores que están presentes en la Web española, pero de forma simbólica, son Zeus, Nginx o Lotus.

- Dominios nuevos y eliminados: El número de dominios de nueva creación o de dominios eliminados puede verse como un indicador del crecimiento de la Web de un país. El año en que empezó el estudio, la Web Española estaba formada por 1207832 dominios. En el siguiente año se produjeron un total de 76277 altas y 36131 bajas, lo cual implica 40146 nuevos dominios, es decir un aumento de un 3.3%. En el 2011 este crecimiento fue mayor, creándose 210393 nuevos dominios, lo cual implica un crecimiento de un 16.6% sobre el año anterior.

  Para estudiar el crecimiento de la Web Española, se ha considerado que cada dominio contiene una media de 400 páginas web por dominio [BYCE07]. En base a ese dato, se puede decir que el crecimiento de la



(a) Global Web - 2009 (b) Global Web - 2010 (c) Global Web - 2011

(d) Spanish Web - 2009 (e) Spanish Web - 2010 (f) Spanish Web - 2011

Figura III.21: Distribución de las tecnologías del lado cliente usadas en la Web Global y Española

Web Española es alto. Si la tendencia continúa, los sistemas de crawling y motores de búsqueda deberán buscar soluciones que indexar y crawlear todo este nuevo contenido. Deberán adaptar sus recursos al crecimiento y necesidad de la Web Española. Una posible solución pasa por distribuir el procesamiento de la Web, creando un nodo para cada país. De esta forma los motores de búsqueda podrán adaptar los recursos utilizados por cada nodo a los recursos necesarios según la Web de cada país.

Características de la Web Global

- Servidor Web: En la Web Global el software más usado como servidor es Apache con un 65.4%, un 5% menos que en la Web Española. El siguiente servidor web más extendido es el Microsoft IIS con un 18.22%, aproximadamente, un 8% menos que en la Web Española. Otro servidor que tiene una importante presencia en la Web es Ngisx, con un 10% del mercado, servidor que en la Web Española no tenía mas de un 2%. Entre otros servidores con menos de un 1% de presencia en la Web se encuentran: Apache Tomcat, IBM Web Server, Oracle Web Server, etc.

Hemos observado que en la Web Global el uso de los diferentes servidores Web esta más repartido que en la Web Española, y que la distancia entre el uso de Apache y Microsoft IIS es superior en la Web Global que en la Española.

Tanto en el estudio de la Web Global como de la Web Española, nos hemos encontrado que muchas de las versiones utilizadas no son actuales. El uso de versiones desactualizadas es un problema potencial de seguridad, ya que pueden contener errores que permitan comprometer las páginas alojadas y por lo tanto a los usuarios que las visitan. En base a los resultados obtenidos los buscadores deberían penalizar aquellos dominios que usen servidores desactualizados y que puedan presentar un peligro al usuario final.

III.3 Conclusiones y Discusión de Resultados

En esta última sección se presentan las conclusiones y resultados obtenidos a partir de los estudios realizados. Estos estudios han permitido identificar los aspectos que debe abordar un crawler eficiente y efectivo de la Web, desde el punto de vista de los contenidos. Estos estudios han sido la base de los siguientes trabajos en los que se basa la arquitectura de crawling que se propone en la tesis. En primer lugar se explicarán los resultados obtenidos del estudio sobre la eficacia de los sistemas de crawling en el tratamiento de la Web Oculta del lado cliente, y posteriormente se describirán las conclusiones obtenidas en el estudio sobre la evolución de la Web Global y Española.

III.3.1 Conclusiones y Resultados del Análisis de la Web Oculta del Lado Cliente

En la sección III.1 se ha propuesto una escala que permite clasificar a los sistemas de crawling en base a su eficacia en el tratamiento de la Web Oculta del lado cliente. El estudio analiza las tecnologías web más importantes sobre más de 120 millones de páginas web.

Los resultados obtenidos muestran que la mayor parte de los crawlers, tanto Open-Source y comerciales como de motores de búsqueda, tratan de extraer los enlaces por medio de expresiones regulares. Este procesamiento ahorra gran cantidad de recursos y permite obtener un alto porcentaje de los enlaces, pero provoca que muchos de los enlaces presentes en las tecnologías del lado cliente no se estén procesando. Los resultados obtenidos muestran que únicamente Google y WebCopier han logrado procesar alguno de los enlaces presentes en las tecnologías del lado cliente.

El estudio realizado ha dejado patente la necesidad de mejorar los métodos de procesamiento de las tecnologías del lado cliente por parte de los crawlers.

Sin embargo, el tratamiento de este tipo de tecnologías web presenta importantes desafíos. Su procesamiento lleva asociado un importante uso de recursos y de tiempo de cómputo, por lo que debe maximizarse la calidad

de las páginas que vayan a ser procesadas. Por este motivo, y debido al hecho de que muchas páginas de Spam o Soft-404 hacen uso de estas tecnologías, es importante, como paso previo a su procesamiento, filtrar este tipo páginas, que consumirán recursos, y además, disminuirán la calidad de los resultados mostrados al usuario.

III.3.2 Conclusiones y Resultados del Análisis General sobre la Web Española y Global

En esta sección se resumirán los resultados más relevantes del estudio de la Web Global y Española, y discuten las conclusiones obtenidas, desde el punto de vista de los sistemas de crawling.

A continuación se resumen algunos de los resultados extraidos y sus posibles causas:

- Las 50 etiquetas HTML más usadas son comunes durante los 3 años estudiados, tanto en la Web Global como en la Española.
- El segundo lenguaje más usado en la Web Española es el inglés, con un 28.35% de presencia y en continuo aumento. La causa de esto es la globalización, que provoca que los contenidos esten internacionalizados, para así permitir el acceso al mayor número de usuario con independencia de su lengua.
- Las páginas de la Web Española contienen un mayor número de keywords que las páginas de la Web Global, concretamente 15 palabras por página.
- El formato de audio más utilizado en la Web Española y Global es el MP3.
- Disminución generalizada del uso del atributo “refresh” de la etiqueta HTML Meta. La causa más probable de este hecho es el aumento de ciertas tecnologías como AJAX que permiten una carga dinámica, total o parcial, de la página, sin necesidad de hacer uso de dicha etiqueta.
- El charset más utilizado en 2011, y que ha ido aumentando desde el comienzo del análisis, es UTF-8. De la misma forma que el lenguaje, la causa más probable de esta evolución es la internacionalización de los contenidos, debido a las exigencias de la globalización y el comercio internacional.
- El formato de imagen más extendido en ambas Webs es el GIF, seguido de PNG que está en continuo aumento debido al aumento del ancho de banda lo cual ha permitido una mejora en la calidad de los contenidos de las páginas web.
- El formato de vídeo más presente tanto en la Web Española como en la Global es el WMV.
- En la Web Global y en la Española el tipo de documento más frecuente es el PDF.

Los resultados obtenidos han permitido detectar ciertos aspectos y módulos de los sistemas de crawling que deben ser mejorados. En la Tabla III.9 se muestran las conclusiones y mejoras propuestas. En la primera columna se explican las conclusiones obtenidas en base a los resultados, en la segunda columna se resume la mejora que debería hacerse y en la tercera columna se muestran los módulos a los que afecta dicha mejora.

A continuación se explica que parte de cada módulo puede ser mejorada:

- Módulo de almacenamiento: mejorar el uso de recursos utilizados en el almacenamiento de los contenidos web y mejorar las políticas de caching URL.
- Filtro de contenido: mejorar la calidad de los recursos indexados, con el fin de reducir los recursos utilizados en procesar e indexar recursos innecesarios (Web Spam, páginas Soft-404, contenidos repetidos, etc.). De esta forma también se mejorará la experiencia del usuario y se le evitarán posibles riesgos.
- Módulo de actualización: mejorar el proceso de actualización de los contenidos.
- Módulo de extracción de URLs: mejorar el tratamiento de la Web Oculta.

Los resultados obtenidos durante el análisis de la Web Oculta del lado cliente indican que páginas de Spam y Soft-404 deben ser filtradas antes de su procesamiento para la extracción de enlaces. Por otro lado, el análisis de la Web Global y Española, ha aportado una serie de resultados y conclusiones que indican que una serie de módulos pueden ser mejorados. Estas necesidades de mejora han sido la base de la arquitectura del sistema crawling que se propone en la presente tesis. La arquitectura propuesta se centrará en mejorar dos de los módulos:

- El filtro de contenido, se mejorará incluyendo dos nuevos detectores, uno de Web Spam y otro de páginas Soft-404, que ayuden a los crawlers a detectar dicho tipo de páginas antes de procesarlas. De esta forma se ahorrarán recursos al no procesar y extraer enlaces de dichas páginas (tampoco serán indexadas y, por lo tanto, no aparecerían en los resultados de las búsquedas de los usuarios.
- El módulo de actualización, se modificará para que use un nuevo método que le permita conocer cuándo una página web ha sido modificada. Esto evitará la perdida de recursos revisitando páginas o sitos web que todavía no han cambiado, y mejorando los contenidos almacenados, ya que estos estarán más actualizados.

En el capítulo IV se describirá y analizará la arquitectura de crawling propuesta. Tras esto, en los capítulos V, VI y VII se explicarán cada uno de los módulos con los que se propone extender la arquitectura del crawler, así como también las novedosas técnicas y métodos que utilizan.

Conclusiones de los resultados	Posibles mejoras	Componente
El contenido de las páginas web en la Web Española es la mitad que en la Web Global	Mejorar y adaptar las políticas de almacenamiento a cada país	Módulo de almacenamiento
La Web Española crece muy rápido, concretamente entre 2010 y 2011 han aparecido un 16.6% de nuevos dominios	Mejorar y adaptar las políticas de almacenamiento a cada país	Módulo de almacenamiento
La relación entre el contenido útil y el contenido total es inferior en la Web Global que en la Española, en la cual representa el 50% del total	Mejorar y adaptar las políticas de almacenamiento a cada país	Módulo de almacenamiento
Incremento del tamaño de las URLs	Mejorar y adaptar las políticas de almacenamiento a cada país	Módulo de almacenamiento
El ratio de compresión de las páginas web en la Web Española está decreciendo	Detectar y penalizar a aquellos sitios web que realicen técnicas de Spam	Filtro de contenido
Gran cantidad de sitios web contienen contenido repetido	Detectar y penalizar a aquellos sitios web que realicen técnicas de Spam	Filtro de contenido
La Web Global y la Española utilizan versiones obsoletas de servidores y tecnologías web	Detectar y penalizar a aquellos sitios web que las sigan utilizando	Filtro de contenido
Aumento del número de sitios web con un gran número de enlaces entrantes con la finalidad de mejorar su relevancia en la Web	Detectar y penalizar a aquellos sitios web que realicen técnicas de Spam	Filtro de contenido
La edad de la páginas web está en continuo decrecimiento	Incrementar y mejorar las políticas y métodos de actualización del contenido	Módulo de actualización
El uso de formularios web para acceder a la información está aumentando	Proporcionar más recursos para el procesamiento de la Web Oculta	Módulo de extracción de URLs
Las tecnologías web más usadas en la Web Española y Global son Flash y JavaScript	Desarrollar y mejorar intérpretes de Flash y JavaScript	Módulo de extracción de URLs

Tabla III.9: Resumen de resultados y conclusiones obtenidas en el estudio de la evolución de la Web Global y Española

IV

Arquitectura de Crawling Eficiente

Como se ha demostrado en capítulos anteriores, no todos los recursos enlazados de la Web representan páginas con contenido útil para los usuarios. El Web Spam o las páginas Soft-404 son claros ejemplos. Por otra parte, ha quedado patente en los análisis realizados que los contenidos de los sitios web cambian cada vez con mayor frecuencia. Todos estos problemas dificultan la tarea de los sistemas de crawling, que deben intentar minimizar el número de recursos utilizados en páginas irrelevantes para dedicarlos a contenidos válidos, o a mantener lo más actualizados posibles los recursos indexados, pues de ellos dependerá la calidad de los resultados devueltos a los usuarios.

En esta sección se describe la arquitectura propuesta para sistemas de crawling eficiente. El sistema integra componentes para la detección de Web Spam y páginas Soft-404, así como para mejorar el proceso de actualización de los contenidos del repositorio utilizando el menor número de recursos.

Se describirán y analizarán en detalle los diferentes componentes de la arquitectura de crawling propuesta. También se realizará una explicación de alto nivel de los módulos de detección de Spam, Soft-404 y cambios en páginas web. Sin embargo, la explicación detallada de las técnicas y métodos propuestos, tanto para la detección de Web Spam y Soft-404, como para la detección de modificaciones en páginas web, se realizará en los próximos capítulos.

La estructura del capítulo es la siguiente. En la sección IV.1 se explica en profundidad la arquitectura y los componentes del sistema propuesto. Tras esto se discuten sus características y ventajas. En la sección IV.2 se describe el módulo de detección de Web Spam y su interacción con el sistema de crawling. La sección IV.3 analiza el módulo de detección de páginas Soft-404 y su funcionamiento dentro del sistema de crawling. En la sección IV.4 se discute el método propuesto para la detección de cambios en páginas web y cómo dicho módulo interacciona con el sistema de crawling para mejorar el proceso de actualización del repositorio. Por último, en la sección IV.5 se explican brevemente las conclusiones obtenidas.

IV.1 Arquitectura de Crawling propuesta

Los sistemas de crawling son aquellos programas software capaces de procesar y analizar la Web con el fin de generar un repositorio de información sobre el que permitir realizar consultas de usuarios. Un crawler recorre la Web siguiendo los diferentes URL descubiertos, en un cierto orden, analiza el contenido de las páginas web descargadas y lo procesa para obtener nuevos URL que serán tratados.

Diversas compañías de data mining y de motores de búsqueda han realizado numerosos trabajos de investigación sobre los diferentes problemas que plantea la obtención de información de la Web. Desafortunadamente, muchas de las técnicas, y en especial las que tienen que ver con temas de optimización y rendimiento, no se han hecho públicas, ya que dichas compañías las consideran de valor estratégico. Son pocos los documentos de dominio público que discuten ciertos detalles de la implementación de un crawler. Entre

ellos se pueden citar la implementación del crawler "Mercator", realizada por Compaq para Altavista [HN99] y una descripción del primer crawler utilizado por Google [BP98].

Uno de los estudios más detallados es el realizado por Chakrabarti [Cha02] donde se presenta la arquitectura básica de un sistema de crawling eficiente. En la Figura IV.1 se muestra la arquitectura de crawling propuesta que contiene el conjunto de elementos presentados por Chakrabarti. Los elementos que la forman son los siguientes:

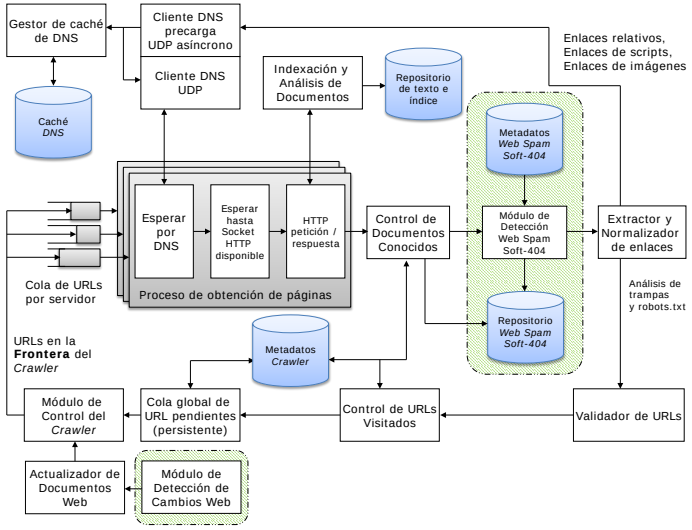


Figura IV.1: Arquitectura de un sistema de crawling eficiente de Chakrabarti (se resaltan con fondo de rejilla las extensiones propuestas en esta Tésis)

- Módulo de Crawling: está formado por un grupo de procesos de crawling que se encargan de recorrer la Web partiendo de un conjunto inicial de URLs, para crear un repositorio con sus contenidos.

- Normalizador de URLs: es el responsable de normalizar los URLs extraídos para evitar que un documento sea accedido varias veces por no haber detectado que dos URLs representan un enlace al mismo recurso.

- Exclusión de Robots: elemento encargado de aplicar el protocolo de exclusión de robots (robots.txt y sitemap.xml) para determinar qué páginas pueden ser indexadas. Para conocer qué páginas pueden ser indexadas, y cuales no, se debe procesar el fichero "robots.txt" del directorio raíz de cada dominio, o la etiqueta Meta de cada página.

- Control de URLs visitados: es el encargado de comprobar si un URL ya ha sido procesado. Los sitios web contienen multitud de enlaces repetidos, por lo que la comprobación de si el URL ya ha sido visitado debe ser muy eficiente, porque es una operación que se ejecuta con cada URL obtenido. Normalmente se realiza calculando una función hash sobre el URL, diferenciando el nombre de máquina del resto del URL.

- Módulo de control de documentos conocidos: es el componente encargado de evitar procesar varias veces aquellos contenidos que el módulo de control de URLs visitados no ha detectado. Esta situación ocurre frecuentemente ya que existen multitud de sitios web legítimos, y no legítimos, que replican el contenido de otros sitios web. Detectar que se trata de un sitio o página web ya procesado debe realizarse cuanto antes para perder el menor número de recursos en ello. La técnica más habitual consiste en realizar el hash de cada contenido procesado. Sin embargo, aquellos contenidos que tengan pequeños cambios serán considerados como diferentes cuando en realidad no es cierto.

- Validador de URLs: en la Web existen multitud de trampas que pueden provocar el mal funcionamiento del crawler y la consecuente pérdida de recursos, como código o enlaces maliciosamente añadidos. Este módulo es el responsable de evitar que el sistema sea víctima de algún código o enlace maliciosamente añadido.

- Control del crawler: se encarga de monitorizar la carga de la red, para decidir qué páginas web deben ser accedidas por los procesos del crawler. Almacena los URLs a ser accedidos en colas que constituyen lo que se conoce como la frontera del crawler.

  Este módulo realizará la selección de las páginas, priorizando los URLs en la cola de forma adecuada a la política usada. Pueden considerarse las siguientes políticas de acceso a URLs:

  - Recorrido en anchura (FIFO, First In, First Out): es la más utilizada. Consigue distribuir la carga entre los diferentes servidores y evitar trampas de crawling, como bucles infinitos.
  - Recorrido en profundidad (LIFO, Last In, First Out): este recorrido realiza todas las peticiones al mismo servidor de forma consecutiva.
  - Los mejores primero: este recorrido prioriza el siguiente URL en ser accedido en base a diferentes métricas de calidad de las páginas.
  - Aleatorio: en este tipo de recorrido la selección de URLs se realiza de forma aleatoria.

- Colas de URLs por servidor: la transferencia de una página web por la red constituye otro recurso compartido, por ello los crawlers deben minimizar su impacto sobre estos recursos.

  Actualmente multitud de servidores HTTP se protegen contra ataques de denegación de servicio. Usualmente para evitar ataques o situaciones de alta carga, los servidores limitan la velocidad o frecuencia de las respuestas a una misma dirección IP.

Un crawler debe evitar dichas situaciones, tanto por motivos de rendimiento, como por problemas legales. Para ello, se limita el número de peticiones activas contra un servidor en un momento determinado, utilizando una cola de peticiones por cada servidor, que van rotando en base a un límite de tiempo.

La tarea de un sistema de crawling, debido tanto a la cantidad como a la variabilidad y calidad de la información que tiene que recopilar, presenta numerosos desafíos: Web Spam, Web Oculta del lado cliente/servidor, repetición de contenidos, contenidos de mala calidad, etc. Todos estos desafíos se pueden resumir en uno, procesar el mayor y mejor número de documentos web con el menor número de recursos y en el menor tiempo posible.

Para lograr esto se debe mejorar la eficiencia de los sistemas de crawling. Existen estudios que se centran en mejorar la eficiencia de los sistemas de crawling, como el realizado por Akamine *et al.* [AKK+09] o el artículo de Kimball *et al.* [KMSB08].

La presente tesis aborda la mejora de la eficiencia de los sistemas de crawling realizando un uso eficiente de sus recursos. Para ello, la arquitectura presentada evita el procesamiento de recursos irrelevantes e intenta refrescar contenidos en el momento más adecuado para minimizar los refrescos innecesarios a la vez que su nivel de obsolescencia. Los demás elementos de la arquitectura de crawling propuesta se basan en la arquitectura presentada en [Cha02], que han sido descritos previamente.

En la Figura IV.1 donde se mostró la arquitectura de crawling propuesta por Chakrabarti, se han añadido los siguiente 3 módulos (marcados en fondo de rejilla):

- Módulo de Detección de Web Spam: es el módulo encargado de analizar las páginas o sitios web descargados y detectar si son Spam. El contenido a analizar se obtiene a partir del módulo de control de documentos, de este modo si una página web se detecta como repetida no se volverá a analizar, con el consiguiente ahorro de recursos. Además, este módulo se comunica con dos repositorios, uno de metadatos, donde estarán almacenados los datos y políticas necesarias para detectar Spam, y otro donde se almacenan los sitios y páginas web detectadas como Spam. El funcionamiento y detalles del módulo se explican en la sección IV.2 y en el capítulo V.

- Módulo de Detección de Páginas Soft-404: es el responsable de detectar si una página web es Soft-404. De la misma forma que el módulo de detección de Spam, recibe los contenidos del módulo encargado de detectar documentos repetidos. Se analizará en profundidad en la sección IV.3 y en el capítulo VI.

- Módulo de Detección de Cambios en Páginas Web: es el módulo encargado de mejorar el proceso de refresco de los contenidos. Este módulo, en base a un novedoso sistema de detección de cambios web, notificará al módulo de actualización de documentos aquellas páginas que han sido modificadas. Tras esto, el módulo de actualización de documentos conoce en tiempo real qué sitios y páginas web han sido modificadas y podrá priorizar el procesamiento de nuevas páginas. De esta forma, el módulo logra

procesar aquellas páginas modificadas en el menor tiempo posible, no pierde cambios y además nunca consumirá recursos procesando páginas que no han sido cambiadas. El método y componentes utilizados por este módulo se explicarán en la sección IV.4 y en el capítulo VII.

IV.2 Módulo de Detección de Web Spam

El módulo de detección de Web Spam tiene como objetivo evitar que el sistema de crawling procese e indexe páginas web de Spam. Para ello se ha realizado un estudio sobre páginas de Spam y se ha identificado un conjunto de heurísticas que permiten detectar dicho tipo de páginas. Para la combinación de estas heurísticas se han utilizado algoritmos de árboles de decisión.

A diferencia de otras aproximaciones existentes en la literatura, este módulo está diseñado para detectar todo tipo de Web Spam: Cloaking [WD05a], Link Farm [WD05b] [GGM05], Redirection Spam [CM07] y Content Spam [FMN05] [GJC09]. Las heurísticas en las que se basa serán explicadas en detalle en el capítulo V. Entre ellas se pueden destacar las siguientes: a) frases específicas, b) funciones decode/encode, c) inyección de HTML, d) numero de KeyWords/Description Words, e) longitud del código evaluado, f) ortografía, etc.

Como ya se ha comentado, el módulo utiliza árboles de decisión para la combinación de las heurísticas, concretamente el algoritmo C4.5 [Qui96a], junto con técnicas de "bagging" [Qui96a] y "boosting" [Qui96a] para lograr mejorar los resultados obtenidos. Los árboles de decisión han sido creados mediante el entrenamiento del módulo con conjuntos de páginas de Spam.

En la Figura IV.2 se muestran los diferentes componentes que forman el módulo. A continuación se describe y analiza cada uno de ellos:

- Analizador de contenido: es el elemento encargado de aplicar cada una de las heurísticas sobre las diferentes páginas web.
- Configurador del sistema: es el responsable de seleccionar el clasificador (árbol de decisión) adecuado, de acuerdo a los requerimientos de detección del crawler y de los recursos disponibles en el mismo.
- Clasificador: es el componente que contiene y ejecuta los diferentes árboles de decisión, para clasificar y decidir cuándo una página web es considerada como Spam.
- Repositorio: elemento donde se almacenarán los datos de las páginas y dominios clasificados como Spam.
- Repositorio de metadatos: es un elemento de almacenamiento donde se guardarán los resultados y umbrales obtenidos de ejecutar las heurísticas sobre las diferentes páginas web.

- Analizador supervisado de resultados: es el responsable de, en base a la etiquetación manual de páginas incorrectamente clasificadas, retroalimentar el clasificador para mejorar los umbrales del árbol de decisión y con ello mejorar la detección de Spam.

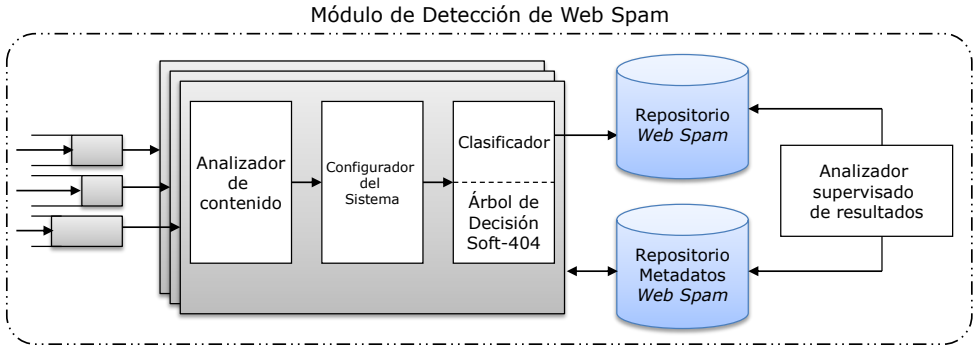


Figura IV.2: Componentes del módulo de detección de Web Spam

Partiendo de los componentes previamente descritos, se explica a continuación el funcionamiento del módulo:

1. El analizador de contenido aplica las diferentes heurísticas y extrae los resultados.
2. El configurador del sistema selecciona el árbol de decisión que se aplicará, dependiendo de los recursos disponibles y del nivel de detección configurado previamente.
3. El clasificador, en base a los resultados obtenidos para las diferentes heurísticas, recorre el correspondiente árbol de decisión para determinar si la página analizada es Spam o no.
4. En caso de ser considerada Spam se detiene el procesamiento de dicha página, y el dominio se almacenará en el repositorio.
5. Finalmente, debido a que las técnicas de Web Spam están en continuo cambio, el Analizador supervisado de resultados recalculará nuevos árboles de decisión en base a la información almacenada en el repositorio de metadatos y a la etiquetación manuel de páginas de Spam. Este proceso permitirá adaptarse a nuevos umbrales y técnicas de Spam, y por lo tanto mejorar los resultados.

El módulo de detección de Web Spam descrito permite a un crawler mejorar su rendimiento y el uso de recursos, ya que no procesará, indexará ni mostrará los resultados de páginas de Web Spam. A pesar de que el uso del módulo de detección provoca un consumo de recursos, será menor que el que ocasionaría el crawling

de recursos Spam. También es importante destacar que otro objetivo que se consigue es no mostrar nunca resultados de Web Spam al usuario final.

Además, dado que el uso de recursos en los crawlers es crítico, se han diseñado una serie de políticas que seleccionan el conjunto de heurísticas a aplicar anteponiendo el procesamiento normal del crawler a la detección de Web Spam. El módulo de detección de Web Spam contiene 3 árboles de decisión diferentes en función del nivel de detección necesario y de los recursos disponibles (Alta Detección-Bajo Rendimiento, Media Detección-Medio Rendimiento y Baja Detección-Alto Rendimiento). Estos árboles de decisión considerarán más o menos heurísticas, según el nivel de detección y rendimiento deseados.

IV.3 Módulo de Detección de Páginas Soft-404

Este módulo es similar al de detección de Web Spam, pero orientado a detectar páginas Soft-404. Tras realizar un análisis sobre las características de las páginas Soft-404 se han identificado una serie de heurísticas que permiten detectar este tipo de páginas.

El conjunto de heurísticas creadas se analizarán en el capítulo VI. Las heurísticas y métodos propuestos mejoran en eficacia y eficiencia los métodos presentes en la literatura [BYBKT04] [LKK+09]. El módulo propuesto se basa, entre otras, en heurísticas tales como: a) ratio de contenido útil y contenido total, b) número de imágenes, c) longitud de las palabras, etc.

Del mismo modo que el módulo de detección de Web Spam, el presente módulo utiliza arboles de decisión para clasificar páginas como Soft-404. El módulo usa el algoritmo C4.5 [Qui96a] junto con "bagging" [Qui96a] y "boosting" [Qui96a] para mejorar los resultados. Estos árboles de decisión han sido creados mediante el entrenamiento del módulo con conjuntos de páginas de Soft-404.

Aunque como veremos en el capítulo VI, los resultados obtenidos por el módulo son satisfactorios, existe un pequeño número de páginas Soft-404 que no son detectadas. Normalmente se trata de casos en los que el servidor devuelve la página raíz ante la petición de una página desconocida. Sin embargo, esto no será un problema para el sistema de crawling ya que el módulo de detección de documentos similares, que evita qué documentos idénticos o similares se procesen e indexen varias veces, detectará que el contenido de la página raíz ya ha sido indexado y no lo procesará nuevamente. Tras esto el crawler eliminará este contenido repetido.

Este módulo, de forma similar al de detección de Web Spam, también contiene 3 árboles de decisión diferentes, que se usarán de acuerdo al nivel de detección configurado previamente y a los recursos del sistema de crawling.

Los componentes y el funcionamiento de este módulo es similar al comentado previamente para Web Spam, por lo que no se entrará a describir cada uno de ellos. En el capítulo VI se discutirán las heurísticas propuestas, el método utilizado para su combinación y se comparán los resultados obtenidos con los presentes en el estado

del arte.

IV.4 Módulo de Detección de Cambios en Páginas Web

El objetivo de este módulo es permitir al sistema de crawling conocer las modificaciones realizadas por los usuarios en sus páginas en "tiempo real". Este módulo está basado en el sistema Web Change Detection (WCD). En el momento en que una página web utiliza este sistema, dicha página será monitorizada y el sistema detectará todos los cambios producidos en ella.

El sistema WCD está basado en la arquitectura distribuida de la World Wide Web e incorpora un servidor WCD y un agente WCD para crear una arquitectura colaborativa y así poder detectar cambios en páginas web casi en tiempo real. Los principales componentes de la arquitectura del sistema WCD son los siguientes:

- Cliente Web: es un navegador web que es responsable de cargar la página web, enviar una petición al servidor WCD para recibir el correspondiente agente WCD y ejecutarlo.
- Servidor Web: es un servidor web común que almacena y envía una o más páginas web monitorizadas.
- Servidor WCD: es un servidor multithread encargado de realizar principalmente dos tareas. Por un lado almacenar y enviar los diferentes agentes WCD. Y por otro, debe detectar modificaciones en las páginas web monitorizadas, y almacenar y actualizar dicha información.
- Agente WCD: es una aplicación ejecutada en el navegador web del usuario que se encarga de enviar información al servidor WCD sobre las modificaciones realizadas en las páginas web. En concreto el agente WCD es una aplicación JavaScript que se descarga del servidor WCD para ser ejecutada. Esta aplicación creará un "resumen" del contenido de la página web visualizada. Esto es lo que permite al servidor WCD detectar cambios en las páginas web monitorizadas.

  Inicialmente se han considerado sólo dos tipos diferentes de agentes, aunque el sistema está diseñado para gestionar cualquier número de agentes. Los dos agentes considerados son los siguientes:

  - Agente Digester: es el agente encargado de procesar la página web y realizar el resumen de la misma. A grandes rasgos este resumen se hará obteniendo el contenido útil de las diferentes partes de la página web y generando un hash de las mismas.
  - Agente Void: no realiza ningún procesamiento ni resumen en el lado cliente, y por lo tanto no enviará ninguna notificación al servidor WCD. Este agente se utiliza cuando el servidor WCD considera que la información sobre los cambios realizados en la página es suficientemente actual.

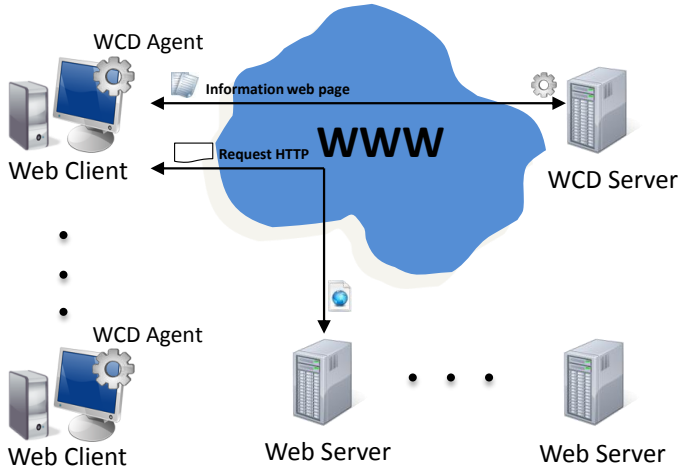


Figura IV.3: Arquitectura de alto nivel del sistema WCD

En la Figura IV.3 se muestra la arquitectura de alto nivel del sistema WCD.

El sistema ha sido diseñado para ser altamente escalable debido a su arquitectura distribuida y a que gran parte del procesamiento lo realizan los navegadores web de los usuarios. Además, se han creado diversos protocolos que ayudan a mantener estable y seguro el sistema y la información que este contiene. Las situaciones que han sido consideradas son las siguientes:

- Situaciones de alta carga.
- Ataques DDoS (Distributed Denial of Service).
- Modificación de datos enviados al sistema.

Todos los componentes mostrados en esta sección, la arquitectura que forman, el funcionamiento del sistema y diferentes cuestiones relativas a la implementación se explicarán en más detalle en el capítulo VII.

IV.5 Conclusiones

Este capítulo describe la arquitectura de crawling eficiente propuesta. Parte de la arquitectura de crawling eficiente de Chakrabarti y la extiende con nuevos componentes que permiten liberar recursos de crawling que podrán ser utilizados en el procesamiento de un mayor número de páginas.

En particular, las ventajas que aporta la arquitectura de crawling propuesta son las siguientes:

- Mejora de los recursos indexados: un mayor número y de mejor calidad. Los métodos propuestos para la detección temprana de páginas de Spam y Soft-404, permiten evitar el procesamiento de páginas y sitios web irrelevantes. Por ello, se consigue mejorar la calidad de los contenidos indexados y no malgastar recursos indexando y procesando documentos web inadecuados.

- Mejora del proceso de actualización de los contenidos: contenidos más actualizados y con coste de matenimiento mínimo. El método de detección de cambios en páginas web propuesto permite mantener más actualizados los contenidos indexados a la vez que evita visitar páginas y sitios web cuando estos todavía no han sido modificados. De esta forma se mejora la calidad de los contenidos y la utilización de los recursos.

En los capítulos V, VI y VII, se describen cada uno de los nuevos módulos propuestos y se comparan los resultados obtenidos con las aproximaciones existentes en el estado del arte.

V

Módulo de Detección de Web Spam

Este capítulo describe detalladamente el módulo de detección de Web Spam que contiene: un conjunto de técnicas innovadoras para detectar Web Spam y un método para combinarlas adecuadamente y así mejorar los resultados obtenidos.

En la sección V.1 se explican las técnicas de detección existentes. En la sección V.2 se discuten y evalúan cada una de las nuevas heurísticas que se proponen para la detección de Web Spam. Analizando cada una de ellas de forma independiente y comprobando la probabilidad de Spam asociada a cada una de ellas. La sección V.3 explica el método creado, a partir de la combinación de diferentes heurísticas, para la detección de Web Spam. Tras esto en la sección V.4 se describen los datasets utilizados para la realización de los experimentos, y los resultados obtenidos con las técnicas propuestas. También se comparan los resultados obtenidos a nivel de eficacia y eficiencia con los presentes en la literatura. Por último, en la sección V.5 se resumen los resultados obtenidos, comparándolos con los de sistemas existentes, y se analizan las conclusiones obtenidas.

V.1 Técnicas Existentes de Detección de Web Spam

El conjunto de técnicas y métodos presentes en el estado del arte ha sido analizados en la sección II.3.1. Entre los estudios presentes en la literatura, destacan los propuestos por Ntoulas *et al.* [NM06] y Gonzalez *et al.* [GJC09]. Se basan en una serie de heurísticas para la caracterización de Web Spam. A continuación se resumen las heurísticas usadas en ambos estudios:

- Palabras en una página: los autores proponen analizar el número total de palabras de la página web, excluyendo etiquetas HTML. Esto es debido a que algunas páginas web hacen Spam incluyendo en su contenido palabras populares de queries ampliamente usadas por usuarios web.
- Palabras en el título: idea similar a la anterior, pero centrándose en las palabras contenidas en el título de la página web. En sus análisis comprueban que un mayor número de palabras en el título, aumenta la probabilidad de ser Web Spam.
- Longitud de las palabras: analiza la longitud media de las palabras de una página web.
- Palabras del anchor: que estudia el número total de palabras contenidas en los "anchor" de cada página web. La idea de esta técnica de detección está en el hecho de que los spammers usan "anchor" para describir, con palabras populares y de alta relavancia, enlaces que apuntaban a páginas con un contenido diferente, páginas de Spam.
- Relación del contenido visible: que analiza la relación entre el contenido total de una página web y el contenido sin etiquetas HTML. De esta forma, los autores pretenden detectar las desviaciones producidas en las páginas de Spam, al tratar de ocultar palabras clave en ciertas zonas del código HTML.

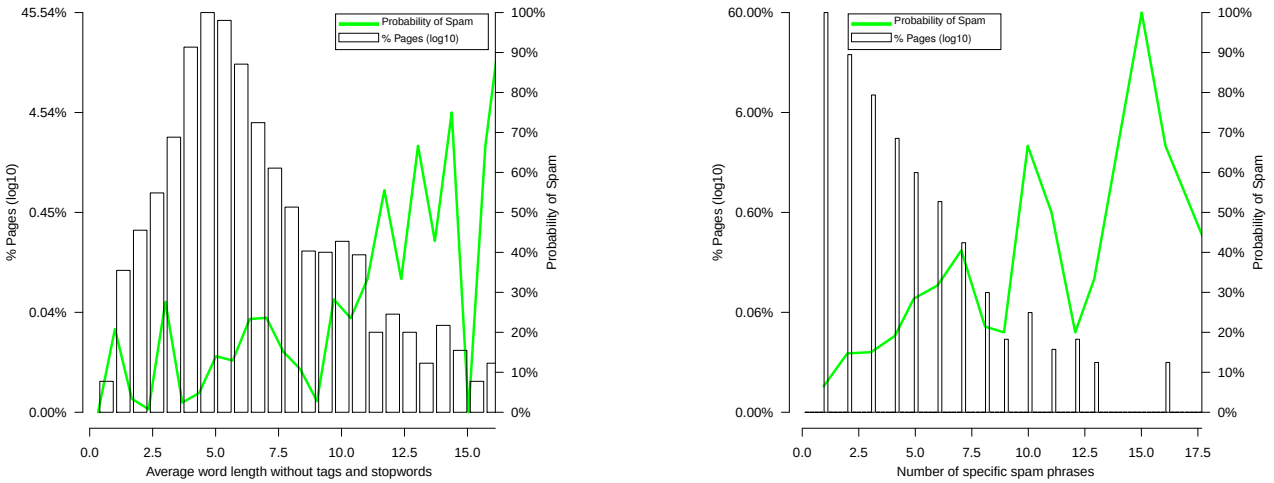
- Ratio de comprensión: técnica que estudia la relación entre el contenido comprimido y el contenido sin comprimir. Esta técnica se basa en la idea de que el contenido de las páginas de Spam es redundante.
- Palabras populares: se estudia la relación entre las N palabras más comunes y el número total de palabras populares consideradas.
- Fracción de palabras populares: técnica similar a la anterior, pero que analiza únicamente las apariciones diferentes de palabras populares y el número total de palabras populares consideradas.
- Probabilidad n-gram independiente: técnica de análisis estadístico del contenido, basada en dividir el contenido de cada página en n-grams de n palabras consecutivas.
- Probabilidad n-gram dependiente: técnica similar a la anterior, pero que considera los n-grams dependientes.

V.2 Técnicas Propuestas de Detección de Web Spam

Las técnicas explicadas en la sección anterior sólo permiten detectar aquellas páginas de Web Spam que usen técnicas de Spam Content. Con el fin de poder abordar todos los tipos de Spam comentados anteriormente (Cloacking, Redirection Spam, Content Spam y Link Spam), se ha realizado un análisis, a partir del cual se han identificado nuevas heurísticas. El conjunto de heurísticas que se proponen es mayor y trata de detectar todos los tipos de Web Spam, por ello su combinación ofrecerá una detección más robusta y segura.

A continuación se comentan de forma detallada las principales heurísticas en las que se basa el módulo de detección. Para justificar la eficacia de cada una de ellas, se han aplicado sobre el dataset descrito en la sección V.4, mostrando su probabilidad de Spam en función de los diferentes valores posibles que se han obtenido para cada una de ellas, y la fracción de páginas a las que afecta cada valor. Para mejorar la representación de los datos del eje que indica la fracción de páginas afectadas por cada vector de cada heurística, hemos usado una progresión log10, pero mostrando el porcentaje y no el valor absoluto.

- Longitud de las palabras II: partiendo de la idea mostrada en la sección anterior, que trataba de detectar Web Spam basándose en la longitud de las palabras, se propone una mejora a este método. Tomar la longitud media del contenido, sin tener en cuenta las etiquetas HTML y las stopwords. Esto es debido a que las etiquetas HTML no son contenido que se muestre al usuario e introducirían ruido en los resultados. También se ha observado que las páginas no-Spam usan una mayor cantidad de stopwords. Esto ocurre porque una página legítima usa preposiciones, conjunciones o artículos en una medida normal, pero las páginas de Spam se centran principalmente en introducir keywords que mejoren su posicionamiento en las búsquedas.



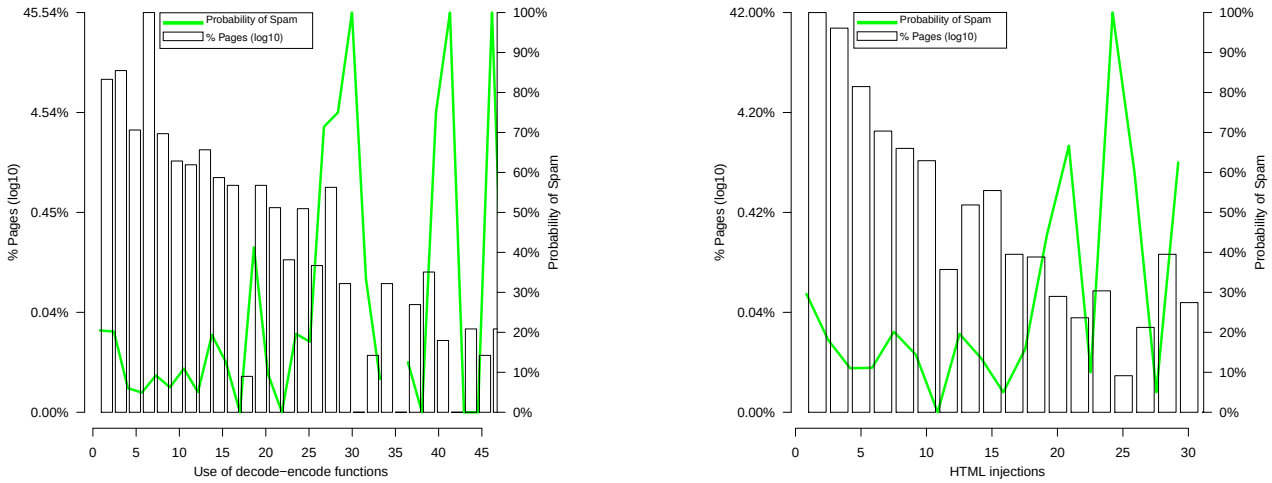
(a) (b)

Figura V.1: Probabilidad de Web Spam relativa a la longitud media de las palabras de una página sin etiquetas HTML y stopwords (a), y probabilidad de Spam relativa a las frases específicas usadas en una página web (b)

En la FiguraV.1a se observa que gran parte de las páginas contienen palabras con una longitud entre 2,5 y 7,5 caracteres. De igual forma podemos decir que una página con una longitud media de palabras menor de 8 tiene una probabilidad por debajo del 25% de ser Spam, en cambio con valores superiores podemos ver que dicha probabilidad va aumentando progresivamente hasta llegar casi al 90%. Comparando los resultados con los obtenidos por Ntoulas *et al.* [NM06], se ve que los resultados obtenidos son mas concluyentes, ya que a partir de 8 la progresión es creciente y con mejores probabilidades.

- Frases específicas: es habitual que las páginas de Spam contengan frases usuales de lenguajes o de queries de buscadores. Esto también se ha observado en otros ámbitos como el Email Spam donde suelen contener frases, o palabras específicas de Spam. Tras analizar nuestro dataset hemos creado un listado de dichos términos. Entre otros contiene palabras como: ”drugs“, ”viagra“, ”urgent“, ”discount“, etc.

  En la FiguraV.1b se muestran los resultados obtenidos utilizando el listado comentado. Podemos ver que a partir de 7 palabras típicas de Spam la probabilidad de Spam va aumentando progresivamente entre un 40% y un 100%. Esta técnica será usada tanto en el contenido como en los atributos keywords y description words de la etiqueta Meta.



(a) (b)

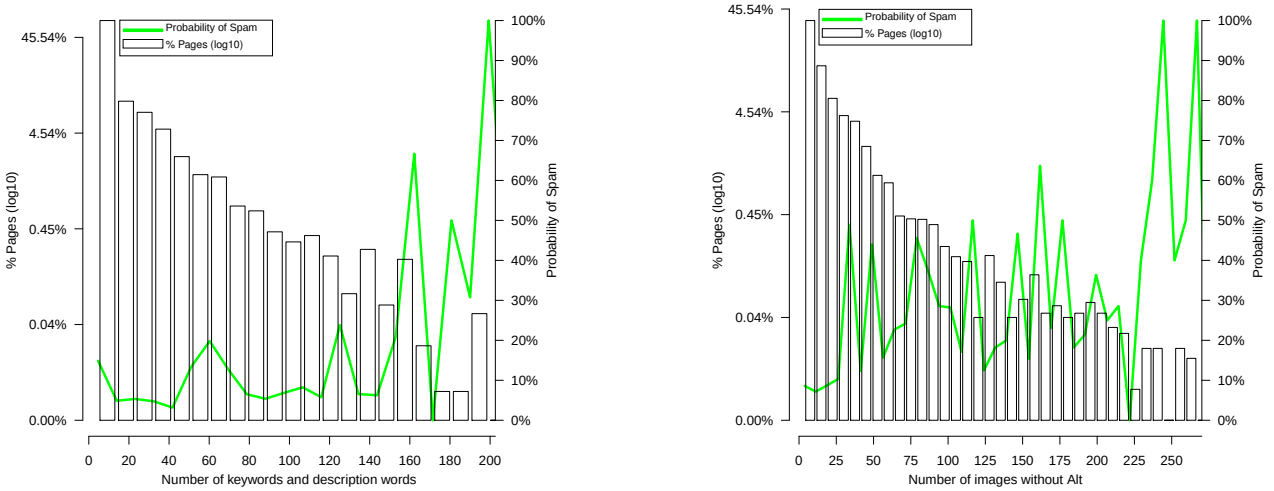
Figura V.2: Probabilidad de Web Spam relativa a las funciones decode/encode de una página web, y probabilidad de Spam relativa a las inyecciones de HTML realizadas en una página web

- Funciones de codificación: en Web Spam muchas veces se trata de ocultar redirecciones a páginas, funciones o cierto contenido, codificándolas. Por ello, es habitual el uso de las funciones JavaScript escape/unescape. Como técnica mas avanzada se ha observado el uso combinado de dichas funciones, unas dentro de las otras, con la idea de dificultar aún más su detección.

  En la FiguraV.2a se observa que entre 0 y 17 funciones, la probabilidad de Spam es relativamente baja por lo que lo más probable es que sea no-Spam. En cambio entre 17 y 22 y a partir de 25 la probabilidad aumenta progresivamente llegando a alcanzar el 100% de probabilidad de Spam.

- Inyección de HTML: a pesar de que gran cantidad de HTML en las páginas se genera en base a acciones del usuario, se ha observado que es superior en páginas con Spam. Por ello, se propone analizar los scripts en busca de funciones JavaScript que generen código HTML, tales como: “innertext”, “outerhtml”, “createElement”, “appendChild”, etc.

  Los resultados se muestran en la FiguraV.2b dónde se observa que aquellas páginas con menos de 15 inyecciones de HTML pueden ser consideradas como no-Spam. En cambio, para valores superiores a 15 la probabilidades de Spam suben a 60%, 70% e incluso al 100%.



(a) (b)

Figura V.3: Probabilidad de Web Spam relativa al número de keywords y description words de una página web (a), y probabilidad de Spam relativa al número de imágenes sin el atributo Alt (b)

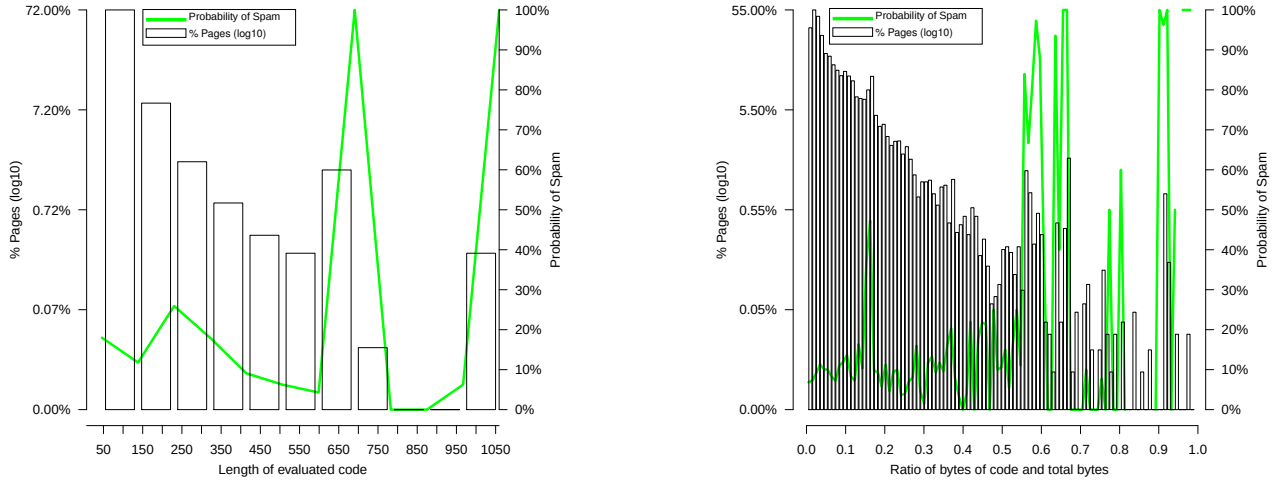
- Número de KeyWords/Description Words: una técnica muy usada es el "keyword stuffing", por ello analizamos las keywords de diferentes formas. Para ello, se realizó un análisis de la cantidad de palabras usadas en los atributos "keywords" y "description" de la etiqueta Meta, en páginas web de Spam y de no-Spam.

  Los resultados se muestran en la FiguraV.3a, donde podemos ver que páginas con menos de 130 keywords tienen menos de un 20% de probabilidad de ser Spam, y en páginas donde ese número va aumentando también aumenta la probabilidad de Spam. En base a estos resultados se decidió usar como heurísticas no solo las keywords en los atributos "keywords" y "description" de la etiqueta Meta, sino también el número de repeticiones de dichas palabras clave en los propios atributos y a lo largo del contenido de la página.

- Imágenes sin Alt: partiendo de la base de que el contenido de muchas páginas Web Spam son generados dinámicamente, se ha analizado el contenido del Alt de las imágenes. La Figura V.3b indica que para páginas web con entre 0 y 25 imagenes sin Alt, la probabilidad de Spam es menor del 10%. Superando esta cifra aparecen continuos picos de hasta el 60% y superando los 220 la probabilidad de Spam supera el 50%, teniendo picos del 100%.

- Longitud del código evaluado: se ha observado que las cadenas evaluadas en páginas con Web Spam son más largas de lo habitual, ya que contienen en muchos casos gran cantidad de variables, funciones y datos escapados, que se ejecutarán al mismo tiempo.

  En la Figura V.4a, se puede observar que la probabilidad de que una página web sea Spam es relativamente baja para valores entre los 0 y 600 bytes, por lo que es probable que esas páginas no sean Spam. Sin embargo, para valores entre los 600 y 800 bytes y a partir de los 950 bytes, la probabilidad aumenta progresivamente alcanzando incluso el 100% de probabilidad de Spam para algunos valores.



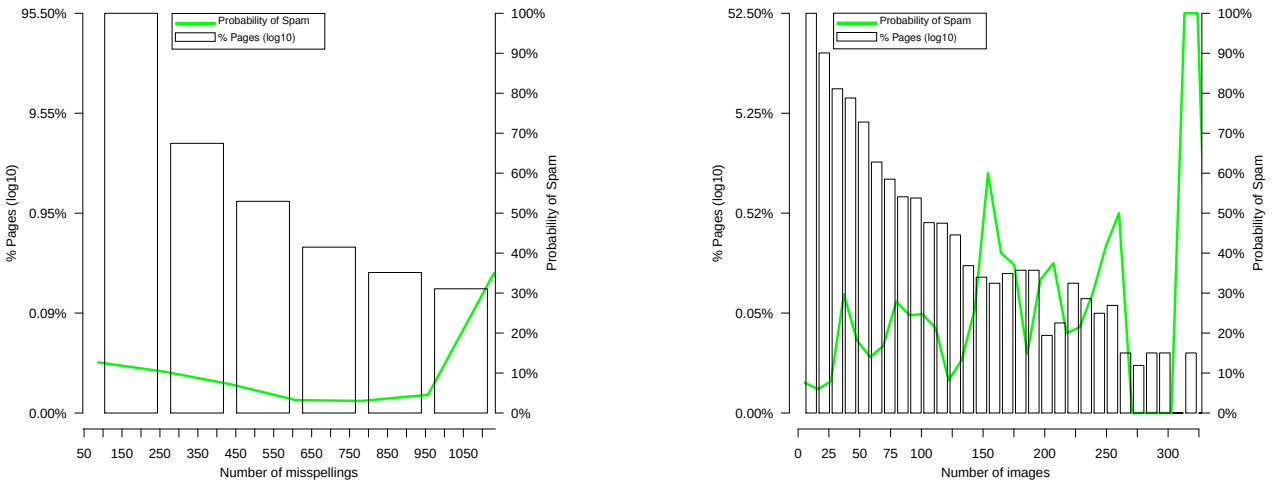
(a) (b)

Figura V.4: Probabilidad de Spam relativa a la longitud del código evaluado en una página web (a), y probabilidad de Spam relativa al ratio de bytes de código y bytes totales de la página web (b)

- Ratio de bytes de código y bytes totales: con la idea de seguir buscando métodos que ayuden a determinar si una página es Spam o no, durante el análisis de las páginas se observó que la relación entre el tamaño en bytes de código de scripting de una página, y los bytes totales es mayor cuanto mayor es su probabilidad de Spam. Es decir, cuándo la mayor parte del contenido HTML sea código, mayor será la probabilidad de que dicha página sea Spam.

  Los resultados mostrados en la Figura V.4b, indican que para valores entre 0 y 0.6, la probabilidad de que una página sea Spam es menor al 20%. Para valores superiores a 0.6, la probabilidad de que una página sea Spam aumenta hasta el 60%, 80%, 90% e incluso el 100%.

- Ortografía: como se ha explicado, las páginas de Web Spam suelen generar el contenido mediante "copy&paste" o de forma automática. Es por esto que al igual que en el ámbito del Email Spam, el índice de faltas de ortografía es superior al de una página no-Spam. En los resultados mostrados en la Figura V.5a, se observa que aquellas páginas web con pocas faltas de ortografía pueden ser consideradas como no-Spam.



(a) (b)

Figura V.5: Probabilidad de Spam relativa al número de errores ortográficos en una página web (a), y probabilidad relativa al número de imágenes en una página web (b)

- Número de imágenes: basándose en el hecho de que las páginas de Spam suelen generarse automáticamente, es habitual que wl númweo de imágenes sea superior al número de imágenes en páginas no-Spam. Tomando este hecho como base, medimos la cantidad total de imágenes en cada página.

  En la Figura V.5b se muestran los resultados obtenidos aplicando esta heurística. Se observa que para valores superiores a 125 la probabilidad de ser Spam aumenta progresivamente, alcanzando valores de 40%, 50%, 60% y 100%.

- Tamaño en bytes de la página web: se comprobó que el tamaño de muchas páginas web de Spam era superior a la media del tamaño de una página web normal. La Figura V.6a, indica que con valores entre 0 y 150000 bytes por página, la probabilidad de Spam es menor del 25%. Por otro lado, para valores

superiores a 150000 bytes, la probabilidad de ser Spam aumenta entre el 50% y el 88%.

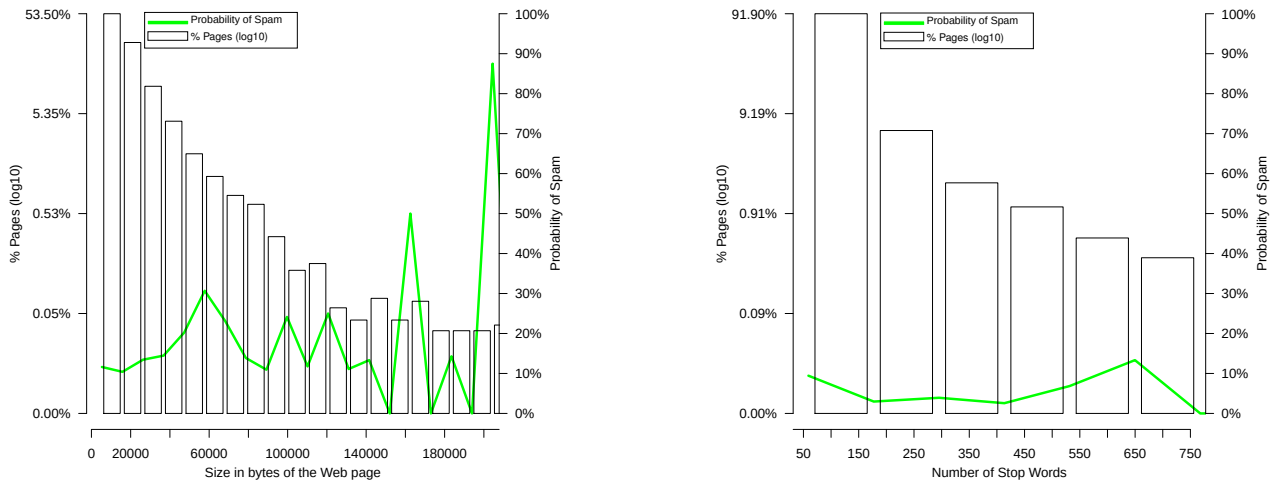


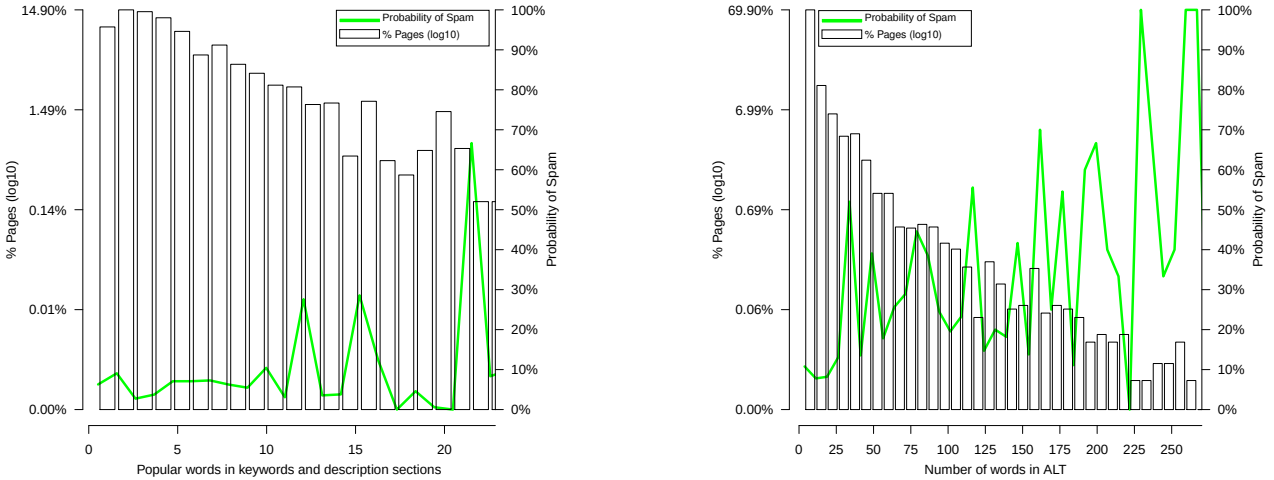
Figura V.6: Probabilidad de Spam relativa al tamaño en bytes de la página web (a), y probabilidad relativa al número de stopwords utilizadas en la página web (b)

- Número de stopwords: un método común de indexación por parte de los motores de búsqueda, es extraer las stopwords de la página web e indexarla. Por ello, ya que no son relevantes para los buscadores, los spammers optan por poner contenido inconexo, sin conjunciones, artículos, preposiciones u otras palabras comunes, es decir sin stopwords. Como técnica para detectar esto, y con ello el Web Spam, se propone analizar la cantidad de stopwords presentes en el contenido (sin etiquetas HTML). En la Figura V.6b se muestran los resultados obtenidos aplicando dicha heurística. Para páginas web con menos de 400 stopwords la probabilidad de Spam no supera el 5%, sin embargo, para valores entre 400 y 650 la probabilidad aumenta hasta el 15%.

- Palabras populares: en la sección V.1 se explico cómo detectar Web Spam midiendo de diferentes formas las palabras populares que formaban parte del contenido. En este caso se propone centrarnos exclusivamente en las sección de keywords y descriptionwords, ya que es el lugar donde más uso se hacen de palabras populares.

  Los resultados de esta heurística se muestran en la Figura V.7a, donde se observa que con valores

comprendidos entre 11 y 17, la probabilidad aumenta hasta el 25%. Para valores superiores a 21 la probabilidad de Spam crece hasta el 68%.



(a) (b)

Figura V.7: Probabilidad de Spam relativa al número de palabras populares en la página web (a), y probabilidad relativa al número de palabras en el atributo Alt en una página web (b)

- Número de palabras en el atributo Alt: Analizando el contenido de las páginas de Spam, se ha observado que es usual que el atributo Alt de las páginas web de Spam contenga un mayor número de palabras que las páginas web normales. Este hecho probablemente ocurre debido a que los spammers utilizan el atributo Alt para hacer "keyword stuffing".

  En la Figura V.7b, observamos los resultados obtenidos que muestran que aquellas páginas web con valores entre 0 y 150, la probabilidad de que sean Spam es inferior al 40%. Sin embargo, para valores superiores a 150, la probabilidad de ser Spam aumenta, alcanzando picos del 60%, 70% e incluso del 100%.

- Redirección Meta: en el estudio se ha detectado que es habitual el uso de redirecciones en la etiqueta Meta y usualmente con un tiempo de espera a la redirección inferior a 5 segundos. Por un lado los spammers ponen contenido en la página web con la finalidad de que los sistema de crawling la procesen e indexen. Por otro lado, redirigen al usuario a otra página con un contenido diferente de la que el sistema

de crawling procesa.

Por último, comentar que también se han considerado heurísticas basadas en características teóricas de páginas de Web Spam, como pueden ser la presencia de gran cantidad de texto oculto, utilización masiva de redirecciones, funciones de script, invocaciones dinámicas de funciones, activeX, etc. Sin embargo, estas heurísticas no se han mostrado tan efectivas como se esperaba (y menos que las presentadas previamente), por lo que no se comentan en detalle.

Como ya se ha dicho, algunas de las técnicas anteriores pueden ser usadas para fines lícitos, si bien el uso intensivo de ellas y su combinación constituyen un indicador determinante para la detección de páginas de Web Spam.

V.3 Método para la Detección de Web Spam

En esta sección se describe el método propuesto para la detección de Web Spam. El método utilizado se basa en el análisis de contenido como el propuesto por Ntoulas *et al.* [NM06] y Gonzalez *et al.* [GJC09]. Sin embargo se han considerado un conjunto de heurísticas (sección V.2) que permiten, analizando el contenido, detectar otros tipos de Web Spam como Cloacking, Link Spam, Redirection Spam, Malware Spam y Content Spam.

Para la combinación apropiada de las heurísticas propuestas, se han estudiado diferentes técnicas de clasificación: árboles de decisión, técnicas basadas en reglas, redes neuronales, etc. Finalmente, los mejores resultados se obtuvieron utilizando árboles de decisión, concretamente el algoritmo C4.5 [Qui93] [Qui96b].

Para mejorar los resultados del algoritmo también se evaluaron diversas técnicas. En contreto se utilizaron "bagging" [BB96] [Qui96a] y "boosting" [Qui96a]. Estas técnicas crean un conjunto de N clasificadores, combinando aquellos que obtengan los mejores resultados para construir un clasificador compuesto. La técnica de "bagging" crea N subconjuntos de n elementos aleatorios con reemplazo, es decir, los elementos elegidos para un subconjunto pueden aparecer en multiples subconjuntos, ya que la elección de los elementos para cada subconjunto se considera independiente. De esta forma se tendrán N clasificadores. Cada página web que desea ser clasificada será evaluada por cada uno de los N clasificadores. La clase en la que se introducirá la página web (Spam o no-Spam) depende de los votos de la mayoría de los N clasificadores. La técnica de "boosting" funciona de una forma similar. Cada uno de los ítems tiene un peso asignado, que indica la probabilidad de ocurrencia en el conjunto. Durante N iteraciones generará N clasificadores, pero para cada ítem mal clasificado su peso se aumenta. Nuevamente, la decisión final de si es Spam o no vendrá dada por la mayor parte de resultados de los N clasificadores.

La Figura V.8 muestra una porción del árbol de decisión que clasifica una página web como Spam. Cada nodo usa una de las heurísticas presentadas en la sección V.2 y, de acuerdo con los diferentes umbrales, decide si es Spam o no, o delega en el siguiente nodo para refinar la clasificación.

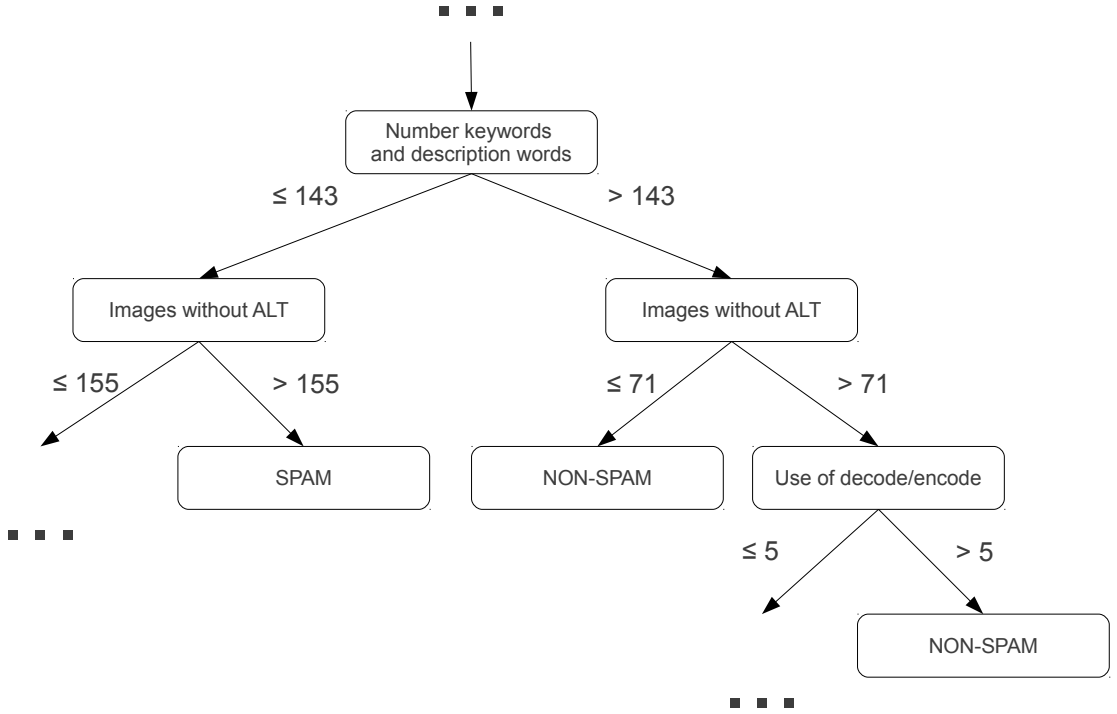


Figura V.8: Fragmento del árbol de decisión

En la sección V.4 se muestran los resultados en los dataset seleccionados y se comparan con otros importantes métodos existentes en la literatura.

V.4 Resultados de Detección Web Spam

En esta sección se explicarán las características de los datasets utilizados para el análisis de las heurísticas y para la obtención de los resultados del módulo. Posteriormente se explicarán diversas cuestiones que se han tenido en cuenta para la ejecución y obtención de los resultados. Por último, se analizarán y compararán los resultados obtenidos tanto a nivel de eficacia como de eficiencia.

V.4.1 Datasets

Para evaluar las diferentes heurísticas y el método propuesto para su combinación, se han usado dos datasets ampliamente conocidos y públicos:

- Webb Spam Corpus: creado por Webb *et al.* [Web06], [web11c], es considerado el mayor dataset de Web Spam, incluyendo mas de 350000 páginas web diferentes con Spam. Para su creación se usaron técnicas de detección de Email Spam para detectar correos de Spam, tras esto siguieron los enlaces que contenían dichos emails y almacenaron las páginas web obtenidas.
- WEBSPAM-UK2006/7: dataset creado por Yahoo! [yah11a], que incluye más de 100 millones de páginas web, de un total de 114529 máquinas, de los cuales se han etiquetado manualmente 6479 sitios web. El porcentaje de Spam en este dataset representa el 6%. Dicho dataset fue creado para los "Web Spam Challenge 2008" [Yah11b], reto en el que los diferentes grupos de investigación compiten para lograr los mejores resultados de detección de Spam.

Debido a que el primer dataset contenía exclusivamente páginas con Web Spam, se seleccionaron aleatoriamente aquellas páginas etiquetadas como no-Spam del dataset de Yahoo! y se combinaron con las de éste para tener un dataset con páginas de Spam y también páginas legítimas. El hecho de que sean datasets públicos permite a cualquier investigador evaluar y comprobar los métodos y resultados que mostramos, para validar su fiabilidad.

V.4.2 Configuración de los Experimentos

Para la realización de los experimentos, se ha desarrollado una herramienta que permita ejecutar y almacenar el resultado de cada una de las heurísticas explicadas anteriormente en cada página web analizada de los dataset seleccionados, así como el tiempo de procesamiento de cada una de las heurísticas. Estos datos permitieron comprobar qué heurísticas obtenían mejores resultados y cuales usaban menor tiempo de procesamiento. Esta herramienta también se encarga de filtrar las páginas de los datasets, impidiendo que páginas sin contenido, con contenido mínimo o cuyo contenido consistía únicamente en redirecciones JavaScript, se analicen.

Para la ejecución de los experimentos utilizamos la herramienta de aprendizaje automático y minería de datos WEKA [HFH+09]. Incluye diferentes tipos de clasificadores y diferentes algoritmos de cada uno de ellos.

Para evaluar el clasificador usamos "cross validation" [Koh95], técnica que consiste en construir k subconjuntos de datos. En cada iteración se construye y evalúa un modelo, usando uno de los conjuntos como "test set" y el resto como "training set". Hemos usado 10 como valor para la k ("ten-fold cross validation"), un valor ampliamente utilizado.

	Spam		No-Spam	
	Precision	Recall	Precision	Recall
C4.5				
Gonzalez et al.	0.895	0.715	0.962	0.988
Ntoulas et al.	0.894	0.743	0.965	0.988
Módulo Detección Web Spam	0.910	0.813	0.975	0.982
Módulo Detección Web Spam y Ntoulas	0.915	0.826	0.976	0.996
C4.5 - Bagging				
Gonzalez et al.	0.958	0.719	0.962	0.979
Ntoulas et al.	0.954	0.761	0.968	0.995
Módulo Detección Web Spam	0.969	0.815	0.975	0.996
Módulo Detección Web Spam y Ntoulas	0.968	0.828	0.977	0.990
C4.5 - Boosting				
Gonzalez et al.	0.932	0.790	0.972	0.992
Ntoulas et al.	0.942	0.791	0.993	0.949
Módulo Detección Web Spam	**0.972**	**0.850**	**0.978**	**0.997**
Módulo Detección Web Spam y Ntoulas	**0.982**	**0.851**	**0.979**	**0.998**

Tabla V.1: Resultados en el dataset 1 (22760 páginas)

Los experimentos realizados se han dividido en dos tipos según el dataset sobre el que iban dirigidos. Primero se obtuvieron resultados de las heurísticas propuestas por Ntoulas *et al.* [NM06] y Gonzalez *et al.* [GJC09] en el dataset de "Webb Spam Corpus" y se compararon por un lado con los resultados obtenidos por nuestro sistema y por otro con los obtenidos por nuestro sistema combinado con el de Ntoulas *et al.*. En el dataset de Yahoo! se hizo una comparativa de los resultados obtenidos por nuestro sistema y los resultados que obtuvieron los ganadores del Web Spam Challenge 2008.

V.4.3 Resultados Dataset Webb

En esta sección se analizan los resultados obtenidos en el dataset de Webb [Web06], [web11c]. Se ha evaluado utilizando diferentes tamaños del dataset (22760, 100000, 150000 y 200000 páginas) y usando el algoritmo C4.5 por si solo, y posteriormente combinado con técnicas de Bagging y Boosting.

La Tabla V.1 muestra los resultados obtenidos en el dataset 1 de 22760 páginas web (2760 páginas de Spam y 20000 de no-Spam) y la Tabla V.2 los obtenidos en el dataset 2 de 100000 páginas, 50000 páginas de Spam y otras 50000 de no-Spam. Los resultados obtenidos en el resto de datasets con diferentes tamaños, muestran datos consistentes con los que se discuten por lo que no se incluyen.

Para los resultados obtenidos exclusivamente con el algoritmo C4.5 se observa que nuestro algoritmo mejora el recall aproximadamente un 10%, en ambos datasets, sobre las técnicas de Gonzalez *et al.* y un 7% en el dataset 1 (Tabla V.1) y un 12.5% en el dataset 2 (Tabla V.2) sobre los resultados obtenidos por el algoritmo

	Spam		No-Spam	
	Precision	Recall	Precision	Recall
C4.5				
Gonzalez et al.	0.977	0.66	0.742	0.984
Ntoulas et al.	0.967	0.641	0.705	0.975
Módulo Detección Web Spam	0.96	0.766	0.797	0.969
Módulo Detección Web Spam y Ntoulas	0.936	0.715	0.744	0.944
C4.5 - Bagging				
Gonzalez et al.	0.990	0.669	0.749	0.993
Ntoulas et al.	0.981	0.650	0.712	0.986
Módulo Detección Web Spam	**0.982**	**0.750**	**0.797**	**0.986**
Módulo Detección Web Spam y Ntoulas	0.962	0.712	0.747	0.969
C4.5 - Boosting				
Gonzalez et al.	0.985	0.703	0.768	0.989
Ntoulas et al.	0.974	0.657	0.719	0.98
Módulo Detección Web Spam	**0.983**	**0.767**	**0.818**	**0.987**
Módulo Detección Web Spam y Ntoulas	0.949	0.776	0.810	0.958

Tabla V.2: Resultados en el dataset 2 (100000 páginas)

de Ntoulas *et al.*. La unión de las heurísticas de Ntoulas *et al.* y las nuestras muestra una mejoría de un 1.5% sobre los resultados comentados. En el caso de la precisión se mejora en torno a un 3% en el caso del dataset 1 (Tabla V.1).

En el caso de los resultados obtenidos aplicando Bagging se obtiene una mejora de aproximadamente un 10% en los resultados de Gonzalez *et al.* en el dataset 1 y un 8.2% en el dataset 2 (Tabla V.2). Nuevamente, se consigue mejorar en un 5.6% los resultados de Ntoulas *et al.* en el dataset 1 y un 10% en el dataset 2. En este caso el método propuesto obtiene resultados un 4% mejores que la unión de nuestras heurísticas con las de Ntoulas *et al.*

Por último, centrándonos en los resultados obtenidos aplicando Boosting, se observa que son mejores que los que los comentados anteriormente. En el dataset 1, se mejora en un 6% los resultados de Ntoulas *et al.* y Gonzalez *et al.* En el dataset 2, se consigue mejorar los resultados de Gonzalez *et al.* en un 6% y aproximadamente un 10% los de Ntoulas *et al.* En los resultados obtenidos uniendo las heurísticas propuestas y las de Ntoulas *et al.* sólo se consigue una mejora de un 1% sobre los resultados obtenidos aplicando unicamente las heurísticas del módulo de detección.

En la Figura V.9 se muestran un resumen con los mejores resultados obtenidos por cada algoritmo analizado. Los resultados indican que el método y heurísticas propuestas consiguen obtener los mejores resultados tanto en el dataset 1 como en el 2.

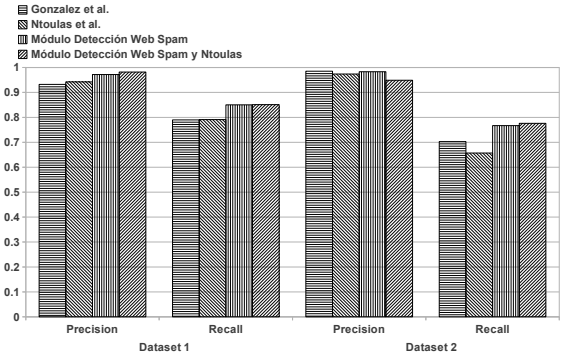


Figura V.9: Resumen de resultados obtenidos en la detección de Web Spam

V.4.4 Resultados Dataset Yahoo!

En esta sección se analizan los resultados obtenidos sobre el dataset de Yahoo!. En la sección anterior solo había dos clases, Spam y no-Spam. En este caso, Yahoo! ha creado una nueva clase denominada "undecided", para aquellas páginas web que están en el límite entre Spam y no-Spam.

Antes de mostrar los resultados comentar que únicamente se mostrarán los resultados de C4.5 aplicando Boosting ya que ha quedado patente en la sección anterior que es la técnica con la que mejores resultados se obtienen. Como se observa, tampoco se muestran los resultados de Ntoulas *et al.* y Gonzalez *et al.* en el Yahoo! Web Spam Challenge [Yah11b], ya que ha quedado demostrado, en el apartado anterior, que nuestros métodos obtienen mejores resultados. Otra cuestión a tener en cuenta es que mostraremos y compararemos el área ROC ya que en los resultados del Yahoo! Web Spam Challenge es el método que utilizan.

Hemos creado dos subconjuntos a partir del dataset de Yahoo!, uno de 20000 páginas web y otro de 50000. Los datasets se han creado eligiendo páginas de forma aleatoria y siempre manteniendo el 6% de Spam que indicaba Yahoo!.

En la Tabla V.3 podemos observar los resultados obtenidos para ambos datasets. El primero, de 20000 páginas, contiene 17679 páginas de no-Spam, 1270 de Spam y 1043 Undecided. El segundo dataset tiene 50000 páginas, de las cuales 45584 son no-Spam, 1368 Spam y 3048 Undecided. Los datos obtenidos son esperanzadores y superan los resultados obtenidos por los ganadores del Yahoo! Web Spam Challenge. El área ROC en la detección de Spam se situa entre 0,997 y 0,996, similares resultados, alrededor de 0,99, se obtienen para las clases de no-Spam y Undecided.

La Figura V.10 muestra claramente las mejoras en los resultados obtenidos con respecto a los resultados obtenidos en el Web Spam Challenge de Yahoo!. Se ha conseguido mejorar los resultados en un 15% en el peor de los casos y en torno a un 27% en el mejor de los casos. Yahoo! no incluye los resultados de detección de

Dataset	20000			
	Clase	Precision	Recall	ROC
C4.5 - Boosting	Spam	0.992	0.966	0.997
	No-Spam	0.99	0.999	0.991
	Undecided	0.985	0.87	0.989

Dataset	50000			
	Clase	Precision	Recall	ROC
C4.5 - Boosting	Spam	0.999	0.971	0.996
	No-Spam	0.996	0.999	0.996
	Undecided	0.986	0.953	0.995

Tabla V.3: Resultados obtenidos en el dataset de Yahoo!

no-Spam y Undecided, por lo que no hemos podido realizar la comparación.

V.4.5 Eficiencia de las Heurísticas

En las secciones anteriores se ha realizado un análisis detallado del conjunto de nuevas heurísticas para la detección de Web Spam que se proponen, desde el punto de vista de su eficacia para la detección de Spam. En esta sección se estudiarán las heurísticas desde el punto de vista de su rendimiento. Esto ayudará a realizar diversas agrupaciones de heurísticas teniendo en cuenta tanto su eficacia como su rendimiento.

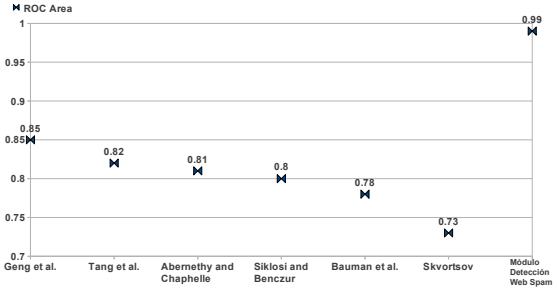


Figura V.10: Comparación de los resultados en base al área ROC

En la Figura V.11 se muestran los tiempos de cómputo requeridos para obtener los resultados para las principales heurísticas. En concreto se muestra el tiempo medio de cada heurística al aplicarse sobre una

página. Algunas de las heurísticas analizadas no se muestran en la figura debido a que los tiempos de ejecución obtenidos son despreciables.

Los resultados indican que las heurísticas que más tiempo utilizan son aquellas que realizan un análisis de las palabras y sus relaciones, como contar el número de palabras populares, el nivel de compresión de una página o su contenido visible con respecto al total. Por otro lado, se observa que las heurísticas que menos recursos necesitan son aquellas basadas en análisis más sencillos como el número total de imágenes, el número de imágenes sin Alt o el tamaño en bytes de la página. La mayoría de las heurísticas no llegan a consumir más de 0.05 segundos en su procesamiento, y muchas de ellas tardan menos de 1 microsegundo. No obstante tenemos que destacar que no mostramos el tiempo de proceso de la heurística de detección de faltas ortográficas, ya que consumía del orden de 1.2 segundos por página, lo que supone cuatro veces más tiempo que la heurística más costosa de las que mostramos, por lo que dicha heurística fue descartada.

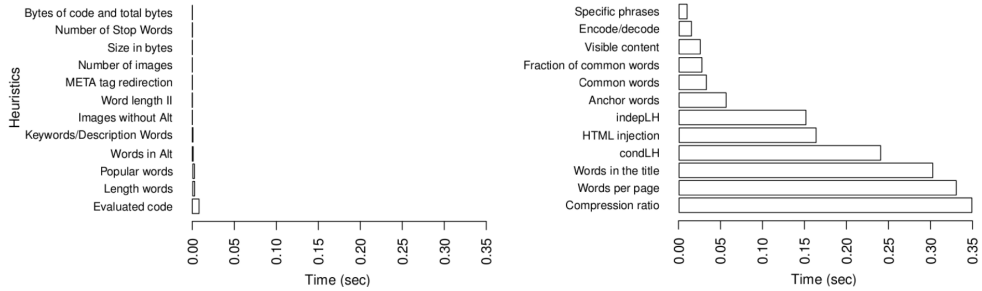


Figura V.11: Eficiencia de las heurísticas presentadas

V.4.6 Resultados de Eficiencia de las Heurísticas Propuestas

Los resultados obtenidos en el análisis de la eficiencia de las heurísticas, organizan las heurísticas según su rendimiento. De esta forma, dependiendo de los recursos disponibles y la seguridad que necesite el sistema, se seleccionará un conjunto diferente de heurísticas. Esto provocará que el sistema sea más o menos seguro y menos o más rápido respectivamente. En esta sección se proponen diferentes agrupaciones de heurísticas con el fin de analizar y comparar sus resultados desde el punto de vista de su eficiencia con los otros algoritmos existentes.

Basándose en los resultados mostrados en la sección anterior, donde se muestra el tiempo medio de cómputo para cada una de las heurísticas estudiadas, hemos creado 5 agrupaciones de heurísticas. Las dos primeras son las que proponen Ntoulas *et al.* [NM06] y Gonzalez *et al.* [GJC09], y los 3 siguientes conjuntos están formados por subconjuntos de heurísticas que hemos propuesto.

	Spam		No-Spam		
	Precision	Recall	Precision	Recall	
C4.5					Tiempo medio (segundos)
Gonzalez et al.	0.895	0.715	0.962	0.988	0.715
Ntoulas et al.	0.894	0.743	0.965	0.988	1,399
Módulo Detección Web Spam #1	0.897	0.699	0.955	0.989	0,0099
Módulo Detección Web Spam #2	0.911	0.734	0.964	0.99	0,187
Módulo Detección Web Spam #3	0.915	0.826	0.976	0.996	0,210

Tabla V.4: Resultados de eficiencia

	Nivel de seguridad del sistema		
Nivel de uso de recursos del sistema	Alto	Medio	Bajo
Alto	#1	#1	#1
Medio	#2	#2	#1
Bajo	#3	#2	#1

Tabla V.5: Políticas de detección de Web Spam y uso de recursos

- Módulo de Detección Web Spam #1: longitud de las palabras II, imágenes sin Alt, frases específicas.
- Módulo de Detección Web Spam #2: funciones encode/decode, inyección HTML, frases específicas, imágenes sin Alt, número keywords/description words, longitud de las palabras II.
- Módulo de Detección Web Spam #3: todas las heurísticas explicadas en la sección V.2.

La idea que se ha seguido para la realización de las agrupaciones ha sido seleccionar para el primer grupo aquellas heurísticas que obtienen resultados relativamente buenos, y cuyo tiempo medio de procesamiento es muy bajo. Para los siguientes grupos se ha aumentando progresivamente el número de heurísticas que funcionan mejor, pero que su tiempo medio de procesamiento es mayor.

En la Tabla V.4, se observa que el módulo de detección propuesto obtiene los mejores resultados a nivel de precision y recall (a excepción de Módulo Detección Web Spam #1) y su tiempo medio de procesamiento es muy inferior a los propuestos por Ntoulas *et al.* y Gonzalez *et al.* Centrándonos en el tiempo promedio de procesamiento se ve que aún eligiendo la política #3, las heurísticas propuestas son más de 6 veces más rápidas que las de Ntoulas *et al.* y casi 3 veces más rápidas que las técnicas propuestas por Gonzalez *et al.* En el caso de la política #1, las diferencias son mucho mayores, entre 141 y 72 veces más rápidos que los otros conjuntos de heurísticas.

Estos resultados permiten crear las 3 políticas de actuación que se muestran en la Tabla V.5. Dependiendo de la situación en la que se encuentra el sistema se aplicará una u otra de estas políticas. Como podemos ver,

el sistema prioriza el nivel de recursos que tiene disponibles frente a una detección óptima de Web Spam. En el caso en que el crawler este en una situación de alto uso de recursos, la configuración que utilizará será #1, de tal forma que el sistema reducirá el número de recursos usados para la detección de Web Spam y dejará recursos libres para que otros procesos puedan utilizarlos.

V.5 Conclusiones y Comparación de Resultados

El módulo de detección de Web Spam contiene un conjunto de novedosas heurísticas que aumentan y mejoran las presentes en el estado del arte. Las heurísticas presentadas no tratan las relaciones entre páginas o sitios Web, sino que analizan por un lado el contenido web, y por otro, tratan de detectar Web Spam analizando el código de scripting que contenga la página web.

Los métodos discutidos han sido probados contra dos datasets públicos y ampliamente conocidos. Por una parte, en un dataset creado a partir de Email Spam (Webb Spam Corpus), que exclusivamente contiene páginas con Spam, y por otra parte, en un dataset de Yahoo!, etiquetado de forma manual.

Para la combinación de las heurísticas se han utilizado árboles de decisión, concretamente el algoritmo C4.5. Tras esto se ha conseguido mejorar los resultados aplicando Bagging y Boosting. Los mejores resultados se han obtenido aplicando C4.5 y Boosting.

El módulo de detección logra mejorar entre un 6% y un 10% los resultados presentados por Ntoulas *et al.* y Gonzalez *et al.* En el caso del dataset de Yahoo! [yah11a], se ha conseguido un área ROC de hasta 0.99, es decir entre un 15% y un 27% mejor que los resultados que obtuvieron los ganadores del Web Spam Challenge 2008 [Yah11b].

Cada una de las técnicas presentadas podría identificar por sí sola una página de Web Spam, pero también existiría la posibilidad de clasificarla como tal y que en realidad se tratase de una página legítima. Del mismo modo, si confiáramos en las técnicas de forma independiente le sería sencillo a un spammer saltarse cada una de ellas. Por ello, se han combinado todas las heurísticas, creando un detector más robusto. De este modo, se mejora mucho la probabilidad de éxito en detección de Web Spam, y a mayores se dificulta el hecho de que un spammer pueda evitar a la vez todas las heurísticas.

Debido a que las heurísticas presentadas van a ser usadas en un sistema de crawling donde limitar el uso de recursos es una tarea importante, se han analizado cada una de las heurísticas analizadas desde el punto de vista del rendimiento. Se ha determinado qué heurísticas eran más o menos eficientes. Partiendo de este estudio hemos creado 3 subconjuntos de heurísticas, balanceando en cada uno de ellos la probabilidad de éxito detectando Web Spam y el rendimiento de las heurísticas usadas. El subconjunto #1 obtiene resultados de detección similares a Ntoulas *et al.* y Gonzalez *et al.* y consigue ser entre 141 y 72 veces mas rápido que ellos. En el caso del conjunto de heurísticas #3, sobre las cuales se han obtenido los resultados de eficacia mostrados,

consigue ser entre 6 y 3 veces mas rápido que los citados estudios.

En definitiva, a pesar de que el uso del módulo de detección provoca un consumo de recursos, el uso del módulo propuesto en los sistemas de crawling permitirá mejorar la eficiencia del sistema, ya que ahorrará recursos al no analizar, indexar, e incluso no mostrar como resultado, aquellas páginas que son Spam. Además, las políticas definidas permiten ajustar el comportamiento del crawler a las necesidades y recursos de cada momento.

VI

Módulo de Detección de Páginas Soft-404

Este capítulo describe detalladamente el módulo de detección de páginas Soft-404, compuesto por: un conjunto de técnicas innovadoras para detectar páginas Soft-404 y un método que permite combinarlas adecuadamente para mejorar su capacidad de detección.

En la sección VI.1 se explican las técnicas de detección existentes. En la sección VI.2 se analizan y evalúan cada una de las nuevas heurísticas que se proponen para la detección de páginas Soft-404. Se estudia cada una de ellas de forma independiente, evaluando la probabilidad de Soft-404 asociada. La sección VI.3 explica el método creado, a partir de la combinación de diferentes heurísticas, para la detección de páginas Soft-404. En la sección VI.4 se describen los datasets utilizados para la realización de los experimentos, y la importancia de este tipo de páginas en la Web. Tras esto se discuten los resultados obtenidos por el sistema y se comparan con otros presentes en la literatura. Por último, se analizan los tiempos de ejecución de las heurísticas y la eficiencia computacional de las mismas. En la sección VI.5 se resumen los resultados obtenidos, comparándolos con los sistemas existentes, y se analizan las conclusiones.

VI.1 Técnicas Existentes de Detección de páginas Soft-404

En esta primera sección se explicarán brevemente las técnicas presentes en la literatura para la detección de páginas Soft-404. Estas técnicas han sido explicadas en mayor detalle en la sección II.3.2.

Existen dos estudios que se centran en la problemática de la detección de páginas Soft-404. El primero de ellos es el estudio llevado a cabo por Bar-Yossef *et al.* [BYBKT04]. Los autores proponen un algoritmo para la detección de este tipo de páginas. Dicho algoritmo se basa en la realización de dos peticiones HTTP, una a la página que desea ser analizada y otra a una página del mismo sitio web pero generada de forma aleatoria con el fin de maximizar la probabilidad de que no exista. De esta forma conocerá cómo se comporta el servidor web ante peticiones de recursos no existentes, y podrá compararlo con el comportamiento ante la página analizada. Un servidor web ante peticiones de páginas web no existentes deberá responder, en base al protocolo HTTP, con un código de error 404. En este caso, el algoritmo clasificará la página analizada como una página normal, es decir no-Soft-404. Si, en cambio, el servidor ha enviado un código HTTP 200, el algoritmo comparará el número y el contenido de las redirecciones realizadas en cada una de las dos peticiones. En caso de ser necesario, realizará una comprobación del contenido de las dos páginas que el servidor ha devuelto, para finalmente poder decidir si dicha página es clasificada como Soft-404 o no.

El otro estudio que trata la detección de páginas Soft-404 es el realizado por Lee *et al.* [LKK+09]. Dicho estudio analiza las limitaciones del estudio de Bar-Yossef *et al.* [BYBKT04] y propone una serie de mejoras. Los autores tratan de detectar este tipo páginas desde un punto de vista diferente, analizando los logs de un sistema de crawling. Para ello definen 3 heurísticas: a) el número de redirecciones de cada host, b) la relación entre el número de páginas y el número de redirecciones desde este host y c) el número de hosts distintos alcanzados desde redirecciones de este host.

VI.2 Técnicas propuestas para la Detección de Páginas Soft-404

Los técnicas existentes presentan importantes limitaciones e inconvenientes que han sido discutidas en la II.3.1. Además, en las secciones VI.4.4 y VI.4.6 se demostrará que el método propuesto obtiene mejores resultados a nivel de eficacia y eficiencia.

Para solucionar los problemas que presentan (ver sección II.3.1), se propone un método de detección basado en el análisis del contenido de las páginas web, en lugar de en el análisis de peticiones HTTP o de logs de los crawlers. Para ello, se han analizado páginas Soft-404 para definir un conjunto de características comunes a este tipo de páginas web. En esta sección se analizan un conjunto de heurísticas de detección para páginas Soft-404, basadas en las características previamente observadas.

Para justificar la eficacia de cada una de ellas, se han aplicado sobre los datasets descritos en la sección VI.4, mostrando la probabilidad de que sea una página Soft-404 en función de los diferentes valores posibles que se han obtenido para cada una de ellas, y la fracción de páginas a las que afecta cada valor. Para mejorar la representación de los datos del eje que indica la fracción de páginas afectadas por cada vector de cada heurística, hemos usado una progresión log10, pero mostrando el porcentaje y no el valor absoluto.

- Ratio de bytes de contenido útil de la página y bytes totales: esta heurística se centra en el hecho de que las páginas Soft-404 tienen poco contenido útil. Para demostrar este hecho, se analiza la relación entre los bytes de contenido útil y los bytes totales de la página.

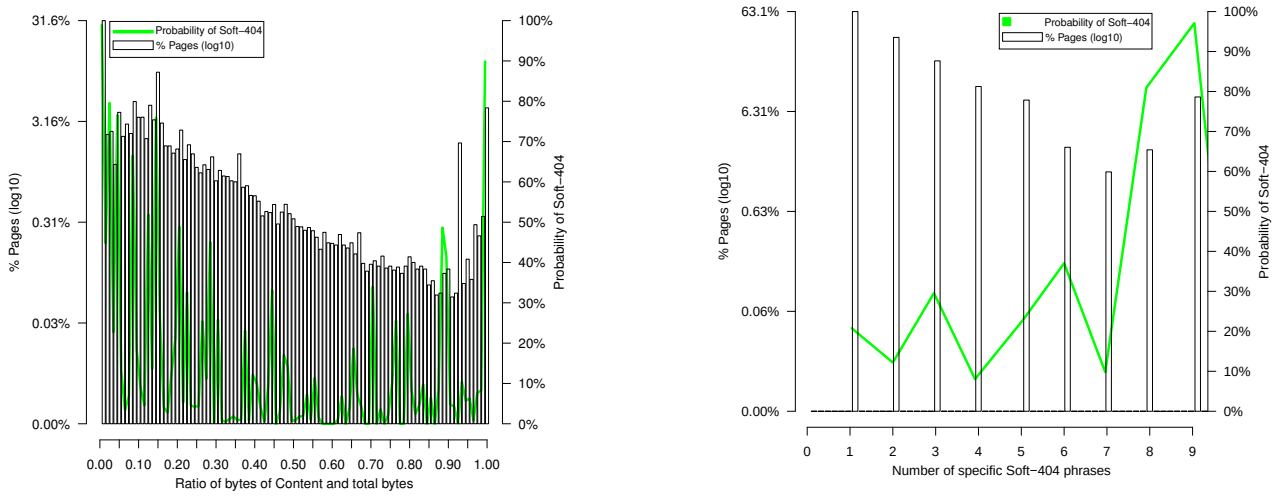
  En la Figura VI.1a se puede observar que con ratios de contenido útil y bytes totales inferiores a 0.30 la probabilidad de Soft-404 asciende progresivamente obteniendo picos de 60%, 80% y hasta 100%.

- Frases/Palabras específicas: se ha observado que es común que las páginas Soft-404 contengan palabras o frases comunes, entre las que se puede destacar: “urgent”, “removed”, “error” y “404”.

  Los resultados mostrados en la Figura VI.1b indican que con páginas con 5 o más palabras del listado comentado, la probabilidad de que una página sea Soft-404 asciende progresivamente desde 30% hasta casi el 100%.

- Número de “description words”: como se ha demostrado en las heurísticas anteriores, las páginas Soft-404 son páginas con contenido escaso y de mala calidad. Es por esto por lo que se analiza el número de palabras que contiene el atributo “description” de la etiqueta Meta de HTML, ya que este tipo de páginas contendrán un número de palabras, en dicho atributo, inferior al obtenido en un conjunto de páginas normales.

  Analizando los resultados que muestra la Figura VI.2a se puede decir que para valores inferiores a 10 obtiene una probabilidad de ser una página Soft-404 del 75%, y para valores superior la probabilidad desciende, en general, a valores por debajo del 20% y del 10%.



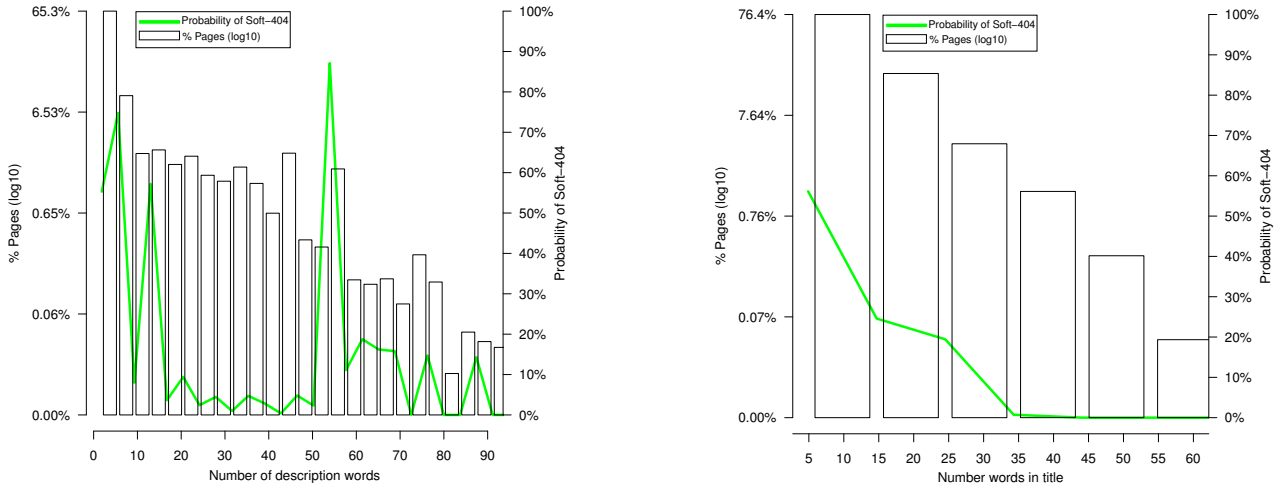
(a) (b)

Figura VI.1: Probabilidad de Soft-404 relativa al ratio del contenido útil de la página y el contenido total de la misma (a), y probabilidad de Soft-404 relativa al número de frases especificas de páginas Soft-404 usadas en la página web (a)

- Número de palabras en el título: analizando el conjunto de páginas Soft-404, se observó que gran parte de las páginas no tenían título o su título era más corto que en páginas normales. En la Figura VI.2b se observa que con títulos de página inferiores a 10 palabras la probabilidad de que sea una página Soft-404 es del 40%. Con valores inferiores a 5 palabras dicha probabilidad llega hasta el 60%.

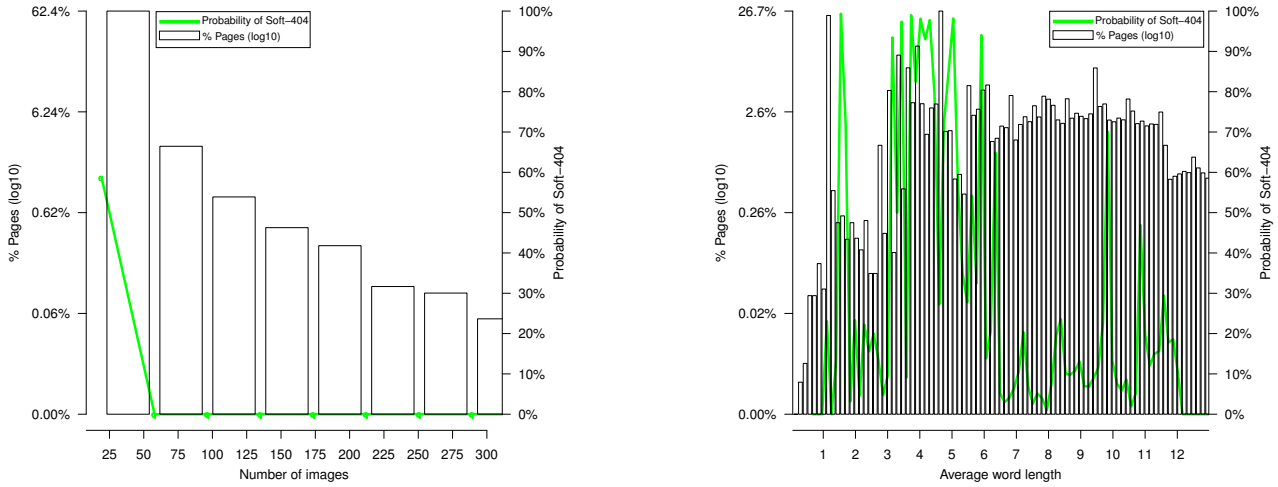
- Imágenes por página: en base a las propiedades observadas en páginas Soft-404, se observó que se trataba de páginas poco elaboradas y casi sin ningún contenido. Siguiendo esta idea observamos que no es común que este tipo de páginas contengan imágenes, y en caso de tener imágenes su número de ocurrencias es inferior al de las páginas normales.

  Los resultados de esta heurística se resumen en la Figura VI.3a. Se puede ver que en páginas con menos de 50 imágenes, la probabilidad de ser una página Soft-404 asciende hasta el 60%. Esta no es una heurística que determine inequivocamente qué páginas son Soft-404, pero sí es de gran ayuda a la hora de determinar si es una página normal, ya que páginas con más de 50 imágenes, la probabilidad de ser Soft-404 es del 0%.



(a) (b)

Figura VI.2: Probabilidad de Soft-404 relativa al número de description words utilizadas en la página web (a), y probabilidad de Soft-404 relativa al número de palabras utilizadas en el título de la página web (b)



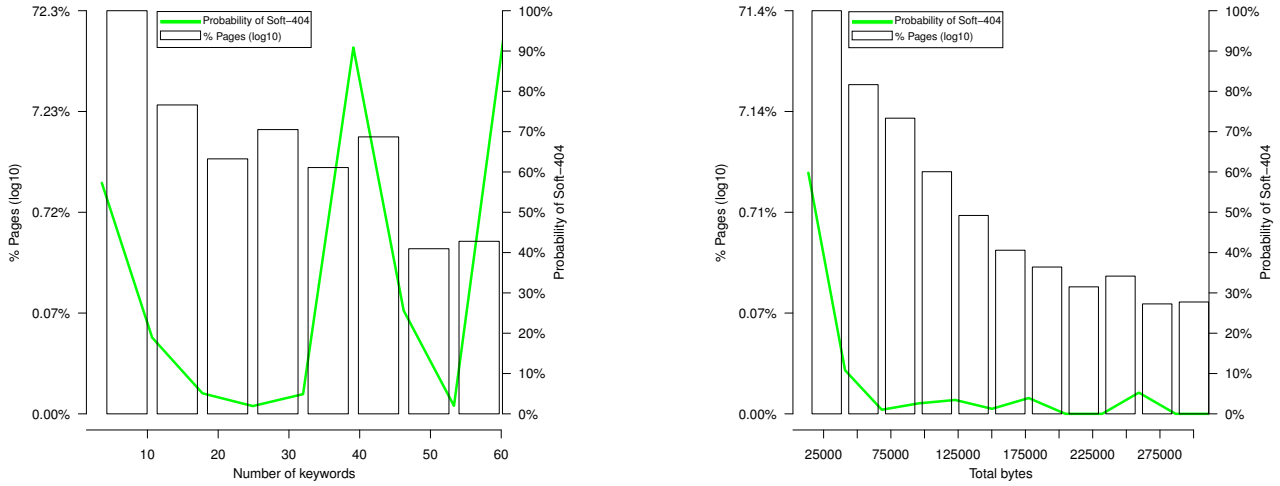
(a) (b)

Figura VI.3: Probabilidad de Soft-404 relativa al número de imágenes por página (a), y probabilidad de Soft-404 relativa a la longitud media de las palabras utilizadas en la página web (b)

- Longitud media de palabras: esta heurística sigue la idea de que la longitud de las palabras que contienen las páginas Soft-404 tiende a ser baja, debido a que no contiene frases elaboradas ni complejas. En muchas ocasiones están formadas por "stopwords", cuya longitud es pequeña.

  La Figura VI.3b muestra que para valores entre 1.5 y 3, y a partir de 3 hasta 6.5 la probabilidad de ser una página Soft-404 aumenta hasta valores del 90% o 100%, respectivamente. Para valores superiores a 6.5 vemos que la probabilidad desciende hasta el 20%.

- Número de "keywords": continuando con la idea de la heurística de "description words", se estudiaron las páginas Soft-404 analizando el contenido del atributo "keywords" de la etiqueta Meta. Observando los datos de la Figura VI.4a vemos que con valores inferiores a 10 la probabilidad asciende progresivamente hasta casi el 60%. Para valores de entre 30 y 45 keywords, y superiores a 55 la probabilidad vuelve a subir nuevamente hasta alcanzar picos del 90%. Esto puede ser debido a páginas de parking que devuelven Soft-404 y pretenden hacer "keywords stuffing" de su dominio para lograr aumentar su tráfico y conseguir beneficios mediante publicidad.



(a) (b)

Figura VI.4: Probabilidad de Soft-404 relativa al número de keywords por página (a), y probabilidad de Soft-404 relativa al tamaño en bytes de la página web (b)

- Tamaño en bytes de la página web: se ha estudiado el tamaño de las páginas Soft-404, demostrado que éstas tienen un tamaño en bytes inferior a las páginas normales. La Figura VI.4b demuestra que con valores inferiores a 50000 bytes la probabilidad de que una página sea Soft-404 asciende hasta probabilidades del 60%.

VI.3 Método para la Detección de páginas Soft-404

En esta sección se describe el método propuesto para la detección de páginas Soft-404. Se basa en análisis de contenido de la página web, aplicando las diferentes heurísticas propuestas. De esta forma pretendemos caracterizar dicho tipo de páginas para poder clasificar una página desconocida como Soft-404 o como no-Soft-404.

Una vez obtenidos los resultados para cada heurística, se estudiaron diferentes técnicas de clasificación (arboles de decisión, técnicas basadas en reglas, redes neuronales, etc.), para optimizar la combinación de nuestras heurísticas de detección de páginas Soft-404. Teniendo en cuenta los resultados, nos decantamos por los arboles de decisión, concretamente por el algoritmo C4.5 [Qui93] [Qui96b]. El algoritmo utilizado y las técnicas para la mejora de la detección, son similares a los usados para la detección de Web Spam (capítulo V).

La Figura VI.5 muestra una parte del árbol de decisión generado para clasificar páginas Soft-404. Cada uno de los nodos representa una de las heurísticas presentadas en la sección VI.2. De acuerdo a los diferentes umbrales, decidirá si se trata o no de una página Soft-404, o delegará en otro nodo para continuar la clasificación.

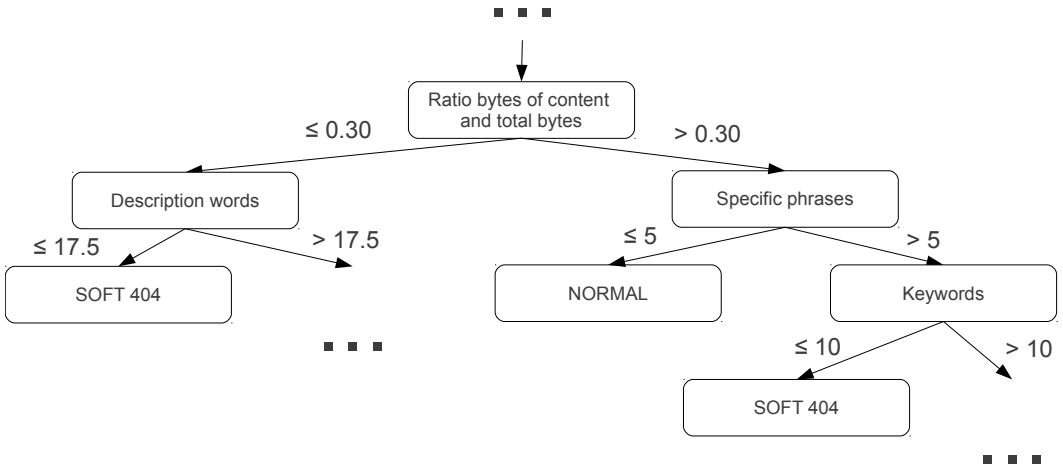


Figura VI.5: Porción del árbol de decisión

VI.4 Resultados Detección Soft-404

En esta sección describimos los datasets utilizados para la realización de los experimentos, y se estudia la importancia de la presencia de este tipo de páginas en la Web. Tras esto se comentará la configuración

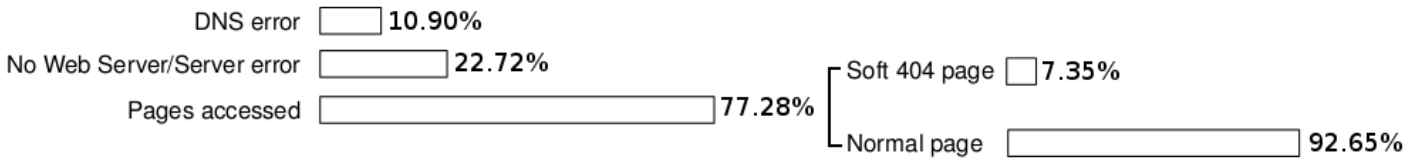


Figura VI.6: Páginas Soft-404 presentes en la Web

utilizada para la realización de los experimentos y se discuten y analizan en detalle los resultados obtenidos por el sistema. También se hará una comparativa entre los resultados obtenidos por el módulo de detección y los resultados de las técnicas presentes en la literatura. Por último, se analizan los tiempos de ejecución de las heurísticas y la eficiencia computacional de las mismas.

VI.4.1 Datasets

Para evaluar cada una de las heurísticas propuestas y para la ejecución de los experimentos, se ha construido un dataset con páginas normales (no-Soft-404) y páginas Soft-404.

Para ello se partió de un listado de 680000 dominios existentes del dominio nacional .es, obtenidos de "Red.Es"[1]. Para obtener las páginas Soft-404 creamos un sistema que comprobó que el servidor DNS responde ante dicho dominio y que tras la IP correspondiente existe un servidor web. Tras esto, el sistema genera peticiones a una página web aleatoria de más de 30 caracteres alfanuméricos (lo que asegura que la posibilidad de que dicha página exista sea extremadamente pequeña) a los dominios del listado de .es, de tal forma que se espera que el servidor web responda con un código HTTP 404. En los casos en los que el servidor web responda con un código 200, estaremos ante una página Soft-404, en cualquier otro caso consideramos que se trata de una página normal (no-Soft-404). Los resultados obtenidos se muestran en la Figura VI.6.

Analizando los resultados se observa que el 89.10% de los dominios ha respondido a la petición DNS y que de estos el 22.72% no tenían un servidor web o han respondido con algún tipo de error. Del subconjunto de páginas con servidor web (77.28%), el 7.35% ha respondido con una página Soft-404 cuando se le ha hecho una petición a una página desconocida. Este dato demuestra que las páginas Soft-404 constituyen un problema significativo para la Web.

Partiendo de los conjuntos de páginas Soft-404 y páginas normales (no-Soft-404), se han construido dos datasets. El primero de ellos, llamado #1, contiene 52000 páginas Soft-404 extraídas del dominio .es. Para completar este dataset con páginas normales (no-Soft-404) se ha combinado con un conjunto de páginas del dominio .es seleccionadas aleatoriamente. El segundo dataset fue generado partiendo de la base de las 52000

[1] http://www.red.es

páginas Soft-404, pero combinándolo con páginas no-Soft-404 de la Web Global, obtenidas del "Stanford WebBase Project"[2], que forma parte del "The Stanford Digital Libraries Technologies Project"[3]. Para la generación de los citados datasets no se han usado exclusivamente páginas raíz, ya que como es lógico esto sesgaría el resultado.

A partir de esos dos datasets, se han construido 4 subconjuntos, uno de ellos siguiendo la probabilidad de páginas Soft-404 en la Web, estudiada por Bar-Yossef *et al.* en su artículo [BYBKT04], el 25%, y otro subconjunto asumiendo que el 50% de las páginas son Soft-404. En resumen, estos son los datasets que han sido utilizados:

- Dataset #1-A: únicamente páginas .es con un 25% de páginas Soft-404 (50,000 páginas normales y 15625 Soft-404).
- Dataset #1-B: únicamente páginas .es con un 50.00% de páginas Soft-404 (50,000 páginas normales y 50000 Soft-404).
- Dataset #2-A: páginas de la Web Global con un 25% de páginas Soft-404 (50,000 páginas normales y 15625 Soft-404).
- Dataset #2-B: páginas de la Web Global con un 50.00% de páginas Soft-404 (50,000 páginas normales y 50000 Soft-404).

Con el dataset B se pretende demostrar que las heurísticas propuestas no dependen del número de páginas Soft-404 con las que se entrene el clasificador, es decir, con la probabilidad de que una página Soft-404 aparezca en la Web. Por último, con los datasets #1 y #2, se quiere demostrar que las heurísticas no dependen del lenguaje o de otra característica exclusiva de un país.

VI.4.2 Configuración de los Experimentos

Para la evaluación experimental de las técnicas propuestas, se ha desarrollado una herramienta que permita ejecutar y almacenar el resultado de cada una de las heurísticas, junto con el tiempo de procesamiento de cada una de ellas.

Del mismo modo que en el caso de la detección de Web Spam (capítulo V), para la ejecución de los experimentos se ha utilizado la herramienta WEKA [HFH+09]. Los resultados también fueron evaluados mediante el uso "cross validation" [Koh95], usando un valor de $k = 10$.

[2] http://dbpubs.stanford.edu:8091/~testbed/doc2/WebBase/
[3] http://diglib.stanford.edu:8091/

Los experimentos se han realizado sobre el dataset #1 y #2. Para demostrar la validez de los resultados obtenidos, y que estos no dependen del conjunto de datos de entrenamiento, se han llevado a cabo experimentos sobre los datasets pero utilizando diferentes tamaños del mismo como conjunto de entrenamiento (50%, 25%, 15%, 10%, y 5%).

En resumen, para la obtención de los resultados, se ha combinado: a) datasets de diferentes tamaños, b) páginas web de la Web Global y Española, c) diferentes técnicas para obtener los resultados y d) diferentes tamaños del conjunto de entrenamiento.

VI.4.3 Eficacia del Sistema

A continuación se muestran y discuten los resultados obtenidos sobre los datasets descritos anteriormente. Las tablas VI.1 y VI.2 muestran los resultados para los datasets #1 y #2 respectivamente.

	Soft-404		Página normal	
	Precision	Recall	Precision	Recall
Dataset #1-A				
C4.5	0.806	0.947	0.983	0.932
C4.5 - Bagging	0.813	0.937	0.981	0.936
C4.5 - Boosting	**0.807**	**0.947**	**0.983**	**0.937**
Dataset #1-B				
C4.5	0.922	0.980	0.979	0.918
C4.5 - Bagging	0.927	0.980	0.980	0.924
C4.5 - Boosting	**0.928**	**0.983**	**0.983**	**0.952**

Tabla VI.1: Resultados en el dataset #1

Analizando los resultados del dataset #1-A (Tabla VI.1) se observa que el mejor recall es de 0.947, y se consigue aplicando el algoritmo C4.5 y Boosting. Sin embargo, la mejor precisión es de 0.813 y se logra aplicando C4.5 y Bagging, aunque este resultado es inferior a los obtenidos en los siguientes datasets.

En el caso del dataset #1-B (Tabla VI.1) los resultados mejoran. Se obtiene una precisión de 0.928 y 0.983 en páginas Soft-404 y páginas normales, respectivamente, y un recall de 0.983 y 0.952 en páginas Soft-404 y páginas normales. Estos resultados se obtienen aplicando el algoritmo C4.5 y Boosting. Los resultados obtenidos demuestran que las heurísticas propuestas son adecuadas para detectar páginas Soft-404.

La Tabla VI.2 muestra los resultados obtenidos en el dataset #2. En dicho dataset el recall mejora con respecto al resultado obtenido en el dataset #1 hasta alcanzar 0.973. Nuevamente los resultados obtenidos en ambos datasets (#2-A y #2-B) confirman la idea de que las heurísticas que se han propuesto son adecuadas y que no dependen del número de páginas Soft-404 que contenga el conjunto de prueba.

La precisión obtenida en el dataset #1 es menor, ya que el dataset 1 contiene un mayor número de páginas

	Soft-404		Página normal	
	Precision	Recall	Precision	Recall
Dataset #2-A				
C4.5	0.980	0.960	0.979	0.918
C4.5 - Bagging	0.986	0.963	0.996	0.999
C4.5 - Boosting	**0.994**	**0.973**	**0.997**	**0.997**
Dataset #2-B				
C4.5	0.992	0.987	0.987	0.992
C4.5 - Bagging	**0.996**	**0.987**	**0.987**	**0.996**
C4.5 - Boosting	0.992	0.98	0.979	0.998

Tabla VI.2: Resultados en el dataset #2

raíz como páginas normales, es decir páginas no-Soft-404, que el dataset #2. Las páginas raíz tienden a ser más genéricas y difíciles de caracterizar, por lo que el sistema necesita un mayor número de páginas para entrenarse. Es por ello por lo que se ha obtenido una mayor precision en el dataset #2.

Analizando los resultados del dataset #2 se puede decir que los mejores se obtienen aplicando el algoritmo C4.5 y Boosting.

Los resultados mostrados en las tablas VI.1 y VI.2 indican que hay un pequeño número de páginas que no son detectadas. Como se ha dicho, esto es debido a que el servidor web no envía una típica página Soft-404 sino que envía el contenido de la página raíz del sitio web cuando se solicita una página que no existe.

En ambos datasets se observa que las heurísticas no dependen de la cantidad de páginas con la que se entrene el sistema. Para evaluar en más detalle la eficacia del sistema y comprobar que no depende del número de páginas con la que se entrene, se han realizado diversos experimentos sobre los datasets descritos anteriormente, pero usando diferentes tamaños del conjunto de entrenamiento. Concretamente se han realizado experimentos usando el 50% de los datasets como conjunto de entrenamiento y posteriormente se ha ido reduciendo dicho porcentaje al 25%, 15%, 10%, y 5%. Los resultados obtenidos se muestran en la Figura VI.3.

Los resultados obtenidos son muy similares a los discutidos anteriormente (tablas VI.1 y VI.2). Usando un 50% del total del dataset como conjunto de entrenamiento el sistema logra una precisión de 0.989 y un recall de 0.970. En el caso en el que el sistema entrena únicamente con el 5% del total de cada dataset, la precisión y el recall disminuyen muy poco, concretamente 0.062 y 0.075 respectivamente. Con estos resultados podemos confirmar el correcto funcionamiento del sistema para detectar páginas Soft-404 y su independencia de la cantidad de páginas con las que se entrene.

Training %	Dataset		Soft-404 Precision	Soft-404 Recall	Normal page Precision	Normal page Recall
50%	#1-A	C4.5 - Bagging	0.821	0.913	0.973	0.941
		C4.5 - Boosting	0.815	0.922	0.973	0.937
	#1-B	C4.5 - Bagging	0.926	0.978	0.977	0.924
		C4.5 - Boosting	0.924	0.980	0.980	0.921
	#2-A	C4.5 - Bagging	0.978	0.958	0.996	0.998
		C4.5 - Boosting	0.989	0.970	0.997	0.999
	#2-B	C4.5 - Bagging	0.994	0.984	0.984	0.994
		C4.5 - Boosting	0.995	0.988	0.988	0.995
25%	#1-A	C4.5 - Bagging	0.825	0.900	0.969	0.941
		C4.5 - Boosting	0.805	0.937	0.980	0.933
	#1-B	C4.5 - Bagging	0.926	0.971	0.970	0.924
		C4.5 - Boosting	0.924	0.974	0.973	0.921
	#2-A	C4.5 - Bagging	0.971	0.950	0.995	0.997
		C4.5 - Boosting	0.990	0.948	0.995	0.999
	#2-B	C4.5 - Bagging	0.993	0.980	0.980	0.993
		C4.5 - Boosting	0.993	0.985	0.985	0.993
15%	#1-A	C4.5 - Bagging	0.817	0.896	0.968	0.940
		C4.5 - Boosting	0.813	0.897	0.968	0.939
	#1-B	C4.5 - Bagging	0.925	0.965	0.964	0.924
		C4.5 - Boosting	0.921	0.967	0.965	0.919
	#2-A	C4.5 - Bagging	0.965	0.954	0.995	0.997
		C4.5 - Boosting	0.980	0.955	0.995	0.998
	#2-B	C4.5 - Bagging	0.993	0.975	0.975	0.993
		C4.5 - Boosting	0.993	0.980	0.980	0.993
10%	#1-A	C4.5 - Bagging	0.815	0.886	0.965	0.940
		C4.5 - Boosting	0.813	0.873	0.961	0.940
	#1-B	C4.5 - Bagging	0.926	0.959	0.958	0.925
		C4.5 - Boosting	0.919	0.962	0.961	0.917
	#2-A	C4.5 - Bagging	0.954	0.921	0.992	0.996
		C4.5 - Boosting	0.969	0.935	0.993	0.997
	#2-B	C4.5 - Bagging	0.989	0.970	0.970	0.989
		C4.5 - Boosting	0.991	0.978	0.978	0.991
5%	#1-A	C4.5 - Bagging	0.830	0.836	0.951	0.949
		C4.5 - Boosting	0.805	0.902	0.970	0.935
	#1-B	C4.5 - Bagging	0.918	0.948	0.947	0.917
		C4.5 - Boosting	0.912	0.956	0.954	0.909
	#2-A	C4.5 - Bagging	0.878	0.896	0.989	0.987
		C4.5 - Boosting	0.927	0.895	0.989	0.984
	#2-B	C4.5 - Bagging	0.985	0.966	0.966	0.985
		C4.5 - Boosting	0.988	0.971	0.971	0.988

Tabla VI.3: Resultados del sistema usando el 50%, 25%, 15%, 10% y 5% de cada dataset para entrenar el clasificador

VI.4.4 Comparación con otros Sistemas

En secciones previas se mostró el comportamiento de las heurísticas y del sistema propuesto aplicando el algoritmo C4.5 junto con Bagging y Boosting, demostrando el correcto funcionamiento del mismo. A continuación se van a comparar los resultados obtenidos por nuestro sistema con resultados que obtienen otros sistemas presentes en el estado del arte, concretamente con el algoritmo propuesto por Bar-Yossef *et al.* [BYBKT04] y el sistema de Lee *et al.* [LKK+09]. La comparación se realizó desde dos aproximaciones: teórica y práctica.

Los experimentos necesarios para la comparación teórica y empírica con el sistema de Bar-Yossef *et al.* [BYBKT04] han sido realizados en el dataset #2, ya que contienen páginas de la web Global y por lo tanto era más similar al usado por Bar-Yossef *et al.* [BYBKT04].

Para la comparación teórica se definió una cota superior de los tipos de páginas Soft-404 que cada sistema podía detectar. Bar-Yossef *et al.* explican en su artículo que no son capaces de detectar páginas Soft-404 en ciertos casos: a) páginas raíz y b) páginas web de parking.

Tras esta afirmación, un importante dato sería conocer el porcentaje de páginas raíz y parking que el sistema de Bar-Yossef *et al.* esta clasificando de forma erronea. Respecto a las páginas raíz no se ha podido obtener dicho dato. Para el estudio de las páginas de parking se seleccionó aleatoriamente un subconjunto del 20% del total de páginas Soft-404 y se identificó aquellas que eran de parking. El resultado fue que un 47.2% de las páginas eran de parking. Por lo tanto, sin tener en cuenta las páginas raíz, en el mejor de los casos el sistema propuesto por Bar-Yossef *et al.* [BYBKT04] detectará en el mejor de los casos el 52.8% de las páginas Soft-404. Por el contrario, el sistema propuesto, en base a los resultados obtenidos, detecta un 98% de las páginas Soft-404.

Para la comparación empírica, hemos implementado el algoritmo que explican los autores en su artículo. La Tabla VI.4 muestra los resultados obtenidos aplicando el algoritmo de Bar-Yossef *et al.* [BYBKT04] y los resultados obtenidos aplicando el sistema basado en las heurísticas comentadas.

Centrándonos en los resultados obtenidos para la detección de páginas Soft-404, observamos que en el dataset #2-A el sistema de Bar-Yossef *et al.* ha logrado una precisión y un recall de 0.787 y 0.945 respectivamente, mientras que nuestro sistema ha obtenido un 0.994 y 0.973, respectivamente. En el dataset #2-B, el sistema propuesto obtiene una precisión de 0.992 y un recall de 0.980, mientras que el sistema de Bar-Yossef *et al.* ha obtenido una precisión de 0.797 y un recall de 0.935. En resumen, el sistema propuesto mejora el recall obtenido por Bar-Yossef *et al.* para la detección de páginas Soft-404 en aproximadamente un 5% y un 25% la precisión. En el caso de páginas normales (no-Soft-404), nuestro sistema mejora la precisión y el recall, sobre el algoritmo de Bar-Yossef *et al.*, aproximadamente un 6% y un 32% respectivamente.

Analizando el algoritmo de Bar-Yossef *et al.* y los resultados que obtiene, se ha concluido que no detecta multitud de páginas Soft-404. Las desventajas de su algoritmo se identifican en los puntos siguientes:

	Soft-404		Página normal	
	Precision	Recall	Precision	Recall
Dataset #2-A				
Bar-Yossef *et al.*	0.787	0.945	0.914	0.768
Módulo Detección Soft-404	**0.994**	**0.973**	**0.997**	**0.997**
Dataset #2-B				
Bar-Yossef *et al.*	0.797	0.935	0.920	0.757
Módulo Detección Soft-404	**0.992**	**0.980**	**0.979**	**0.998**

Tabla VI.4: Resultados del algoritmo de Bar-Yossef *et al.* y del sistema propuesto en el dataset #2

- El sistema que presentan no funciona con servidores web que envían páginas Soft-404 sin realizar redirecciones HTTP. Es decir, el servidor web realiza una redirección interna y enviará un contenido diferente al esperado, pero sin enviar un error 404 u otro código HTTP de redirección.

- Hay multitud de casos en los que el número de redirecciones y/o las URLs a las que apuntan son diferentes entre la página web original y la página web generada de forma aleatoria. Concretamente, el 6.5% de las páginas Soft-404 no son detectadas debido a este hecho.

- Finalmente, otra desventaja importante es que la detección esta basada en la comparación de contenido. El proceso de comparación de contenido entre dos páginas web tiene un coste computacional alto y su resultado no es fiable. Los autores han considerado que una página es Soft-404 si el contenido de la página original y la aleatoria es idéntico o casi idéntico ("almost-identical content"). El problema radica en la interpretación numérica de la frase: "casi idénticos". En los experimentos llevados a cabo se ha considerado que dos páginas web son casi idénticas cuando el 80% de su contenido es igual.

  Para la comparación del contenido, el primer paso es extraer el contenido útil de cada página web (eliminar HTML tags, scripts, images, etc.). El proceso de extracción del contenido útil de una página web es un proceso complejo. Una de las aproximación existentes para la extracción de contenido útil es la de Donghua *et al.* [DP08]. Esta técnica fue la que incluimos en la implementación del algoritmo de Bar-Yossef *et al.* desarrollada para llevar a cabo los experimentos. Este estudio se basa en que la localización del contenido principal esta centralizado y tiene una estructura jerárquica. Para diferenciar aquellos nodos DOM [w3d11] con contenido útil de los demás nodos, los autores proponen un algoritmo que evalua el contenido de cada nodo en base a 4 parametros: densidad de enlaces en el texto, cantidad de enlaces, densidad de enlaces en cada nodo y longitud del texto de cada nodo. Donghua *et al.* marcan una serie de umbrales para decidir que nodo es diferente a los otros nodos del mismo nivel. Cuándo la densidad de enlaces en un nodo sea muy alta en relación al texto que contiene y a la cantidad de enlaces

	Soft-404	
	Precision	Recall
Dataset #2-A		
Lee	X	0.695
Módulo Detección Soft-404	0.994	0.973

Tabla VI.5: Resultados del sistema de Lee *et al.* y del Módulo Detección Web Spam en el dataset #2-A

totales de la página y de los demás nodos, ese nodo se considerará que no tiene contenido útil. En otro caso, el contenido del nodo será considerado útil.

Respecto a la comparativa del sistema propuesto con el sistema de Lee *et al.* [LKK+09] no hemos podido realizarla directamente debido a que: a) el sistema de Lee *et al.* [LKK+09] sigue una aproximación totalmente diferente a las nuestra (realiza un crawling completo de aquellos hosts que desea analizar), por lo que no puede ser aplicado sobre nuestro dataset y b) el dataset que usan no es público, por lo que no podemos aplicar nuestro método sobre él. Debido a esos dos problemas, se han comparado ambos sistemas analizando las desventajas existentes en el sistema de Lee *et al.*, y comparando los resultados de cada sistema en cada uno de sus correspondientes datasets. Las desventajas del sistema de Lee *et al.* son las siguientes:

- La detección de una página Soft-404 no se realiza en tiempo real, lo cual significa que muchas páginas web pueden ser procesadas y posteriormente el sistema detectará que son Soft-404. Esto es debido a que el sistema de Lee *et al.* realiza un proceso de crawling de los hosts que desea analizar como paso previo a decidir si se trata o no una página Soft-404.

- El sistema se basa en 3 simples heurísticas que pueden fallar en multitud de ocasiones. Por ejemplo, una página con gran cantidad de redirecciones, o un host con muchas páginas web y un pequeño número de redirecciones realizadas desde URLs del host.

- Como ocurre en el sistema propuesto por Bar-Yossef *et al.*, el sistema necesita redirecciones HTTP, en cualquier otro caso el sistema no detectará una página Soft-404

A diferencia del sistema de Lee *et al.*, el sistema propuesto:

- Detecta páginas Soft-404 en tiempo real pues no requiere un crawling previo, y no es necesario llevar a cabo un proceso de crawling de los hosts que se desean analizar. Por este motivo, previene la pérdida de recursos procesando e indexando páginas web que posteriormente pueden ser clasificadas como Soft-404.

- Está basado en un conocimiento aprendido y en un conjunto de heurísticas evaluadas previamente. Por otro lado, no se basa en el tipo de redirecciones realizadas en las páginas, por lo que detectará si una página es o no Soft-404 independientemente de las redirecciones realizadas.

La Tabla VI.5 muestra los resultados obtenidos por Lee *et al.* en su artículo [LKK+09]. Para comparar los resultados obtenidos por el Módulo Detección Web Spam y los de Lee *et al.* hemos utilizado el dataset #2-A ya que contiene páginas web de la Web global.

El sistema de Lee *et al.* obtiene un recall de 0.695 en la detección de páginas web Soft-404. En los resultados de nuestro sistema se observa que obtiene un recall de 0.973 y una precisión de 0.994. Esto supone una mejora del 40% en el recall para la detección de páginas Soft-404. La información sobre la precisión de detección de páginas Soft-404 no aparece indicada en el artículo de Lee *et al.* [LKK+09]. Sin embargo, los autores analizan la precisión de su sistema para detectar páginas con otros "soft errors" (5xx, 403, Spam, etc.). Considerando que dentro de ese conjunto de "soft errors" se incluyen páginas Soft-404, el sistema de Lee *et al.* obtiene una precisión de 0.866. Si lo comparamos la precisión del sistema propuesto, 0.994, se observa que se ha logrado una mejora del 14.78%.

VI.4.5 Eficiencia de las Técnicas Propuestas

Para verificar el coste computacional de las heurísticas propuestas, se analizó el tiempo de ejecución de cada una de ellas en el dataset #2.

La Figura VI.7 muestra un diagrama de caja y bigotes con los resultados de cada una de las heurísticas explicadas previamente.

Analizando los resultados, se observa que el tiempo de ejecución de las heurísticas "Title words", "Total images" y "Description words" es muy similar. Esto es debido a que la lógica y el código que ejecutan cada una de ellas es similar. Concretamente estas heurísticas tienen un tiempo máximo de ejecución de 2 milisegundos y una mediana próxima a 1 milisegundo. La heurística "Average length words" ha obtenido un tiempo de ejecución superior a las anteriores debido a que su complejidad computacional es mayor. En el caso de "Specific phrases" y "Ratio content/HTML", el tiempo de ejecución obtenido es mayor. Esto es debido a que el coste computacional de realizar búsquedas de palabras en el contenido, y separar el contenido útil del código HTML, es superior al de otras heurísticas comentadas. La heurística 'Specific phrases" se ejecuta en el peor de los casos en 5 milisegundos y tiene una mediana de 3 milisegundos. La heurística "Ratio content/HTML" consume aproximadamente 6 milisegundos y tiene una mediana de 3 milisegundos. Por último, comentar que la heurística que menos tiempo consume en su ejecución es "Total bytes", que se ejecuta como máximo en 1 milisegundo y tiene una mediana de 0.5 milisegundos.

Los resultados muestran que hay cierta distancia entre la mediana y el tiempo máximo de ejecución. Esto es debido a que el tiempo de ejecución de cada heurística depende del tamaño de la página a analizar y de la

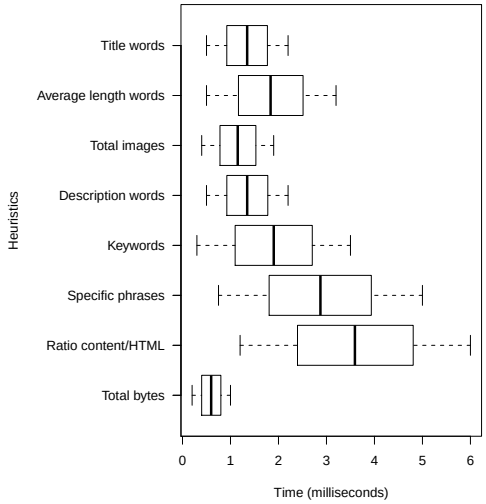


Figura VI.7: Tiempos de ejecución de las heurísticas en el dataset #2

complejidad del código HTML que contenga.

VI.4.6 Comparación de la Eficiencia con otros Sistemas

En esta sección se discuten las mejoras a nivel de rendimiento del sistema propuesto sobre los sistemas actuales. En este caso no se compararán los resultados obtenidos por nuestro sistema con los obtenidos por el sistema de Lee *et al.* [LKK+09], ya que sus técnicas utilizan una aproximación completamente diferente a la nuestra. El sistema de Lee *et al.* [LKK+09] necesita realizar un crawling completo de un host y de los host que están relacionados con él, como paso previo a decidir si es o no Soft-404. Esto provoca que el proceso de decisión sea mucho más lento que en nuestro sistema o que en el sistema propuesto por Bar-Yossef *et al.*

Es cierto que una vez el sistema propuesto por Lee *et al.* ha realizado el correspondiente crawling sobre los hosts a analizar, el proceso de clasificación para determinar que página es o no Soft-404 es más rápido que en el sistema propuesto. Sin embargo, dicho sistema provoca que el crawler procese, indexe y extraiga enlaces de páginas que en un futuro serán clasificadas como Soft-404. Este hecho causa una gran perdida de recursos.

Para analizar el algoritmo propuesto por Bar-Yossef *et al.*, se ha separado en cada uno de las procesos básicos que ejecuta, obteniendo para cada uno de ellos su tiempo de ejecución y su complejidad computacional. Para poder realizar la comparativa se ha realizado lo mismo con el sistema que se propone. En la Tabla

P.	Proceso	Complejidad	Tiempo (s)	Bar-Yossef et al.	Módulo Detección Web Spam
P1	Parsear URL	O(1)	10	Si	Si
	Resolución DNS			Si	Si
	HTTP Get			Si	Si
	Chequear código HTTP			Si	Si
	Seguir redirecciones			Si	Si
P1'	Mismos procesos que P1 para una página aleatoria	O(1)	10	Si	No
P2	Chequear Num. redirecciones	O(N)	t1	Si	No
	Chequear redirecciones			Si	No
P3	Comparar contenido	O(N^2)	t2	Si	No
P4	Ejecutar heurísticas	O(N)	0.0001	No	Si
P5	Clasificar página web	O(log N)	0.00018	No	Si

Tabla VI.6: Procesos, complejidad y tiempo de ejecución de cada uno de los sistemas de detección

VI.6, se muestran los resultados obtenidos. En la primera columna se indica el nombre de cada uno de los subprocesos usados por el algoritmo de Bar-Yossef *et al.* [BYBKT04] y por el sistema propuesto. Las siguientes columnas contienen una breve descripción de lo que realiza dicho proceso, su coste computacional, el tiempo de procesamiento y cual de los sistemas comparados es el que ejecuta dicho subproceso.

Analizando los resultados podemos ver que el sistema de Bar-Yossef *et al.* realiza siempre dos peticiones. La primera es para comprobar si es una página legitima y la segunda para realizar una petición a una página web aleatoria, para determinar si el servidor web devuelve un código HTTP 200 cuando recibe peticiones a páginas web desconocidas. Su sistema siempre necesita dos peticiones, podría realizar dos la primera vez que analiza un sitio web y almacenar el contenido de la página enviada por el servidor para futuras comprobaciones, sin embargo, hay que tener en cuenta que el contenido de la página puede variar, lo cual es común en páginas generadas dinámicamente. Para realizar la comparativa de los tiempo de ejecución se utilizó el tiempo usado por Bar-Yossef *et al.* [BYBKT04] en su artículo para las peticiones HTTP (proceso P1 y P1').

Por otro lado el sistema de Bar-Yossef *et al.* [BYBKT04] siempre realiza comprobaciones sobre el número de redirecciones y el lugar al que apunta cada una de ellas (proceso P2). En el caso de que su sistema no sea capaz de decidir, el sistema deberá realizar una comparación del contenido de ambas páginas web. El proceso P3 comprueba si el contenido es idéntico, pero también si es similar, mediante un proceso de "shingling" [BGMZ97b], lo cual tiene un alto coste computacional.

El sistema propuesto solamente realiza una petición, P1, y no realizará ninguna comprobación más si el servidor web devuelve un código HTTP 404. En los demás casos, el sistema ejecutará las heurísticas explicadas

	Complejidad		Tiempo de procesamiento (s)	
	Mejor caso	Pero caso	Mejor caso	Peor caso
Dataset #1-A				
Bar-Yossef *et al.*	O(1)	O(N^2)	20	20+t1[4]+t2[5]
Módulo Detección Web Spam	O(1)	O(N)	10.00028	10.00028

Tabla VI.7: Complejidad y tiempo de ejecución de los sistemas estudiados

previamente, P4, y ejecutará el proceso de clasificación, P5. No tiene que realizar ninguna comprobación más.

El tiempo de ejecución de los procesos P4 y P5 es el tiempo de ejecución del conjunto de las heurísticas y del tiempo consumido por el algoritmo de clasificación. Los resultados del análisis se muestran en la Tabla VI.7.

La complejidad de los procesos P1 y P1' es constante, es decir O(1). Sin embargo la complejidad del proceso P2 depende del número de redirecciones por lo que su complejidad es lineal, O(N), donde N es el número de redirecciones. La complejidad de P3 es O(N^2), donde N representa el número de caracteres de la página web, ya que el algoritmo propuesto por Bar-Yossef *et al.* compara el contenido de las páginas web. Por otro lado la complejidad del proceso P4 es también O(N), donde N representa el tamaño de la página web. Finalmente, el proceso P5 que recorre el árbol de decisión, tiene una complejidad de O(log N), ya que es un árbol binario donde N representa la profundidad del árbol. Sin embargo, en este caso se conoce la profundidad del árbol ya que el árbol de decisión es creado previamente durante el proceso de entrenamiento del algoritmo. Por ello, la complejidad del proceso P5 es O(1). En resumen, el sistema propuesto, en el mejor de los casos, tiene la misma complejidad computacional que el propuesto por Bar-Yossef *et al.*, O(1), pero en el peor de los casos tiene una complejidad de O(N), mientras que el algoritmo de Bar-Yossef *et al.* tiene una complejidad de O(N^2).

Con respecto al tiempo de análisis, independientemente de que no se incluyan los tiempo $t1$ y $t2$, ya que su artículo no los indicaba, vemos que el sistema propuesto es más rápido que el sistema de Bar-Yossef *et al.* [BYBKT04]. En el mejor caso, nuestro sistema reduce a la mitad el tiempo de ejecución (de 20 segundos a 10 segundos), y en el peor de los casos consume 10.00028 segundos mientras que el sistema de Bar-Yossef *et al.* consume $20 + t1 + t2$ segundos.

A modo de resumen en la Figura VI.8 se muestra un diagrama de secuencia de los procesos utilizados por ambos sistemas (incluyendo el tiempo y la complejidad computacional).

[4]Tiempo de procesamiento de un proceso con complejidad O(N)
[5]Tiempo de procesamiento de un proceso con complejidad O(N^2)

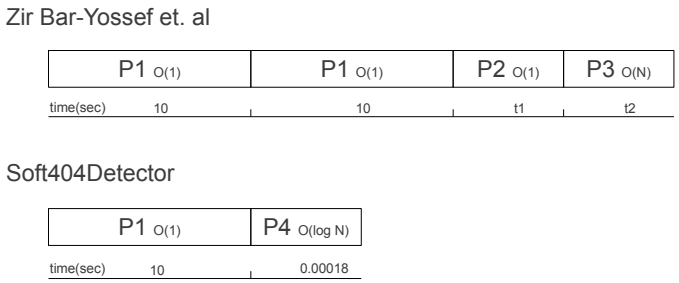


Figura VI.8: Secuencia de ejecución de cada uno de los sistemas

VI.5 Conclusiones y Comparación de Resultados

Se ha demostrado que en torno al 7.35% de los servidores devuelven páginas Soft-404 ante peticiones de páginas desconocidas. Es un resultado muy alto, superior a los datos de Web Spam en la Web, y que indica que dicho problema debe ser estudiado.

En el presente capítulo se ha presentado un módulo de detección de páginas Soft-404, que contiene un conjunto de novedosas heurísticas y un método que permite combinarlas maximizando los resultados obtenidos. Las heurísticas propuestas se basan en el análisis del contenido para lograr caracterizar las páginas Soft-404. Analizando el estado del arte no hemos encontrado estudios que analicen y traten esta problemática como una aproximación basada en el análisis del contenido.

Los resultados de las heurísticas estudiadas se han usado como entrada de varios algoritmos de clasificación para tratar de maximizar el éxito de la detección. Finalmente se han usado arboles de decisión, concretamente, el algoritmo C4.5 junto con técnicas de Bagging y Boosting.

Ya que no existía nada similar en el estado del arte, se han construido dos dataset de páginas Soft-404, los cuales nos han permitido probar nuestras heurísticas y obtener los resultados del clasificador. En el dataset de .es, hemos conseguido una precisión de 0.928 y un recall de 0.983 en páginas Soft-404 y una precisión de 0.983 y un recall de 0.952 en páginas normales. En el dataset global, hemos conseguido una precisión de 0.996 y un recall de 0.987 en páginas Soft-404 y una precisión de 0.987 y un recall de 0.996 en páginas normales.

Se han comparado los resultados obtenidos por el módulo de detección con los sistemas presentes en la literatura. Concretamente, se han realizado comparaciones con los sistemas presentados por Bar-Yossef *et al.* [BYBKT04], y Lee *et al.* [LKK+09]. Por un lado, se han analizado las ventajas del sistema propuesto sobre las limitaciones de los existentes, ya que existen clases de páginas Soft-404 que no detectan y múltiples escenarios donde los métodos existentes no tienen éxito. Por otro lado, se ha implementado el algoritmo propuesto por Bar-Yossef *et al.* y se ha probado sobre los datasets generados. Resumiendo, los resultados obtenidos demuestran

que el sistema propuesto logra mejorar el recall obtenido por Bar-Yossef *et al.* en aproximadamente un 5% y la precisión en torno a un 25%. En el caso de páginas normales (no-Soft-404) el sistema propuesto mejora la precisión y el recall en aproximadamente un 6% y un 32%, respectivamente.

Debido a que el sistema desarrollado por Lee *et al.* [LKK$^+$09], trata de detectar páginas Soft-404 desde una una aproximación diferente, se ha realizado una comparativa desde el punto de vista teórico. Se ha demostrado que el sistema propuesto detecta con mayor exactitud y en mayor número de casos páginas Soft-404 que el sistema de Lee *et al.* [LKK$^+$09].

Por último, destacar que también se ha realizado un análisis del rendimiento del sistema propuesto y de los sistemas presentados por Lee *et al.* y Bar-Yossef *et al.*. Tras el análisis, ha quedado patente que el módulo de detección presentado necesita menos de la mitad del tiempo que el sistema de Bar-Yossef *et al.* y que la complejidad del mismo es, en el peor de los casos, de O(N) frente a una complejidad de O(N^2) del sistema de Bar-Yossef *et al.*

VII

Módulo de Detección de Cambios en páginas web

Este capítulo analiza el módulo de detección de modificaciones en páginas web. El módulo está basado en un sistema que permite al crawler conocer las modificaciones realizadas por los usuarios en sus páginas en "tiempo real". De ahora en adelante nos referiremos a dicho sistema como sistema WCD, nombre que proviene de las siglas Detección de Cambio Web, del inglés Web Change Detection.

El capítulo se estructura de la siguiente manera. En primer lugar, en la sección VII.1 se discutirán brevemente las técnicas actuales para la detección de cambios en páginas web. La sección VII.2 muestra un amplio estudio sobre las políticas de recrawling utilizadas por los principales motores de búsqueda (Google, Yahoo! y Bing). En la sección VII.3 se explica la arquitectura del sistema. Se analizarán los componentes que la forman, su funcionamiento y su integración con un motor de búsqueda. La sección VII.4 describe y muestra los resultados obtenidos con el sistema propuesto. Se mostrarán resultados obtenidos en un sitio web propio, en un conjunto de dominios monitorizados y en una simulación sobre los cambios realizados en páginas web y los accesos de los usuarios a estas. También se analizarán los resultados de rendimiento del sistema. Finalmente, en la sección VII.5 se resumen los resultados obtenidos, comparándolos con los sistemas existentes, y se analizan las conclusiones obtenidas.

VII.1 Técnicas para la Detección de Cambios en páginas web

Los sistemas de crawling actuales utilizan diferentes mecanismos para tratar de detectar cambios en páginas y sitios web. Estos mecanismos han sido explicados en mayor detalle, y junto con los artículos relacionados en la sección II.4. A continuación recordamos cada uno de ellos:

- La cabecera Last-Modified del protocolo HTTP, que permite conocer la fecha de la última modificación de un recurso.
- Algunos servidores web proporcionan feeds RSS [1], para proporcionar información sobre los cambios realizados en las páginas web.
- Frecuencia de cambios de sus recursos a través de alguno de los ficheros usados en el protocolo de exclusión de robots ("robots.txt" y "sitemap.xml").
- Acuerdos entre sistemas de crawling importantes con ciertos sitios web para recibir notificaciones sobre las modificaciones realizadas.
- Detección de cambios basada en módelos estadísticos (frecuencia de cambio, longevidad de los contenidos, etc.).

[1] http://www.rssboard.org/rss-specification

VII.2 Motivación

En la sección anterior se han presentado brevemente los mecanismos y métodos existentes para la detección de cambios en páginas web. En la sección II.4, se explicarón la multitud de problemas e inconvenientes que estos sistemas tienen. Por ello, se propone un sistema que permita a los sistemas de crawling conocer cuándo ha cambiado una página web.

Como paso previo al diseño del sistema propuesto, hemos estudiado el comportamiento respecto al refresco de páginas, durante el proceso de recrawling, de los principales motores de búsqueda (Google, Yahoo! y Bing). Para ello, se ha llevado a cabo un experimento para medir cuándo una página web ha cambiado y cuándo ha sido detectado este cambio por los motores de búsqueda.

Para el experimento, se han seleccionado aleatoriamente un conjunto de 150 dominios del sitio web SeeTheStats[2]. Este sitio web proporciona información sobre un conjunto de URL de diferentes dominios entre la que se incluyen ciertos datos de Google Analytics, como son los accesos de los usuarios. Únicamente se ha considerado para el estudio la página principal de cada dominio. Con el fin de no sesgar el experimento, se han seleccionado dos tipo de sitios web: sitios web con bajo PageRank (entre 0 y 2), y sitios web con alto PageRank (entre 3 y 5). En total se han seleccionado 25 sitios web de cada tipo de PageRank. No se han considerado sitios web con PageRank más alto, ya que no fueron aleatoriamente seleccionados en nuestro dataset (de hecho, sitios web con PageRank mayor de 5 no están disponibles en el dataset). Se ha distinguido entre sitios web con PageRank alto y bajo, porque se han considerado que la relevancia de una página web, es decir su PageRank, podría ser uno de los factores usados en las políticas de recrawling de los motores de búsqueda (es decir, los sitios web más relevantes estarán más actualizados).

Para determinar cuándo un motor de búsqueda ha detectado un cambio, se ha obtenido la versión en caché de cada página web para cada uno de los motores de búsqueda. La fecha de caché proporciona la información de último acceso a esa página web. Cada página web ha sido monitorizada durante 30 días (desde el 30 de Abril hasta el 30 de Mayo de 2012). Debido a las limitaciones de los motores de búsqueda, cada página web sólo ha podido ser monitorizada cada 12 horas.

Al mismo tiempo, para detectar cuándo una página web ha cambiado, cada página web fue monitorizada por un crawler desarrollado por nosotros para tal fin, cada 12 horas durante el mismo periodo de tiempo. La principal dificultad de este experimento es decidir cuándo la página web ha sido modificada. Para ello, se ha extraído y comparado el contenido útil de cada versión de la página web. Para el proceso de extracción del contenido útil, se ha usado la aproximación propuesta por Donghua *et al.* en [DP08]. En esta aproximación se describe el proceso de cómo obtener información útil de una página web, es decir sin etiquetas HTML, enlaces, imágenes, etc.

[2] `http://www.seethestats.com/`

Para realizar la comparación entre el contenido útil de cada versión de la página web, y poder decidir si la página web ha sido modificada, se ha seguido la aproximación propuesta por Manku *et al.* [MJDS07]. Para ello, se ha usado la herramienta *shash* que permite detectar documentos casi idénticos [3]. Esta herramienta divide el documento en n tokens, le asigna un peso a cada uno de ellos y calcula su correspondiente hash. Finalmente, con el peso y el hash de cada token, se crea la firma del contenido útil de la página web. En trabajos previos, Manku *et al.* consideraban que un par de documentos eran casi idénticos, sí y sólo sí, sus firmas difieren menos de 3 bits. Para este estudio, se ha considerado que dos versiones de una página web son casi idénticas si sus firmas difieren menos de 6 bits. De esta forma, los cambios mínimos realizados en una página web son descartados, y solamente son considerados aquellos cambios significativos. Esto beneficiará a los motores de búsqueda en la comparación, ya que solamente se les requerirá que detecten aquellos cambios significativos.

Con estos dos tipos de información, se ha podido medir:

- Cuánto tiempo los motores de búsqueda tienen sus datos desactualizados.
- El número de cambios no detectados por los motores de búsqueda. Esto ocurre cuando una página web ha sido modificada dos o más veces entre dos visitas consecutivas del crawler.
- El número de veces que un crawler procesa una página de forma innecesaria, por no haber sido modificado desde su último procesamiento.

La Tabla VII.1 muestra una visión general de los resultados. Por un lado, se presenta el tiempo medio transcurrido entre dos cambios consecutivos. Como se esperaba, el PageRank incrementa la frecuencia de cambio: una página con PageRank 5 es modificada, de media, dos veces más que una página web con PageRank 0. Es decir, cuánto más relevante es un sitio web, más dinámico será, modificando su contenido con mayor frecuencia. De media, un sitio web con bajo PageRank cambiará al menos 11 veces al mes, mientras que un sitio web con alto PageRank lo hará cerca de 17 veces.

Por otro lado, el tiempo medio transcurrido entre dos visitas de un buscador a una página web, sigue el mismo patrón: páginas con mayor PageRank son visitadas más frecuentemente. Los tres motores de búsqueda revisitan una página web con alto PageRank, de media, cada dos días y medio. En las páginas con bajo PageRank, Bing, Google y Yahoo!, regresan a una página, de media, cada 4 días.

En términos generales, los motores de búsqueda siguen una aproximación conservadora: revisitarán una página web cuando es altamente probable que ésta haya cambiado. Como consecuencia, durante algún tiempo, sus índices estarán desactualizados y pueden perder ciertas modificaciones.

La Tabla VII.2 muestra los detalles de los resultados obtenidos, agrupados por PageRank. Los resultados representan los valores medios para los 150 dominios estudiados. La fila *Tiempo Desactualizado* hace referencia

[3] `http://d3s.mff.cuni.cz/~holub/sw/shash/`

	PageRank Bajo				PageRank Alto			
	PR 0	PR 1	PR 2	**PR Bajo**	PR 3	PR 4	PR 5	**PR Alto**
Cambio	76.02h	66.02h	60.01h	**67.35h**	51.83h	41.33h	36.83h	**43.33h**
Google	114.17h	111.71h	98.74h	**108.10h**	78.47h	57.07h	65.82h	**67.12h**
Yahoo!	121.24h	96.80h	86.21h	**101.42h**	84.42h	66.23h	59.64h	**70.10h**
Bing	108.34h	96.79h	92.33h	**99.16h**	84.42h	63.97h	60.24h	**69.55h**

Tabla VII.1: Frecuencia de cambios en páginas web y accesos de los motores de búsqueda. La unidad de tiempo utilizada son horas

al porcentaje de tiempo que los motores de búsqueda tienen sus índices desactualizados, es decir, una página web ha cambiado, pero los motores de búsqueda no lo han detectado todavía. *Refrescos Innecesarios* muestra el número de visitas realizadas por los crawlers cuando una página web todavía no había cambiado, y *Cambios no Detectados* cuenta el número de veces que una página web ha cambiado entre dos accesos consecutivos del motor de búsqueda.

En el caso de Google, se observa que para sitios web con bajo PageRank, su caché e índices están desactualizados el 61.31% del tiempo. Para los sitios web con alto PageRank, este porcentaje se disminuye hasta el 53.00% del tiempo, aproximadamente 8 puntos menos. Por otro lado, se observa que el número de accesos innecesarios hechos por Google, es inferior a la unidad. Esto implica que durante el mes estudiado, sólo algunos de los sitios fueron visitados más de una vez de forma innecesaria. Sin embargo, Google pierde una importante cantidad de modificaciones: 6.91 para sitios web de bajo PageRank y 7.5 para alto PageRank. Obsérvese también que, aunque el número de modificaciones no detectadas es mayor en los sitios web más relevantes, ya que estos cambian más frecuentemente, el porcentaje de cambios no detectados es menor en los sitios web de alto PageRank (45% de los cambios no son detectados) que en los sitios web de bajo PageRank, donde cerca del 65% de los cambios no son detectados.

En resumen, se observa que Google utiliza una aproximación conservadora, de modo que vuelve a visitar las páginas web sólo cuando está bastante seguro de que dicha página web puede haber cambiado. Google se centra principalmente en páginas web con alto PageRank, visitándolas con más frecuencia, y por lo tanto, detectando más cambios. Sin embargo, sus índices están desactualizados más de la mitad del tiempo para ambos tipos de sitios web (alto y bajo PageRank), y muchas de las modificaciones no son detectadas (siendo peor para los sitios web con bajo PageRank).

Los resultados obtenidos por Yahoo! indican que sus repositorios están desactualizados un 49.92% del tiempo para los sitios web con bajo PageRank, y un 58.92% para sitios web con alto PageRank. Respecto a los refrescos innecesarios, los resultados son ligeramente peores que los de Google, y para los cambios no

	Google								
	PR 0	PR 1	PR 2	**PR Bajo**	PR 3	PR 4	PR 5	**PR Alto**	**Resultado Medio**
Tiempo Desactualizado (%)	50.19	69.20	64.54	**61.31**	51.39	38.08	69.51	**53.00**	**57.16**
Refrescos Innecesarios	1	0.5	0.63	**0.71**	0.18	0.61	1.125	**0.64**	**0.67**
Cambios no Detectados	7.69	7.42	5.63	**6.91**	9.45	6.30	6.75	**7.50**	**7.20**
	Yahoo!								
	PR 0	PR 1	PR 2	**PR Bajo**	PR 3	PR 4	PR 5	**PR Alto**	**Resultado Medio**
Tiempo Desactualizado (%)	59.48	46.62	43.67	**49.92**	62.89	60.24	53.61	**58.92**	**54.42**
Refrescos Innecesarios	1	0.1	1.27	**0.79**	1.54	1.23	2.5	**1.75**	**1.27**
Cambios no Detectados	6.92	5.25	5.27	**5.81**	8.18	5.69	6.625	**6.83**	**6.32**
	Bing								
	PR 0	PR 1	PR 2	**PR Bajo**	PR 3	PR 4	PR 5	**PR Alto**	**Resultado Medio**
Tiempo Desactualizado (%)	53.87	46.62	42.52	**47.67**	62.89	54.80	55.15	**57.61**	**52.64**
Refrescos Innecesarios	1	0.3	1.09	**0.79**	1.27	1.15	2.37	**1.60**	**1.19**
Cambios no Detectados	6.92	4.84	3.45	**5.07**	9.36	5.23	4.87	**6.49**	**5.78**

Tabla VII.2: Detalles de los resultados de Google, Yahoo! y Bing

detectados, los resultados son mejores.

Yahoo! tiene sus repositorios desactualizados, de media, la mitad del tiempo, pero, a nivel global obtiene una ligera mejora sobre los resultados de Google (54.42% frente 57.16%). Sin embargo, Yahoo! está menos compensado y obtiene mejores resultados para los sitios web con bajo PageRank. En Yahoo! también se observa la misma política conservadora, que en Google.

Los resultados de Bing y Yahoo! son bastante similares, como era previsible por el acuerdo comercial entre ambos [sea12]. Bing obtiene el menor tiempo medio de desactualización (52.64%), con resultados similares para los sitios web con bajo PageRank (similarmente a Yahoo!). Los refrescos innecesarios y los cambios no detectados son similares a los obtenidos por Yahoo! pero con ligeras mejoras.

En resumen, se ha observado, como se esperaba, que los principales motores de búsqueda visitan más frecuentemente los sitios web con PageRank más alto (perdiendo menos cambios). Por otro lado, se ha observado el uso de una política conservadora en los tres motores de búsqueda estudiados, prefiriendo mantener sus repositorios desactualizados que consumir recursos en volver a visitar páginas que no han sido modificadas.

Finalmente, la principal diferencia entre los tres motores de búsqueda es que Yahoo! y Bing mantienen sus repositorios más actualizados para páginas web con bajo PageRank, mientras que Google mantiene más actualizado los repositorios corrrespondientes a sitios web con alto PageRank. El comportamiento de Yahoo! y Bing puede parecer ineficiente, ya que la lógica indica que sitios web con mayor PageRank contienen contenidos de mayor calidad y relevancia, y por ello deben ser refrescados con mayor frecuencia para no perder modificaciones. Sin embargo, su comportamiento esta relacionado con los resultados mostrados en el

estudio de Arasu *et al.* [ACGM+01]. En este trabajo, el autor propone que cuando cierta página cambia con demasiada frecuencia, si no es posible mantenerla actualizada debido a que los recursos son limitados, es mejor centrar el uso de nuestro recursos en páginas web que es posible mantener actualizadas.

En esta sección se ha analizado el tiempo de desactualización de los repositorios de los principales motores de búsqueda (Google, Yahoo! y Bing). En términos generales, los tres motores de búsqueda tienen un comportamiento similar, con diferencias menores. Sin embargo, queda patente que el modelo que utilizan produce importantes desajustes en los contenidos indexados por ellos, lo cual puede tener un importante impacto en la experiencia del usuario final. En la siguiente sección, se presenta el sistema WCD que trata de solucionar este problema.

VII.3 Arquitectura del Sistema

El objetivo del sistema de Detección de Cambios Web (WCD, Web Change Detection) es permitir que los motores de búsqueda actualicen el contenido de un sitio web contenidos en "tiempo real". Cuando una página web utiliza el sistema WCD se dice que, dicha página está siendo monitorizada. Desde ese momento, el sistema detectará los cambios realizados en la página y los notificará al motor de búsqueda.

En esta sección, se explicará y discutirá la arquitectura del sistema. Primero, se realizará una descripción a alto nivel, y posteriormente se explicarán en detalle los componentes de la arquitectura y su funcionamiento. Tras esto, se analizarán la integración entre el sistema y el motor de búsqueda. Finalmente, se discutirán las ventajas y desventajas del sistema propuesto.

VII.3.1 Arquitectura del sistema WCD

El sistema propuesto está basado en la arquitectura distribuida cliente/servidor de la Word Wide Web e incorpora un servidor WCD y un agente WCD, para crear una arquitectura colaborativa y así poder detectar en "tiempo real" los cambios realizados en páginas web. La arquitectura del sistema se muestra en la Figura VII.1, y se compone de los siguientes elementos:

- Cliente web (Web Client).
- Servidor web (Web Server).
- Agente WCD (WCD Agent).
- Servidor WCD (WCD Server).

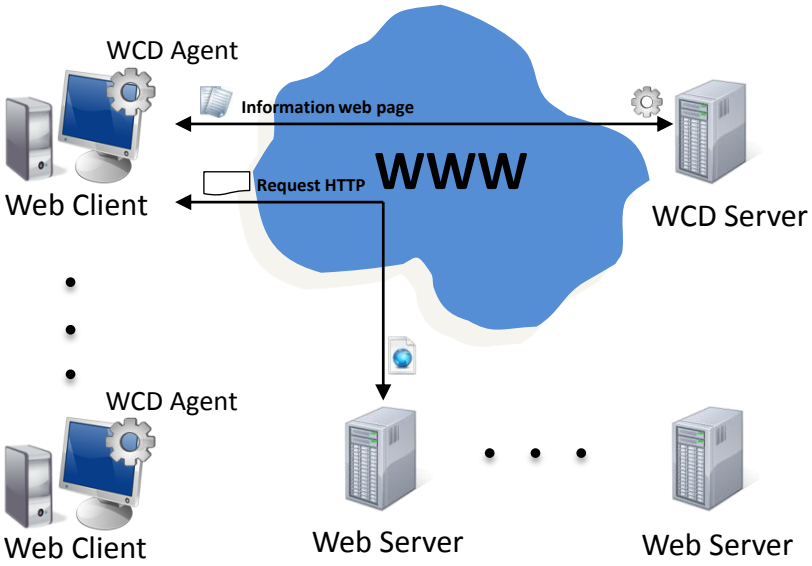


Figura VII.1: Arquitectura colaborativa del sistema WCD

El cliente web es un navegador web, y es el responsable de cargar la página web, solicitar el agente WCD al servidor WCD y ejecutarlo. El servidor web es un servidor web común, que almacena una o más páginas web monitorizadas. Cada página web monitorizada incluye una referencia al agente WCD. Cuando un cliente web realiza una petición a una página web, el servidor le responde enviándole el contenido.

El agente WCD es una aplicación basada en el navegador web, que envía información sobre las modificaciones realizadas en la página web al servidor WCD. El servidor WCD tiene dos tareas principales. Por un lado almacenar y enviar agentes WCD a los clientes web. Y por otro, almacenar y actualizar la información sobre las páginas web monitorizadas.

Servidor WCD

En la Figura VII.2 se muestra el servidor WCD y los dos subsistemas por los que está formado:

- El Gestor de Agentes (Agent Manager): es el responsable de gestionar los agentes WCD, recibir la petición del agente realizada por el navegador web, y decidir (en base a diferentes parámetros) cúal será el agente que se envíe.
- El Detector de Cambios WCD (WCD Changes Detector): es el responsable de recibir la información de

la página web enviada por el agente WCD, y de detectar las modificaciones y actualizar la información de la página web (por ejemplo, la fecha de última modificación) almacenada en el repositorio.

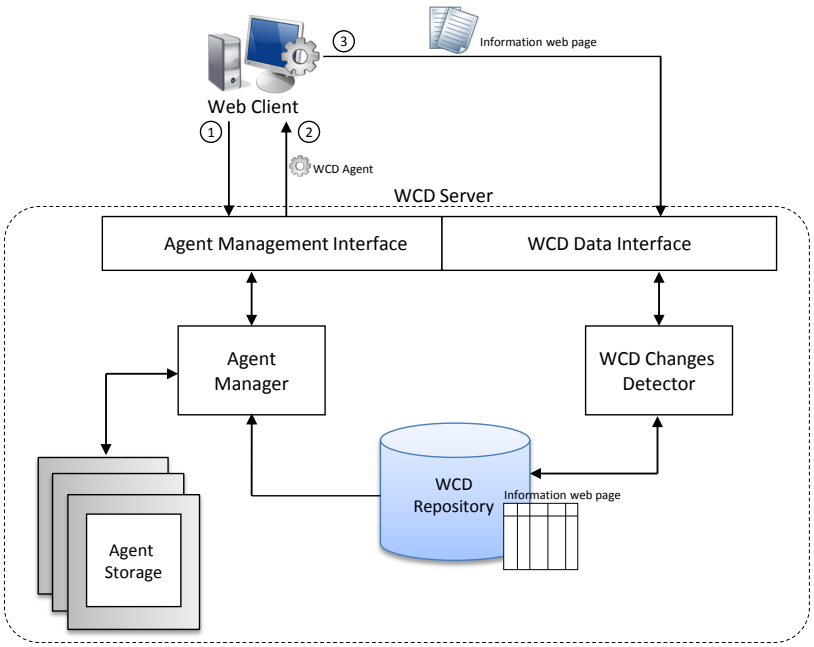


Figura VII.2: Componentes del servidor WCD

El repositorio WCD almacena toda la información sobre las páginas web monitorizadas. Para cada página web, se almacena la siguiente información:

- Resumen de la página web: realizado por el agente WCD.
- Última fecha de modificación.
- Última fecha de acceso.
- Ratio de cambio: definido como el número de cambios detectados por unidad de tiempo (por ejemplo en horas).
- Importancia de la página web: por ejemplo, usando PageRank.

- Tipo de página web, dependiendo de su frecuencia de cambio. Se han considerado los siguientes tipos: un periódico, un foro, una página de opiniones, un blog y una página web de una compañía.

El servidor proporciona dos interfaces: por un lado la interfaz de Control de Agentes, que será la encargada de gestionar las peticiones de los agentes realizadas desde los navegadores, y la interfaz de Datos WCD, que será la responsable de recibir la información de las páginas web enviada por los agentes.

Las peticiones recibidas por la interfaz de Control de Agentes son procesadas por el Gestor de Agentes. El Gestor de Agentes podrá usar la información almacenadá en el repositorio WCD (por ejemplo, la fecha de ultimo acceso) y alguna información sobre el cliente (por ejemplo, el tipo y versión del navegador) para decidir qué agente debe ser enviado en cada caso.

La información recibida a través de la interfaz de Datos WCD es procesada por el Detector de Cambios WCD. Este módulo comparará el resumen de la página web enviado por el agente con el almacenado en el repositorio, para determinar si se ha producido algún cambio. La información del repositorio será actualizada en caso de que se haya detectado algún cambio; almacenando la última fecha de acceso y los campos restantes.

Agente WCD

El agente WCD es un programa en JavaScript que se descarga del servidor WCD y se ejecuta en el navegador web del cliente. Este programa realizará el resumen de la página web y enviará la información al servidor WCD.

Inicialmente, se han considerado solo dos tipos de agentes: el agente Digester y el agente Void. Sin embargo, la arquitectura propuesta está diseñada para operar con cualquier número de agentes.

El agente Digester calculará el resumen de la página, realizando el hash MD5 del contenido útil. El contenido útil hace referencia al contenido principal, dónde se encuentra la información real de la página, sin etiquetas HTML, enlaces, imágenes u otro tipo de información de formato. Se ha seguido la aproximación de Donghua Pan [DP08] (explicada en el capítulo III), para extraer el contenido útil de las siguientes partes de la página web: título, cabecera, cuerpo y pie de página. Tras haber extraído el contenido útil, el agente calcula el hash MD5 de cada parte (esto constituye el resumen de la página web), y envía la siguiente información al servidor WCD: el resumen, el URL y la fecha actual.

El agente WCD envía los datos de la página web a través de una petición AJAX. De esta forma el usuario web no observa retardos de ningún tipo, y por lo tanto su experiencia en el uso de la correspondiente página web, no se ve afectada.

El agente Void no realizará ningún tipo de resumen, y por lo tanto no enviará ninguna notificación al servidor WCD. Este agente es útil cuando el servidor WCD considera que la información sobre la página web es reciente, por ejemplo, si el último acceso se ha realizado hace escasos segundos.

El servidor WCD enviará uno u otro agente, dependiendo de una serie de factores: locales (como el tiempo transcurrido desde la última actualización) o remotos (como la versión del navegador web utilizada por el cliente). Inicialmente, se han considerado factores locales, y más específicamente, el tiempo transcurrido desde la última modificación. Se ha definido t_p, como el tiempo transcurrido desde el último envío de información de una página p y un umbral γt. Cuando el servidor WCD recibe una petición solicitando un agente, desde la página web p, si t_p es mayor o igual a γt, el servidor WCD le enviará el agente Digester, y en cualquier otro caso se le enviará el agente Void. En el caso de que sea una página web nueva la que realice la petición del agente, el servidor WCD enviará siempre el agente Digester.

La siguiente sección explica en más detalle la interación entre el servidor web, el navegador, el servidor WCD y el agente WCD.

VII.3.2 Funcionamiento del sistema WCD

En esta sección se describirá el funcionamiento del sistema WCD. Es importante destacar que el sistema WCD no requiere ningún cambio, add-on o extensión en el servidor web o en el cliente web. El funcionamiento del sistema está basado en el agente y servidor WCD.

Desde el punto de vista del administrador de la web, el funcionamiento del sistema es sencillo. Una vez el administrador decide incluir la página en el sistema WCD, debe insertar una referencia al agente WCD en su página web. En la práctica, esto consiste en añadir un sencillo código JavaScript que invoque el agente.

Una vez hecho esto, éste es el funcionamiento usual del sistema WCD (los diferentes pasos pueden verse en la Figura VII.2):

1. En primer lugar, el usuario realiza una petición desde el navegador al correspondiente servidor web. El servidor web le responde, enviándole el contenido de la página web, que incluye una referencia al agente WCD.

2. Entonces, el navegador web solicita el agente WCD mediante una petición al servidor WCD, para así lograr completar el contenido de la página web. El Gestor de Agentes consulta el repositorio para conocer la fecha de la última actualización de esa página web. Dependiendo del tiempo transcurrido desde dicha fecha, el Gestor de Agentes enviará el agente Digester o el agente Void.

3. Finalmente, el navegador web ejecutará el agente WCD. En caso de que el agente enviado sea el Digester, éste generará el resumen de la página y lo enviará al servidor WCD. Entonces, el Detector de Cambios WCD comparará la información recibida con la información almacenada en el repositorio, y la actualizará si detecta que el contenido almacenado es diferente al actual, es decir si el resumen (hash) de cada versión es diferente.

VII.3.3 Integración del sistema WCD en un Sistema de Crawling

Los motores de búsqueda actuales utilizan crawlers que recorren la Web de forma continuada, buscando nuevos contenidos y páginas web que han sido modificadas. El crawler usa múltiples tipos de procesos de crawling que realizan diferentes tipos de recorridos de la Web, mejorando así su rendimiento global. Algunos de dichos procesos, procesan la Web periódicamente, es decir, comienzan un nuevo proceso de crawling cuando han terminado uno. Otro grupo de procesos de crawling, se centran en ciertos sitios o páginas web, basándose en políticas de re-crawling. Entre dichas políticas, se pueden destacar las siguientes: a) relevancia de la página, b) frecuencia de cambios en las páginas del sitio web, y c) tipo de página web.

Para detectar cambios en páginas web, los crawlers actuales utilizan un método pull. Basándose en diferentes datos estadísticos (relevancia y ratio de cambios), el crawler decide cuándo es probable que una página haya sufrido un cambio y por lo tanto es el momento adecuado de volver a procesar dicha página. Un crawler que tenga integrado el sistema WCD, conocerá casi en tiempo real cuándo una página web ha cambiado y podrá decidir el momento exacto para volver a procesarla.

La integración del sistema WCD con un sistema de crawling se realiza a través de la interfaz de Acceso WCD (Figura VII.3). Esta interfaz proporciona operaciones de gestión y acceso a la información almacenada en el repositorio WCD al sistema de crawling, y viceversa, el sistema WCD puede notificar al crawler los cambios realizados en las páginas web.

La comunicación entre el crawler y el sistema WCD puede realizarse en 3 modos: push, pull o la combinación de ambos, de acuerdo a las necesidades del correspondiente sistema de crawling. En el modo pull, el crawler solicita al sistema WCD el conjunto de páginas web que han sido modificadas. En el modo push, el sistema WCD notifica al crawler las páginas que se han modificado. Ya que ambos modos son complementarios, otra posibilidad es utilizarlos de forma combinada.

Para el uso del sistema WCD en un sistema de crawling, se deben tener en cuenta un conjunto de cuestiones referentes a la seguridad y a la escalabilidad del sistema. A continuación, se discuten en detalle cada una de dichas cuestiones.

En ciertas situaciones, un usuario podría modificar la información enviada al sistema WCD por el agente, para notificar que una página web ha cambiado, cuando en realidad no lo ha hecho. Sin embargo, éste no es un problema importante, ya que esto únicamente implica que la página sea reprocesada sin necesidad, pero su contenido nunca será indexado, ya que el módulo detector de documentos conocidos del crawler se percatará que dicho contenido ya ha sido procesado. Para controlar este tipo de situaciones el sistema WCD utiliza un nuevo módulo denominado Controlador de Trafico Web. Este módulo es el responsable de permitir o denegar el acceso al sistema WCD. Para ello, el Controlador de Trafico Web mantiene un índice con las IPs y los URLs que deben ser filtrados por cuestiones de seguridad. Esta información se almacena en el repositorio WCD. Cuando un crawler reprocesa una página web, comprueba si la página ha cambiado realmente. En el caso de que la

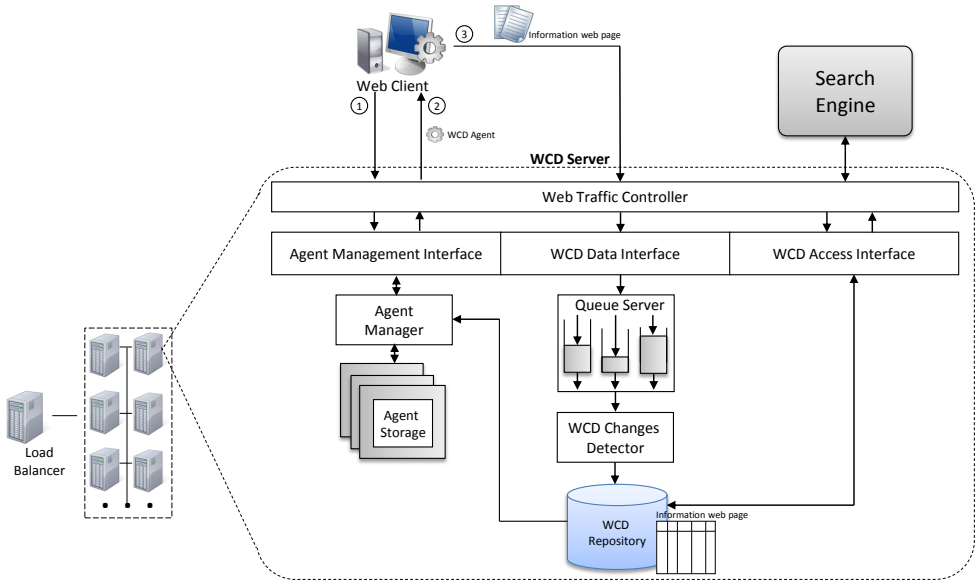


Figura VII.3: Arquitectura completa del sistema WCD

modificación no haya ocurrido, el crawler lo notificará al sistema. De esta forma el Controlador de Trafico Web podrá conocer qué IPs han manipulado los datos, y por lo tanto deben ser filtradas. El Controlador de Trafico Web también mantiene una lista con los accesos recientes, con la finalidad de detectar ataques DDoS, y denegar accesos desde aquellas IPs que aparezcan en dicha lista negra.

Con respecto a la escalabilidad, el sistema debe ser capaz de resistir picos esporádicos de carga, causados por un uso intensivo del sistema o por ataques DDoS. Para ello, en el momento en que el sistema detecta dichas situaciones, pone en marcha el protocolo que se describe a continuación (se ha diferenciado entre la interfaz de Gestión WCD y la interfaz de Datos WCD):

- Interfaz de Gestión WCD: en este escenario el sistema siempre enviará el agente Digester, evitando consultar el repositorio WCD. Esto se realiza para no perder notificaciones de cambios en páginas web, y evitar acceder a la base de datos para comparar cúal es la fecha del último cambio.

  En el caso en que la carga del sistema siga aumentando, el sistema enviará el agente Void. De esta forma es posible que se pierdan notificaciones de cambios, pero se evita que el agente envíe más datos al sistema y por tanto que su carga aumente.

- Interfaz de Datos WCD: esta interfaz recibe la información sobre las modificaciones realizadas en una

página web y las envía al Detector de Cambios WCD por medio del servidor de Colas. El servidor de Colas es el responsable de almacenar la información en la correspondiente cola para su posterior procesamiento. De esta forma, en situaciones de alta carga, el sistema puede devolver al cliente un código de éxito (200) de forma instantánea y realizar el procesamiento de la información cuando la carga del sistema haya disminuido. En ese momento, el sistema recorrerá las colas del servidor, procesando únicamente la información más reciente de cada página web, y eliminando el resto de entradas.

Además de los elementos descritos previamente, y debido al alto número de peticiones que el sistema recibe, se ha considerado la posibilidad de distribuir el sistema. Para ello, se ha creado un nuevo elemento, llamado Balanceador de Carga, que es el responsable de redirigir cada nueva petición al proceso con menor carga. De esta forma, el sistema balanceará la carga, evitando malgastar recursos.

La Figura VII.3 muestra la arquitectura completa del sistema WCD. El Balanceador de Carga, el servidor de Colas y el Controlador de Trafico Web, reducen la carga del sistema, logrando procesar todas las peticiones que llegan al sistema, y ayudando a prevenir ataques DDoS y de modificación de datos.

VII.3.4 Ventajas y Desventajas del sistema WCD

En esta sección se discuten el conjunto de ventajas y desventajas de usar el sistema WCD en un motor de búsqueda. Entre las mejoras logradas, destacan las siguientes:

- Detección de cambios en tiempo real: el sistema WCD permite conocer casi en tiempo real cuándo una página web ha sido modificada, al contrario que los motores de búsqueda actuales, que presuponen cuando una página web ha podido cambiar en base a información estadística.
- Procesamiento distribuido: parte del procesamiento para decidir si la página web ha sido modificada, es decir, el resumen de la página web, se realiza en el lado cliente, lo cual libera al sistema de crawling de este procesamiento y le permite ahorrar recursos.
- Mantenimiento mínimo: desde el punto de vista del creador del contenido, únicamente tendrá que insertar una referencia al agente WCD desde la página a ser monitorizada. El sistema propuesto, no requiere acceso al directorio raíz del sitio web o un conocimiento previo sobre la frecuencia de cambio de cada página web, tal como se necesita para utilizar los ficheros “robots.txt” y “sitemap.xml” [SZG07], y lo que es más importante, no se depende de la implementación de la cabecera Last-Modified del servidor web.

Todas las ventajas descritas, por un lado, mejoran la calidad del contenido indexado, ya que será más actual, y por otro lado, reducen en gran medida el uso de recursos en el proceso de crawling.

Sin embargo, el sistema propuesto tiene una serie de desventajas:

- Notificación falsa de páginas web modificadas, es decir, que un cliente envíe datos modificados, haciéndole creer el servidor WCD que dicha página ha sufrido un cambio.
- Escenarios con alto número de peticiones en un corto período de tiempo, lo cual podría provocar que no todas las peticiones fueran atendidas en tiempo real. Estas situaciones pueden deberse a un uso intensivo del sistema, o por ataques DDoS.

Aunque estos inconvenientes existen, se han propuesto una serie de elementos y protocolos que mitigan el riesgo de que ocurran y que minimicen sus consecuencias (sección VII.3.3).

VII.4 Evaluación del Método Propuesto

Esta sección describe y discute los experimentos realizados para verificar la efectividad y eficiencia del sistema WCD. Para la realización de estos experimentos, se ha implementado un prototipo del sistema propuesto.

Se han llevado a cabo dos tipos de experimentos:

- Experimentos de detección de cambios: se ha analizado la capacidad del sistema WCD para detectar cambios en páginas web, y se han comparado los resultados obtenidos con los de los principales motores de búsqueda. Se han considerado dos opciones diferentes: el mejor y el peor escenario posible.
- Experimentos de rendimiento: se ha estudiado el rendimiento del sistema propuesto con el fin de comprobar si puede ser usado en entornos reales con un gran volumen de datos y de accesos de usuarios. Para ello, se han llevado a cabo tests de carga sobre el sistema.

VII.4.1 Experimentos de Detección de Cambios

Resultados en el Mejor Escenario

Una de las asunciones en las que se basa el sistema propuesto, es que, cuando un administrador modifica una página web, él visitará inmediatamente la página web para comprobar que se visualiza correctamente. Esto constituye nuestro mejor escenario.

Para poder evaluar el comportamiento del sistema en dicho escenario, se ha creado un sitio web propio: un blog. Durante 30 días (desde el 11 de Mayo hasta el 11 de Junio de 2012), se ha añadido, cada 12 horas, una entrada en el blog (es decir, el contenido del sitio web ha cambiado cada 12 horas). El objetivo de este experimento fue comparar los resultados obtenidos por los crawler de los principales motores de búsqueda (Google, Yahoo! y Bing) con los resultados obtenidos por el sistema WCD.

	Google	WCD
Tiempo Desactualizado (%)	61.13	0.00
Refrescos Innecesarios	3	0.00
Cambios no Detectados	5	0.00

Tabla VII.3: Resultados del sistema en el mejor escenario

El blog fue creado en Blogspot `http://webchangedetector.blogspot.com.es/`, por diversas razones:

- Es posible modificar la página web principal e incluir una referencia al agente WCD.
- Es posible tener información totalmente fiable de cuándo han sido realizados los cambios.
- Se pueden conocer los accesos de los usuarios y de los sistemas de crawling. Esto es crucial para determinar el tiempo que los motores de búsqueda tienen el sitio web desactualizado.

Para acelerar el proceso de visita e indexación del sitio web por parte de los motores de búsqueda, se registró el sitio web en los tres motores de búsqueda analizados. Sin embargo, los resultados de Yahoo! y Bing, no son mostrados ya que durante el més en que se realiza el experimento, estos motores de búsqueda no habían indexado el sitio web. La Tabla VII.3 muestra los resultados obtenidos por Google y el sistema WCD.

Antes de nada, se observa que los resultados obtenidos por Google son consistes con los obtenidos en el experimento global (ver sección VII.2), aunque empeora ligeramente el tiempo de desactualización y los refrescos innecesarios. Concretamente, los resultados muestran que Google tiene el blog desactualizado un 61.30% del tiempo. El número de accesos innecesarios es ligeramente superior que en sus resultados a nivel global, probablemente porque el crawler está todavía estudiando la frecuencia de cambio del sitio web. Por otro lado, Google ha perdido 5 modificaciones, lo cual representa una gran mejora con respecto a los resultados obtenidos previamente. Esto ocurre, probablemente, porque los cambios realizados en este experimentos ocurren de forma regular (cada 12 horas), mientras que en los resultados analizados sobre otros sitios web, los cambios se realizaban de forma aleatoria.

Con respecto al sistema WCD, todos los cambios realizados son detectados en tiempo real. Tras cada nueva entrada, el administrador del blog recargaba la página (con el agente WCD), y el agente WCD notificaba las modificaciones al sistema. Un motor de búsqueda que utilizase el sistema propuesto podría lograr una actualización perfecta, con sus índices siempre actualizados, sin ninguna pérdida de cambios (0 cambios no detectados), y sin realizar ningún procesamiento innecesario de la página web (0 refrescos innecesarios).

	PR 0	PR 1	PR 2	**Bajo PR**	PR 3	PR 4	PR 5	**Alto PR**
Accesos de Usuarios	51.10	60.85	89.39	**67.11**	571.43	1332.84	733.85	**879.37**

Tabla VII.4: Número medio de accesos de usuario por día para los sitios web monitorizados

Resultados en el Peor Escenario

En esta sección se analiza el comportamiento del sistema propuesto en el peor escenario posible. En este caso se asume que se realiza un cambio en un sitio web, pero que el administrador no regresa a dicha página para comprobar el cambio realizado. Por lo tanto, el sistema depende de los accesos de los usuarios al sitio web. El objetivo es medir cuándo detecta el sistema propuesto los cambios realizados y compararlo con los resultados obtenidos por los principales motores de búsqueda.

Idealmente, para evaluar el comportamiento en este escenario, se necesitarían un conjunto de páginas web con diferentes PageRanks, que estuvieran utilizando el sistema WCD y que estuvieran indexadas en los principales motores de búsqueda. Al mismo tiempo, se necesitaría los logs de accesos de los usuarios y todos los cambios realizados en cada página web. Desafortunadamente, esto no es posible con los medios disponibles, por lo que, se decidió realizar una simulación para observar el comportamiento del sistema propuesto.

En la sección VII.2, se presentan los resultados de detección de Google, Yahoo! y Bing para 150 sitios web. Esto constituye nuestro punto de partida.

Para simular el comportamiento del sistema WCD es necesario conocer dos variables:

- Los cambios en las páginas web.
- Distribución de accesos de los usuarios.

Con respecto a los cambios realizados en páginas web, se ha estudiado el tiempo transcurrido entre dos cambios consecutivos en los 150 sitios web monitorizados. La hipótesis de partida fue que los cambios podían seguir una distribución de Poisson. Se realizó un test de Kolmogorov-Smirnov para cada página web, y el resultado indica que los cambios ocurridos en cada una de los 150 sitios web se ajustan a una distribución de Poisson con un p-valor inferior a 0.005. La media para la distribución de Poisson fue estimada a partir de ratio de cambio medio obtenido para cada tipo de PageRank (ver Tabla VII.1).

La última variable necesaria, los accesos de usuario, fueron también simulados usando una distribución de Poisson. Hemos seleccionado esta distribución en base a los trabajos presentados por Mikael Andersson *et al.* [ABHN05] y Özsu [GÖ03], que muestran que los accesos de usuario, bajo diferentes circunstancias, siguen una distribución de Poisson.

	PR 0	PR 1	PR 2	**Bajo PR**	PR 3	PR 4	PR 5	**Alto PR**
Google	38.15	45.69	38.73	**40.75**	26.64	15.75	28.99	**23.79**
Yahoo!	45.22	30.75	26.2	**34.07**	32.59	24.90	22.81	**26.77**
Bing	32.32	30.77	32.82	**31.81**	32.59	22.64	23.41	**26.22**
WCD	0.22	0.15	0.25	**0.21**	0.04	0.02	0.04	**0.03**

Tabla VII.5: Tiempo medio de detección de cambios para los motores de búsqueda y para el sistema WCD. Tiempo mostrado en horas

Para el calculo del número de accesos medios por día, se han extraído las estadísticas, del sitio web SeeTheStats, para cada sitio web monitorizado. La Tabla VII.4 muestra los resultados obtenidos. Estos valores serán usados como una estimación de la media para cada distribución de Poisson.

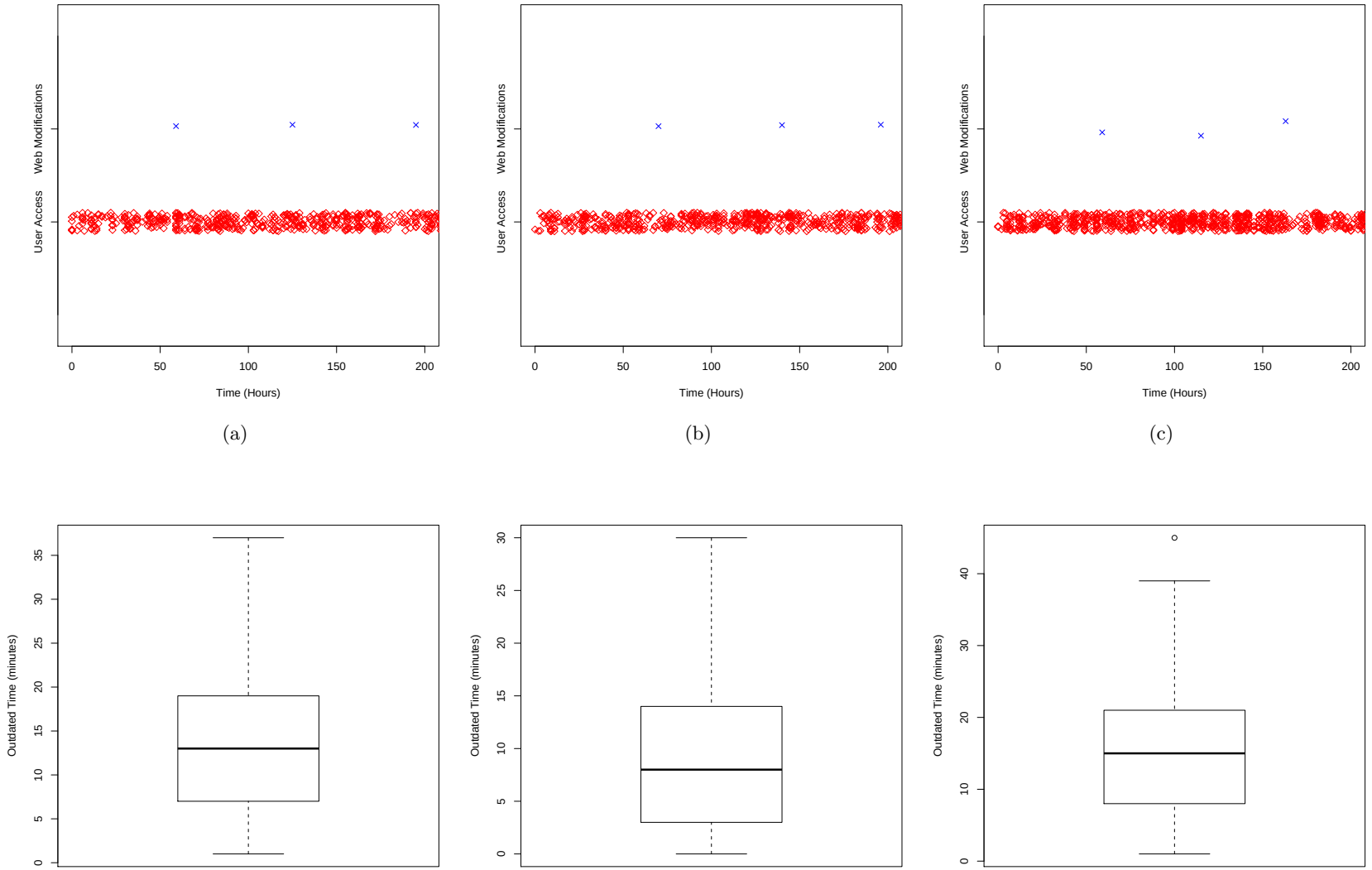
Usando estas dos variables, se ha simulado el comportamiento del sistema WCD para páginas web con PageRanks de entre 0 y 5, durante 200 horas (aproximadamente una semana). En cada caso, se realizaron cinco simulaciones diferentes, mostrándose los valores medios correspondientes.

La Figura VII.4 y la Figura VII.5 muestran: a) los accesos de usuario de acuerdo a su correspondiente PageRank (cuadrados) y las cambios realizados (cruces), y b) un diagrama de caja y bigotes para el tiempo en que el sistema propuesto tiene la información desactualizada (es decir, tiempo que tarda en detectar el cambio), en base a la simulación realizada. Concretamente, la Figura VII.4 incluye los resultados para dominios con PageRank entre 0 y 2, y la Figura VII.5 para los dominios con PageRank entre 3 y 5.

De acuerdo a lo esperado, en aquellas páginas con bajo PageRank el sistema propuesto requiere más tiempo para detectar cambios. Por ejemplo, para sitios web de PageRank 0 el sistema necesita 15.08 minutos de media para detectar un cambio, mientras que en sitios web con PageRank de 5, este tiempo es inferior a un minuto. El tiempo máximo con PageRank 0 es de 37 minutos, mientras que con PageRank 5 es de solamente 7 minutos. Por otro lado, el peor resultado se obtiene para PageRank 2, en donde el sistema tarda 45 minutos en detectar un cambio.

Para poder comparar los resultados obtenidos por el sistema propuesto y los obtenidos en el estudio de los motores de búsqueda, se ha calculado el tiempo que un crawler necesita para detectar una modificación web. Los resultados han sido extraídos utilizando los datos mostrados en la sección VII.2. La Tabla VII.5 muestra los resultados correspondientes.

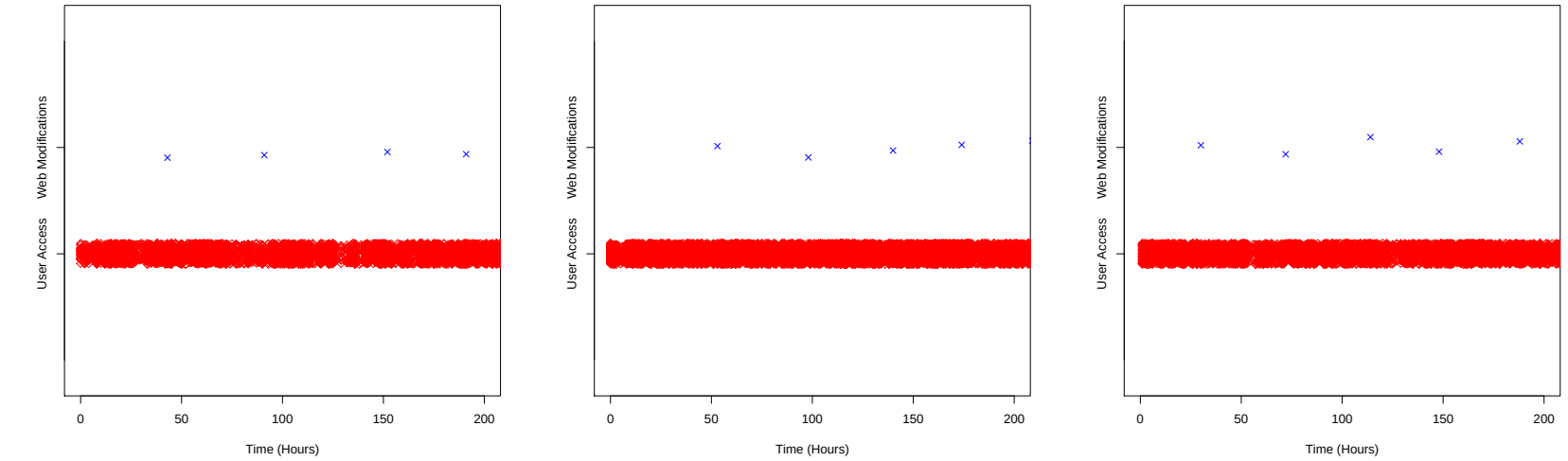
Analizando los resultados obtenidos en páginas web con bajo PageRank, los motores de búsqueda tardan un día y un cuarto (Google incluso más de un día y medio), en detectar una modificación. Sin embargo, el sistema WCD, de media necesita solamente 12 minutos (esto es dos ordenes de magnitud menor). Los resultados de sitios web con alto PageRank, muestran que Google, Yahoo! y Bing tardan aproximadamente un día en detectar



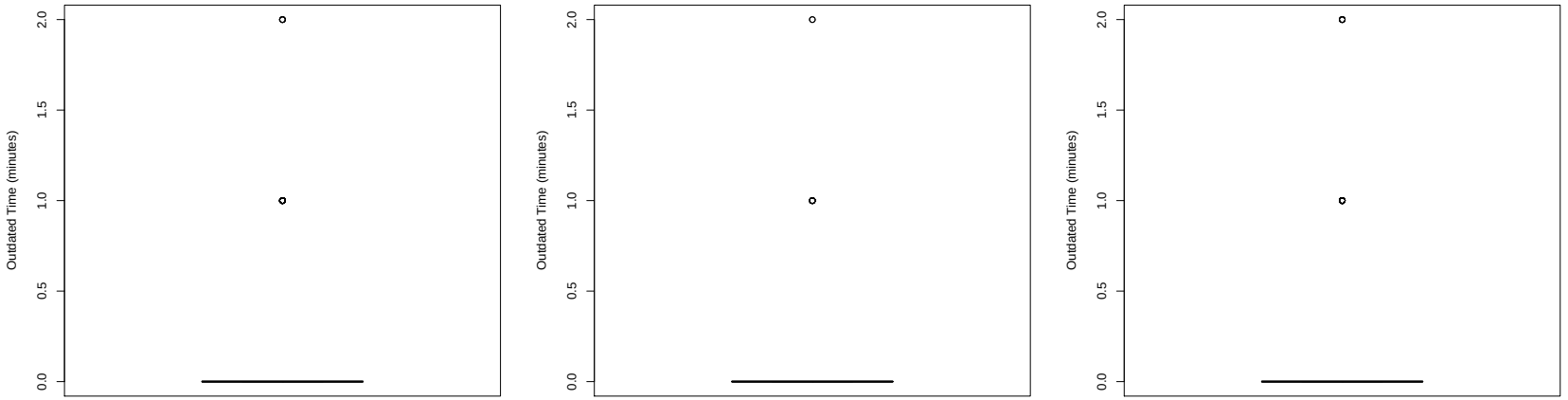
(a) (b) (c)

(d) (e) (f)

Figura VII.4: Resultados del sistema WCD para sitios web con PageRank entre 0 y 2



(a) (b) (c)



(d) (e) (f)

Figura VII.5: Resultados del sistema WCD para sitios web con PageRank entre 3 y 5

un cambio. En este caso, Google logra ser un poco más rápido que los otros buscadores, como se esperaba tras los resultados obtenidos en la sección VII.2. Nuevamente, el sistema propuesto es capaz de mejorar en gran medida estos resultados. En este caso, el tiempo medio que necesita el sistema WCD, es de aproximadamente un minuto, es decir hasta 3 órdenes de magnitud inferior.

Estos resultados muestran que el modelo pull usado por los actuales motores de búsqueda proporciona una baja efectividad para la detección de cambios, básicamente porque el crawler debe estimar, en base a datos estadísticos, cuándo se ha realizado un cambio en una página web. Al contrario, el sistema WCD, está basado en un modelo push, apoyándose en los accesos de los usuarios para la detección colaborativa y para notificar los cambios a los motores de búsqueda (aunque, como se ha explicado previamente, dicha notificación puede realizarse mediante un módelo push, pull o la combinación de ambos). Los resultados que obtiene el sistema, incluso en el peor de los escenarios, muestran que es capaz de detectar, casi en tiempo real, los cambios realizados en páginas web.

VII.4.2 Resultados de Rendimiento

En esta sección se analiza el sistema propuesto desde el punto de vista del rendimiento, tratando de probar que es posible su utilización en entornos reales. Concretamente, este experimento consiste en realizar un test de carga sobre el sistema, para conocer cómo puede escalar de acuerdo al número de peticiones recibidas.

Las pruebas realizadas se han centrado en los siguientes casos de uso del sistema:

- Caso de uso a): peticiones del agente WCD desde el navegador del cliente. La respuesta del sistema WCD depende del tiempo transcurrido desde la última actualización. Para ello, el sistema consulta el repositorio del sistema y decide si debe enviar un agente Void o un agente Digester. En el caso en que el sistema envíe un agente Void, el navegador no volverá a enviar una petición posterior.

- Caso de uso b): el agente WCD tiene que enviar una nueva petición al sistema con la información sobre las modificaciones detectadas. En este caso, el sistema WCD tiene que añadir o actualizar el contenido de su repositorio.

Para los dos casos de uso presentados existen tres escenarios de acuerdo con la probabilidad de que ocurran diferentes acciones en cada uno de ellos. Concretamente, se han realizado experimentos asumiendo tres posibles casos para cada uno de los dos casos de uso: que uno de los escenarios ocurra el 100% de las veces o que ambos tienen el 50% de posibilidades de ocurrir.

También se han realizado experimentos para cada escenario usando diferente número de peticiones en periodos de 100 segundos, concretamente: 1000, 5000 y 10000 peticiones.

	Caso de uso a)			Caso de uso b)		
	100% Nueva URL - 0% URL Conocida			100% Inserciones - 0% Actualizaciones		
N. Peticiones	1,000	5,000	10,000	1,000	5,000	10,000
Rendimiento	599/min	1,855/min	1,952/min	599/min	1,781/min	1,582/min
Error %	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Servidor CPU	26.1%	84.4%	85.5%	26.8%	76.6%	85.6%
Servidor RAM	75.0%	93.4%	96.5%	74.3%	73.2%	78.4%
	0% Nueva URL - 100% URL Conocida			0% Inserciones - 100% Actualizaciones		
Rendimiento	598/min	1,912/min	1,998/min	599/min	1,830/min	2,008/min
Error %	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Servidor CPU	27.9%	82.0%	83.6%	25.84%	78.8%	86.2%
Servidor RAM	75.1%	93.5%	95.5%	50.5%	63.1%	64.9%
	50% Nueva URL - 50% URL Conocida			50% Inserciones - 50% Actualizaciones		
Rendimiento	598/min	1,836/min	2,057/min	804/min	1,746/min	2,045/min
Error %	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Servidor CPU	28.3%	84.9%	86.5%	26.1%	76.9%	85.6%
Servidor RAM	72.7%	91.2%	94.3%	64.8%	69.2%	72.9%

Tabla VII.6: Resultados de los tests de carga para los casos de uso a) y b)

Los resultados se han obtenido desplegando el sistema en un PC Intel Core 2 DUO 6300 a 1.86GHz y 1GB de RAM.

Es necesario explicar que en ocasiones el rendimiento mostrado no es acorde con el porcentaje de error (por ejemplo cuando el error es del 0%). Este hecho ocurre porque, aunque las peticiones han sido completadas exitosamente, es decir, no han causado error, el rendimiento (peticiones/unidad de tiempo) disminuye debido a otras razones como son: los problemas de red, las colas TCP/IP, la carga del servidor, etc.

Resultados de los Tests de Carga

Esta sección muestra y discute los resultados obtenidos en las pruebas de carga realizadas sobre el sistema. La Tabla VII.6 muestra los resultados para los diferentes escenarios considerados en términos de ratio de peticiones realizadas, ratio de error, y carga del servidor medida en porcentaje de CPU y RAM usados.

Caso de uso a): el primer hecho que se observa es que ninguno de los diferentes tests realizados ha causado un error en el cliente. El rendimiento de las peticiones en el lado cliente es estable para los tres escenarios (100% de páginas nuevas, 50% de cada tipo y 0% de páginas nuevas). En el test con 1000 peticiones, se observa que en los diferentes escenarios el rendimiento se mantiene próximo a las 598 peticiones por minuto. En los casos de 5000 y 10000 peticiones, se observa que el rendimiento es similar.

Analizando los resultados obtenidos en el servidor, se observa que con 1000 peticiones el uso de CPU es del 27.43% y el uso de RAM es del 74.26%. En el caso de 5000 peticiones el uso de CPU y RAM se incrementan

hasta el 83.76% y el 92.7%, respectivamente. Por último, en el caso de 10000 peticiones, el uso de CPU se incrementa 2 puntos más hasta el 85.31%, y el uso de RAM 3 puntos, hasta el 95.43%.

Basándonos en los resultados obtenidos, aunque los diferentes escenarios obtienen diferentes rendimientos y usos de CPU y RAM, no se ha observado que ninguno de ellos pueda provocar una caída de rendimiento (en el cliente o en el lado servidor), que pueda significar un cuello de botella para el sistema.

Caso de uso b): de la misma forma que el caso de uso a), ningúno de los diferentes escenarios ha causado un error en el lado cliente. En el escenario en que el 100% de las páginas web sean nuevas, provoca inserciones continuas en la base de datos del repositorio, lo que hace que el rendimiento sea menor con respecto a los escenarios donde las inserciones son del 0% o del 50%. Para el 100% de inserciones con 1000 peticiones, se observa que el rendimiento es de 599 peticiones por minuto, con 5000 y 10000 peticiones, el rendimiento es de 1781 y 1582 peticiones por minuto. Concretamente, para el escenario con 50% y 50% de actualizaciones, con 1000 peticiones el rendimiento disminuye hasta las 804 peticiones completadas por minuto. En los casos de 5000 y 10000 peticiones, el rendimiento aumenta hasta 1746 y 2945 peticiones por minuto, respectivamente.

Finalmente, analizando los resultados de este caso de uso en el servidor, se observa que, independientemente, de la caída de rendimiento que provocan las inserciones, el uso de CPU y RAM es similar en todos los escenarios. Con 1000 peticiones el uso de CPU y de RAM es del 26.24% y del 63.20%, respectivamente. En los casos de 5000 y 10000 peticiones, el uso de CPU aumentó hasta el 77.43% y el 85.80%, respectivamente. Por último, el uso medio de RAM es de 76.3% con 5000 peticiones y del 72% con 10000 peticiones.

En resumen, se concluye que el sistema es estable en todas las pruebas de carga realizadas, manteniendo el rendimiento en el lado cliente, y el uso de CPU y RAM estables en el lado servidor. Como podría esperarse, el incremento del número de peticiones causa un incremento de uso de CPU y memoria. Sin embargo, se observa que a pesar de duplicar el número de peticiones (de 5000 a 10000), el número de recursos utilizados no se duplica.

VII.5 Conclusiones

En este capítulo se ha presentado un novedoso sistema que forma parte de la arquitectura de crawling propuesta y que indicará cuándo una página web ha sido modificada. Actualmente los motores de búsqueda están basados en un modelo pull, el cual proporciona pobres resultados en la detección de cambios. En los experimentos llevados a cabo, se observa que los principales motores de búsqueda (Google, Yahoo! y Bing) tienen sus índices desactualizados más de la mitad del tiempo.

Por otro lado, se ha presentado el sistema WCD, que está basado en el modelo push, y que es capaz de detectar casi en tiempo real cambios en páginas web. El sistema propuesto se apoya en una arquitectura colaborativa y distribuida, compuesta por: servidor web, navegador web, agente WCD y servidor WCD. Para

cada página monitorizada, debe incluirse una referencia al agente WCD. Cada vez que un usuario visita una página web, el agente es ejecutado y éste notificará al sistema si la página ha sido modificada.

La principal idea que está detrás del sistema es que, cuando un administrador modifica una página, éste regresa a la página web para comprobar los cambios realizados, y que dicha página se visualiza de forma correcta. Esto constituye el mejor escenario posible, y nuestros experimentos muestran que, en este caso, el sistema WCD detecta inmediatamente los cambios realizados, mientras los actuales motores de búsqueda (por ejemplo Google) presentan al usuario un contenido desactualizado en el 61% de las ocasiones. Además, nuestro sistema no pierde modificaciones y tampoco recursos visitando una página web que no ha sido modificada.

En el escenario del peor caso, se asume que la página web podría haber sido modificada sin que el administrador la vuelva a visitar para visualizar los cambios. Por ello, el sistema depende de los accesos de los usuarios. Incluso en este caso, el sistema propuesto mejora los resultados obtenidos por los principales motores de búsqueda. En páginas web con PageRank bajo, los motores de búsqueda estudiados necesitan, de media, un día y cuarto para detectar un cambio, mientras que el sistema WCD solamente 12 minutos. En las páginas con PageRank alto, el rendimiento del sistema es incluso mejor. Los motores de búsqueda actuales necesitan aproximadamente un día para detectar un cambio, mientras que el sistema propuesto detecta un cambio en menos de un minuto (3 ordenes de magnitud menos).

También se han realizado experimentos sobre el rendimiento y escalabilidad del sistema, usando tests de carga. Los resultados muestran que una única máquina puede procesar 10000 peticiones en 100 segundos, sin producir ningún error en el lado cliente y con un razonable uso de CPU y memoria.

En resumen, se ha demostrado que los motores de búsqueda actuales tiene sus índices desactualizados más de la mitad del tiempo, debido al modelo de recrawling que usan. Por otro lado, se ha presentado el sistema WCD, basado en el modelo push, que permite detectar cambios en páginas web mucho más rápido que los modelos actuales.

VIII

Conclusiones y Trabajo Futuro

Este capítulo explica las conclusiones obtenidas tras la realización de esta tésis doctoral, y las posibles líneas de trabajo futuro. Concretamente, en la sección VIII.1 se evalúa el cumplimiento de los objetivos iniciales. La sección VIII.2 discute las contribuciones presentadas en la tesis y las conclusiones de este trabajo. Por último, en la sección VIII.3 se explican las líneas de trabajo futuro del autor.

VIII.1 Evaluación de Cumplimiento de Objetivos

Esta sección evalúa el cumplimiento de los objetivos de esta tesis doctoral. Dichos objetivos fueron detalladamente explicados en la sección I.2.

A continuación se presenta un breve resumen de cada uno de los objetivos marcados al inicio del presente trabajo, justificando de qué forma se ha cumplido cada uno de ellos.

1. *Proponer una arquitectura eficiente para sistemas de crawling de la Web.*

   Se ha creado una arquitectura de crawling que permite mejorar la calidad y la cantidad de las páginas web procesadas. La arquitectura se presenta en el capítulo IV.

   La arquitectura se basa en las arquitecturas de crawling existentes y se extiende con un conjunto de módulos que abordan los principales desafíos de la Web. La arquitectura, por un lado, evita el procesamiento de páginas y sitios web de Spam y Soft-404. De esta forma, conseguirá mejorar la calidad de los recursos web procesados y aumentar el número de recursos libres, lo cual permitirá procesar un mayor número de páginas web.

   Por otro lado, la arquitectura también incluye un módulo que le permite mejorar el proceso de actualización de los recursos indexados. Este módulo está basado en un novedoso sistema que permite detectar los cambios realizados en páginas web casi en tiempo real. De esta forma, nuevamente, el crawler logrará mejorar la calidad de los recursos procesados, ya que serán más actuales, y utilizará un menor número de recursos en el proceso de actualización de los repositorios.

   Los módulos citados mejoran las arquitecturas de crawling actuales, permitiendo realizar un crawling eficiente de la Web.

2. *Realizar una serie de estudios que ayuden a conocer las características de la Web, su evolución y las diferencias entre los contenidos de la Web de los diferentes países. Este conjunto de estudios ayudará a identificar los problemas presentes en la Web para su procesamiento por los sistemas de crawling.*

   Se han realizado dos estudios, presentados en el capítulo III, secciones III.1 y III.2. El primero de ellos estudia la Web Oculta del lado cliente. Se han identificado las tecnologías web más extendidas y los diferentes métodos de generación de enlaces utilizados por los creadores de sitios web. Tras esto se ha

creado una escala y un conjunto de métodos de evaluación sobre ella, que permiten medir la capacidad de los sistemas de crawling actuales para el tratamiento de la Web Oculta del lado cliente.

El segundo estudio, analiza y compara, durante 3 años consecutivos, las características generales de la Web Global y Española. Este estudio se ha centrado principalmente en aquellas características de la Web que afectan a su procesamiento desde el punto de vista de los sistemas de crawling. Entre otras características se han estudiado: la similitud de contenidos, la Web Oculta del lado cliente, la edad de las páginas, el crecimiento de la Web, las tecnologías web del lado cliente y del lado servidor, etc. Tras su análisis, se han propuesto un conjunto de medidas, a adoptar por los crawlers, para mejorar su rendimiento.

Los resultados obtenidos de ambos estudios han permitido detectar desafíos en el tratamiento de la Web, y han sido la base de los demás estudios realizados, que han permitido diseñar la nueva arquitectura de crawling propuesta.

3. *Proponer nuevas técnicas y algoritmos para permitir a los sistemas actuales de crawling detectar páginas web de Spam.*

   En el capítulo V, se han presentado un conjunto de novedosas heurísticas para la detección de Web Spam. Dichas heurísticas basan su detección en el análisis del contenido, tratando de detectar la mayor parte de los tipos de Web Spam y las diversas combinaciones de técnicas que existen.

   Las técnicas presentadas mejoran las presentes en el estado del arte, tanto en eficacia como en eficiencia. La detección de Web Spam se basa en un método propuesto que permite combinar el conjunto de heurísticas creadas. Este método, junto con las heurísticas, formarán parte de un nuevo módulo de la arquitectura de crawling propuesta, que se encargará de detectar páginas de Spam. El módulo propuesto analiza las páginas web tras ser descargadas. De esta forma, logrará mejorar los contenidos procesados y evitará la pérdida de recursos que conllevaría ser detectada a posteriori.

4. *Proponer nuevas técnicas y algoritmos para permitir a los sistemas actuales de crawling detectar páginas Soft-404.*

   Se han propuesto y evaluado un conjunto de técnicas que permiten la detección de páginas Soft-404. El análisis de estas técnicas se realiza en el capítulo VI.

   Del mismo modo que las páginas de Spam, la detección se realizará en base al análisis del contenido. El conjunto de heurísticas propuesta mejoran los sistemas de detección de páginas Soft-404 actuales, tanto en eficacia como en eficiencia. Dichas heurísticas junto con el método propuesto que las combina, forman parte de un nuevo módulo de la arquitectura de crawling propuesta. Este módulo se encargará de evitar el procesamiento de páginas Soft-404.

5. *Proponer un sistema que permita conocer las modificaciones realizadas en páginas y sitios web.*

Se ha creado y probado un sistema que permite, por un lado, a los propietarios de páginas web que sus contenidos sean actualizados en "tiempo real" por los motores de búsqueda, y por otro, a los sistemas de crawling mejorar su proceso de refresco de contenidos, ya que conocerán en todo momento qué páginas han sido modificadas. En el capítulo VII se describen todos sus componentes y funcionamiento.

El sistema se basa en una arquitectura colaborativa entre el lado cliente y lado servidor, por lo que es poco intrusiva y altamente escalable. El crawler evitará realizar accesos a recursos web que no han sido modificados y no perderá modificaciones en el tiempo transcurrido entre dos procesamientos. Por otro lado, permitirá mejorar la calidad de los contenidos indexados, ya que serán más actuales, lo cual repercute en la calidad de las búsquedas y en la experiencia del usuario web.

6. *Validar la eficacia y eficiencia de las técnicas y métodos propuestos con experimentos sobre páginas y sitios web reales, demostrando que es posible la construcción de sistemas de crawling capaces de detectar y evitar páginas y sitios web de Spam y Soft-404, así como de realizar un re-crawling eficiente de los contenidos previamente indexados.*

   Los experimentos realizados para determinar la validez de las técnicas de detección de Spam y de páginas Soft-404, han sido presentados en las secciones V.4 y VI.4. Ambos experimentos han sido realizados sobre conjuntos de páginas web reales que forman parte de dataset públicos y ampliamente conocidos. Esto permite que cualquier investigador pueda probar y analizar las técnicas presentadas.

   Los resultados obtenidos para las técnicas de detección de Web Spam y Soft-404, logran porcentajes de efectividad muy altos, superando el 90% de precisión y recall, y en algún caso alcanzando el 100%.

   Por otro lado se han realizado un conjunto de experimentos para evaluar el sistema de detección de cambios web propuesto. Los experimentos realizados se muestran en la sección VII.4. Los experimentos presentados incluyen resultados de: un sitio web propio, 150 dominios públicos, la eficiencia del sistema propuesto y la simulación de resultados.

   Los resultados obtenidos por el sistema mejoran en gran medida los resultados obtenidos por los sistemas presentes en el estado del arte.

VIII.2 Conclusiones

Esta sección presenta las principales contribuciones y conclusiones obtenidas.

VIII.2.1 Principales contribuciones

Entre el conjunto de contribuciones, que se pueden obtener de esta tesis y de los estudios que la conforman, se pueden destacar las siguientes:

1. Una arquitectura de crawling eficiente. La arquitectura se basa en las arquitecturas de crawling existentes y se extiende con un conjunto de módulos que abordan los principales desafíos de la Web. Para ello, el sistema integra componentes para la detección de Web Spam y páginas Soft-404, así como para mejorar el proceso de actualización de los contenidos del repositorio utilizando el menor número de recursos.

2. Un conjunto de estudios que analizan y caracterizan la Web tanto global como nacional. Dichos estudios permiten conocer en más detalle qué problemas tienen los sistemas de crawling para procesar la Web y cómo evolucionan dichos problemas.

   Se han realizado dos estudios. El primero de ellos estudia la Web Oculta del lado cliente, su presencia en la Web y la capacidad de los crawlers actuales para procesar las tecnologías web usadas en el lado cliente. Esto permite conocer qué deficiencias tienen los actuales crawlers, que deben ser mejoradas, y qué parte de la Web no está siendo tratada.

   El segundo de los estudios se centra en un análisis de la Web Global y española analizando diferentes características. Entre otras características se han estudiado: la similitud de contenidos, la Web Oculta del lado cliente, la edad de las páginas, el crecimiento de la Web, las tecnologías web del lado cliente y del lado servidor, etc. Los resultados de este análisis han sido el punto de partida para el diseño de la arquitectura de crawling propuesta y para los módulos que esta contiene.

3. Un conjunto de heurísticas para la detección de páginas web de Spam. En el proceso de crawling de la Web existen multitud de recursos inadecuados que no deben ser procesados, entre ellos están las páginas web de Spam. Las técnicas presentadas mejoran las presentes en el estado del arte, tanto en eficacia como en eficiencia. Están basadas en el análisis del contenido, pero tratan de detectar la mayor parte de los tipos de Web Spam (Content Spam, Cloaking, Redirection Spam y Link Spam). Se presenta un método de combinación del conjunto de heurísticas propuestas, para su posterior uso como un módulo que extienda a un sistema de crawling y le permita detectar y evitar páginas de Spam.

4. Nuevas técnicas para la detección automática de páginas Soft-404. Este tipo de páginas al igual que las de Spam, no deben ser procesadas, ya que su contenido y enlaces no son relevantes. El conjunto de heurísticas propuesto mejora los sistemas de detección de páginas Soft-404 actuales, tanto en eficacia como en eficiencia. Dichas técnicas forman parte de un módulo de la arquitectura propuesta que permite detectar dicho tipo de páginas. De esta forma se mejora la calidad de las páginas procesadas y también el uso de los recursos, ya que aquellos recursos que no traten este tipo de páginas pueden ser utilizados para procesar otras páginas web.

5. Un sistema que permita conocer al sistema de crawling las modificaciones realizadas en las diferentes páginas y sitios web. El sistema permitirá a los propietarios de páginas web que sus contenidos sean actualizados en "tiempo real" por los motores de búsqueda. Para ello, dicha página web deberá utilizar

el sistema propuesto y ésta será monitorizada. De esta forma el sistema de crawling conocerá las páginas que han sido modificadas en "tiempo real". Esto permite mejorar la calidad de los contenidos indexados, ya que serán más actuales, lo cual repercute en la calidad de las búsquedas y en la experiencia del usuario web, y mejorará también el uso de recursos, ya que evitará volver a procesar páginas que no han sufrido ningún cambio.

VIII.2.2 Conclusiones obtenidas

En esta sección se enumeran y discuten las principales conclusiones extraídas de la presente tesis.

CONCLUSIÓN 1: Es posible diseñar soluciones de crawling eficiente.

La arquitectura presentada en esta tesis doctoral considera todas las tareas necesarias para realizar un crawling eficiente desde el punto de vista del contenido procesado. En el presente trabajo, se han propuesto soluciones para aquellas tareas para las que no existían soluciones adecuadas tanto a nivel de eficacia como de rendimiento para ser adaptadas a sistemas de crawling (detección de Web Spam y páginas Soft-404), y se han presentado nuevas técnicas que mejoran las soluciones propuestas en el estado del arte (actualización de contenidos indexados). El resto de las tareas involucradas en la arquitectura, pueden ser implementadas haciendo uso de las técnicas presentes en la literatura, como se describe en la sección IV.1, donde se describe la arquitectura propuesta.

Las técnicas y métodos propuestos han sido implementados y evaluados en diversos datasets públicos, con fuentes web reales, y pertenecientes a diferentes dominios. Los resultados obtenidos en estos experimentos han sido muy prometedores, y avalan el uso de la arquitectura propuesta en sistemas de crawling de altas prestaciones.

CONCLUSIÓN 2: Es posible automatizar el proceso de detección de Web Spam y Soft-404 en la Web. La arquitectura propuesta en la sección IV.1, permite detectar páginas web de Spam y Soft-404, mediante la utilización de un método que permite combinar un conjunto de novedosas heurísticas.

Actualmente, la Web contiene gran cantidad de páginas de Spam y de Soft-404. El estudio presentado por Ntoulas *et al.* [NM06], indica que entorno al 70% de las páginas del dominio .biz son Web Spam, el 35% de las páginas del dominio .us, y el 15% y 20% de las páginas web bajo los dominio .com y .net, respectivamente. Según el estudio realizado por Bar-Yossef *et al.* [BYBKT04], el 29% de los enlaces rotos apuntan a páginas Soft-404. Estos datos reflejan el grave problema que significa este tipo de páginas para el procesamiento de la Web por parte de los sistemas de crawling.

Detectar y evitar páginas de Spam y Soft-404, permite mejorar la calidad de las páginas tratadas y ahorrar recursos, que pueden ser usados en procesar otros sitios web. Por tanto el módulo propuesto, permite mejorar la calidad y la cantidad de recursos web tratados.

CONCLUSIÓN 3: Es posible automatizar la detección de cambios en páginas y sitios web. Otro de los mayores problemas de los crawlers, es el mantenimiento actualizado de los contenidos ya procesados. Actualmente, los crawlers tienen gran cantidad del tiempo los repositorios desactualizados, ya que detectan tarde los cambios realizados en las páginas web, y en muchas otras ocasiones, acuden a páginas web que no han sufrido ninguna modificación. Para evitar esto, los sistemas de crawling y motores de búsqueda definen un conjunto de políticas que les ayudan a decidir cuándo es el momento óptimo para volver a procesar una página web. Sin embargo, estas políticas están basadas en datos estadísticos, y por lo tanto es probable que fallen (perdiendo cambios realizados en páginas web o procesando páginas más veces de lo necesario).

En esta tesis, se ha presentado un sistema (capítulo VII) que permite de forma automática detectar cambios en páginas web. Esto permite, tanto mejorar la calidad de las páginas web indexadas, como ahorrar recursos, evitando procesar páginas web que no han cambiado, lo cual permitirá usar dichos recursos para procesar nuevos contenidos web. Por tanto, al igual que los módulos de detección de Web Spam y de Soft-404, el módulo propuesto permite mejorar la calidad y la cantidad de recursos web tratados.

CONCLUSIÓN 4: Las técnicas de detección de Web Spam y de páginas Soft-404 son efectivas con fuentes web reales. Las heurísticas presentadas en los capítulos V y VI, permiten detectar páginas web de Spam y de Soft-404. Parten del análisis del contenido de la página web y tratan de detectar los diferentes tipo de Spam existentes, y si se trata o no de una página Soft-404.

Los resultados obtenidos en ambos casos son bastante prometedores: todas las métricas muestran valores altos y algunos incluso alcanzan el 100%. Estos experimentos han sido realizados sobre dataset públicos, con fuentes web reales.

CONCLUSIÓN 5: La implementación de las técnicas propuestas supone una solución a los problemas del crawling eficiente.

El prototipo desarrollado para la presente tesis, no constituye una implementación completa de la arquitectura propuesta, sin embargo permite concluir que las técnicas propuestas constituyen una solución efectiva a los problemas de detección de Web Spam y Soft-404, y al proceso de actualización de contenidos por parte de los sistemas de crawling.

Los resultados obtenidos en los diferentes experimentos han aportado resultados prometedores. Dichos resultados muestran que se ha conseguido, por un lado detectar y evitar el procesamiento de páginas web de Spam y Soft-404, y por otro mejorar en gran medida el proceso de actualización de los contenidos. La arquitectura propuesta supone una solución a los problemas de ineficiencia de los sistemas de crawling actuales.

VIII.3 Líneas de Trabajo Futuro

Esta sección describe las lineas de trabajo actuales y futuras del autor.

VIII.3.1 Estudio de la Web

La base del estudio presentado en esta tesis, ha sido analizar la Web para encontrar aquellos elementos que dificultan la tarea de los sistemas de crawling. Este conocimiento da la posibilidad de mejorar los crawlers para hacer frente a dichos desafíos. Evitando dichos problemas o logrando solucionarlos se conseguirá mejorar el rendimiento y la calidad de los sistemas de crawling.

Por ello, el trabajo futuro se centrará en seguir procesando la Web y creando nuevos datasets, que permitan en sucesivos años continuar con el análisis. Estos sucesivos estudios permitirán determinar cómo evoluciona la Web, sus usuarios y los creadores de sitios web. Los resultados de los análisis y las pautas de actuación extraídas ayudarán a crear y modificar las políticas de actuación de los sistemas de crawling, mejorando su rendimiento.

Por otro lado, se quiere realizar otra serie de estudios, en paralelo a los citados, que permitan determinar la calidad de la Web. Comprobar si está en decadencia (calidad de contenidos, enlaces rotos, contenidos repetidos o similares, etc.), o por el contrario está mejorando. Estos estudios incluirán un apartado sobre la seguridad de la Web (protocolos de seguridad que se usan; uso de tecnologías seguras; sistemas correctamente actualizados, etc) y los peligros que presenta para el usuario.

Por último, en relación al análisis sobre la Web, se probarán las políticas de mejora del rendimiento presentadas, en sistemas de crawling de altas prestaciones, con la idea de demostrar su validez y la mejora del rendimiento de dichos sistemas.

VIII.3.2 Tratamiento de Tecnologías del Lado Cliente

Entre los estudios realizados sobre la Web, se analizó el tratamiento de la Web Oculta del lado cliente por parte de los sistemas de crawling actuales. Se comprobó que estos tenían multitud de deficiencias y que existía una amplia parte de la Web que no estaba siendo tratada.

En primer lugar, se propone mejorar los métodos de evaluación en base a la escala. Por otro lado, se plantea estudiar cómo afectan las diferentes características de un sitio web a su indexación y al análisis que los crawlers hacen de su contenido para localizar nuevos enlaces. De esta forma se podría determinar si la importancia, la temática, la relevancia y el número de visitas de un sitio afecta a cómo son tratados por los crawlers.

Por último, debido a que los resultados obtenidos por los sistemas de crawling en el tratamiento de la Web Oculta del lado cliente, y a que los crawlers que han obtenido mejores resultados no tienen su código público, se diseñará e implementará un algoritmo que permita extraer enlaces de las tecnologías mostradas sin necesidad de utilizar mini-navegadores ni intérpretes completos. Esto permitirá que pueda ser usado en un crawling a nivel global. El siguiente paso será diseñar y desarrollar un sistema de crawling que permita adaptar dicho algoritmo para así poder tratar una mayor cantidad de Web Oculta. Dicho crawler logrará procesar un mayor y mejor

número de recursos web, por lo que los resultados mostrados al usuario y su experiencia también mejorarán.

VIII.3.3 Técnicas de Detección de Web Spam

Con respecto a las técnicas de detección de Web Spam presentadas, se realizará un estudio sobre los beneficios de incluir el módulo en un navegador web, que permita detectar este tipo de páginas al usuario y de esta forma le ayude a evitar posibles riesgos.

En este estudio se ha analizado el uso de las técnicas de detección propuestas, como parte de un módulo de un sistema de crawling que posibilite la detección de Web Spam antes de ser procesado. Como estudio complementario, se quiere discutir la posibilidad de generalizar el detector de Web Spam a una arquitectura distribuida, basada en el modelo Map-Reduce [DG08], el más usado en entornos de crawling.

Otro estudio, consiste en estudiar en más detalle el tiempo de cómputo de cada heurística, proponiendo un mayor número de agrupaciones de técnicas, que permitirá proponer más políticas de actuación de un sistema de crawler. El reto será estudiar el comportamiento de cada política propuesta, analizando su eficacia y su eficiencia en las diferentes situaciones en las que se puede encontrar un crawler.

Por ultimo, se desea seguir estudiando páginas de Web Spam que permitan conocer las nuevas técnicas utilizadas por los spammers. Esto permitirá proponer nuevas heurísticas que ayuden a identificar Web Spam. Por otro lado, se quiere estudiar la posibilidad de combinar las diferentes técnicas de detección de Web Spam. Por ejemplo, combinar el estudio presentado con análisis de grafos de la web y la importancia en la relación entre sitios web; reconocimiento de texto generado automáticamente [MS99]; detección de texto oculto (colores, tamaño de texto, tipografía, etc.) en base al análisis del código HTML y de la hoja de estilo (CSS) asociada.

VIII.3.4 Técnicas de Detección de Soft-404

En esta tesis se han presentado técnicas para la detección de páginas Soft-404. Un estudio futuro será tratar de diferenciar entre las páginas Soft-404, aquellas que son de parking de aquellas que no lo son. Para ello, se necesitará crear y etiquetar un nuevo dataset diferenciando páginas de parking, páginas Soft-404 y páginas normales, y crear un nuevo conjunto de heurísticas que permitan clasificarlas lo mas correctamente posible.

Por otro lado, de igual forma que para las técnicas de detección de Web Spam, se estudiará la posibilidad de generalizar el detector de páginas Soft-404 a una arquitectura distribuída, basada en el modelo Map-Reduce [DG08], que es el más usado en entornos de crawling actualmente.

VIII.3.5 Sistema de Detección de Cambio Web en Entornos Reales

En este trabajo se presenta un sistema, basado en una arquitectura colaborativa, que permite detectar modificaciones realizadas en páginas web casi en tiempo real. Por otro lado se ha presentado dicho sistema como un nuevo módulo de la arquitectura de crawling eficiente propuesta.

El trabajo futuro se centrará en diseñar e implementar dicho sistema para que pueda ser utilizado en sistemas de crawling existente (Nutch [KC04], Heritrix [MKSR04], etc.). Con la misma idea de adaptar el sistema a un crawler, se estudiará el diseño del sistema siguiendo una arquitectura Map-Reduce [DG08]. Esto permitirá el uso del sistema en crawlers basados en esa misma arquitectura.

El sistema propuesto, realiza el resumen de la página web generando un hash MD5 de las diferentes partes de dicha página web. Un trabajo futuro será cambiar el modo en que se genera el resumen, creando un único hash que resuma el contenido, y que a mayor número de bytes diferentes, entre un hash antiguo y uno actual, indique un mayor porcentaje de cambio en el contenido [MJDS07]. De esta forma, se podrá saber aproximadamente qué porcentaje de la página ha cambiado y se podrá decidir mejor cuándo volver a procesar la página.

Por otro lado, se pretende realizar un estudio que analice en mayor detalle las políticas de recrawling utilizadas por los motores de búsqueda. Discutiendo diferentes características de las páginas y sitios web que ayudan a estimar la frecuencia de cambios y que permitan obtener políticas de refresco complementarias a la información que aporta el sistema. De esta forma se logrará mejorar la probabilidad de éxito (revisitar una página en el menor tiempo posible desde que una página web ha sido modificada) cuando se decide reprocesar una página web.

Por último, se estudiará la optimización de diferentes aspectos del sistema propuesto, tales como la seguridad del sistema ante posibles ataques.

IX

Conclusions and Future Work

This chapter concludes the thesis. Section IX.1 evaluates the compliance with the initial aims. Section IX.2 discusses the contributions presented in the thesis and the conclusions of this work. Finally, section IX.3 explains lines for future work by the author.

IX.1 Assessment of the Objectives Compliance

This section assesses the achievement of the objectives of this thesis. These aims were explained in detail in section I.2.

Below is a brief summary of each of the aims set out at the beginning of this work, justifying how each has been achieved.

1. *Propose an efficient architecture for web crawling systems.*

   A crawling architecture that improves the quality and quantity of the processed web pages was created. The architecture is presented in chapter IV.

   The architecture is based on existing architectures and is extended with a set of modules that address the key challenges of the web. The architecture, on the one hand, avoids the processing of pages and websites of Spam and Soft-404. In this way, it improves the quality of the processed web resources and increases the number of free resources, which can be used to process a greater number of web pages.

   On the other hand, the architecture includes a module that allows the updating process of indexed resources to be improved. This module is based on a novel system to detect changes in web pages almost in real time. Thus, once again, the crawler manages to improve the quality of processed resources, as the indexed contents are more up-to-date, and will use fewer resources in the updating process of the repositories.

   The stated crawling modules improve the current crawling architectures, allowing an efficient crawling of the Web.

2. *Perform a set of studies to know the characteristics of the Web, its evolution and the differences between web content in different countries. This set of studies will help to identify the problems in the processing of the Web by crawling systems.*

   Two studies have been presented in chapter III, sections III.1 and III.2. The first one studies the client side hidden Web. This study identifies the main web technologies and the different methods of generating links used by web site creators. Following on, the study presents a scale and a set of evaluation methods with it, which measure the capability of existing crawler systems to process the client side hidden Web.

The second study, analyzes and compares, during 3 consecutive years, the general characteristics of the Global and Spanish Web. Mainly, this study is focused on those features of the Web that affect the processing from the point of view of crawling systems. Among other features which have been studied, are: the similarity of contents, the client side hidden Web, the age of the pages, the growth of the Web, client side web technologies and server side, etc. After these analysis, a set of rules to be adopted by the crawlers have been proposed to improve their performance.

The results obtained from both studies have identified challenges in processing Web content (Spam, content update, Hidden Web, etc.), and have formed the basis of other studies, which have led to the proposed crawling architecture.

3. *Propose new techniques and algorithms to allow existing crawling systems to detect Web Spam.*

   In chapter V, a set of innovative heuristics to detect Web Spam have been presented. These heuristics use the analysis of the content for detection, trying to detect most types of Web Spam and the diverse combinations of techniques that exist.

   The proposed techniques improve the present state of the art in terms of both effectiveness and efficiency. Web spam detection is based on a method proposed that combines the set of heuristics proposed in the thesis. This method, along with heuristics is part of a new module of the proposed crawling architecture, which is responsible for detecting Spam pages. The proposed module analyzes the downloaded web pages, through this the crawling system allows processed contents to be improved and avoid wasting resources, which would occur if the detection were made subsequently.

4. *Propose new techniques and algorithms to allow existing crawling systems to detect Soft-404 pages.*

   A set of techniques that allow the detection of Soft-404 pages have been proposed and evaluated. The analysis of these techniques is carried out in chapter VI.

   In the same way as with Spam pages, the detection will be made from the content analysis. The proposed set of heuristics improve the detection systems of Soft-404 pages in terms of both effectiveness and efficiency. This method, along with heuristics, is part of a new module of the proposed crawling architecture. This module will be responsible for avoiding the processing of Soft-404 pages.

5. *Propose a system to let the crawler know modifications made on web pages and websites.*

   We have created and tested a system that allows, on one hand, website owners to have content updated in "real time" by search engines, and on the other hand, crawling systems to improve content updating processes, as they will know when a web page has been modified. In chapter VII describes its components and operations.

   The system is based on a collaborative architecture between the client side and server side, so it must not be intrusive and must be highly scalable. So, the crawler will avoid accessing web resources when they

have not been modified and will not lose changes in the time between two consecutive processings of the web page. On the other hand, the quality of indexed contents will be improved, since the contents will be more up-to-date, which in turn will affect the quality of search results and the web user experience.

6. *Validate the effectiveness and efficiency of the methods and techniques proposed with experiments on real web pages and sites, showing that it is possible to develop crawling systems capable of detecting and avoiding Spam and Soft-404 pages, as well as performing an efficient recrawling of previously indexed content.*

   The experiments carried out to determine the validity of the Web Spam and Soft-404 detection techniques have been presented in sections ref resultadoswebspam and ref resultadossoft404. Both experiments were performed on sets of real web pages that are part of public and well known datasets. This allows any researcher to test and analyze the techniques presented.

   The obtained results for the detection techniques of Web Spam and Soft-404 pages achieved very high effectiveness rates, around 90% of precision and recall, and in some cases reached 100%.

   On the other hand, a set of experiments to evaluate the proposed web change detection system have been carried out. The experiments are shown in Section VII.4. The experiments presented include results on: a personal website, 150 public domains, the efficiency of the proposed system and the simulation results.

   The obtained results showed that the system greatly improves the results obtained by the systems present in the state of art.

IX.2 Conclusions

This section presents the main contributions and conclusions.

IX.2.1 Main contributions

Among the set of contributions, which can be obtained in this thesis and the studies that form it, we can highlight the following:

1. An efficient crawling architecture. The architecture is based on existing crawling architectures and is extended by a set of modules that address the main challenges of the Web. The system integrates components for Web Spam and Soft-404 detection, and to improve the updating process of the contents using the fewest resources possible.

2. A set of studies to analyze and characterize the global and national Web. The idea of these studies is to understand in detail the problems of the crawling systems to process the Web and how these problems evolve.

   There have been two studies. The first one studies the client side hidden Web, their presence on the Web and the capability of existing crawlers to process web technologies used on the client side. This lets us know the deficiencies of the crawlers, which should be improved, and that part of the Web is not being processed.

   The second study analyses different characteristics on the Spanish and Global Web. Among other features, the study analyses: the similarity of content, client side Hidden Web, the age of pages, the growth of the Web, the technologies on the client/server side Web, etc. The results of this analysis have been the starting point for the design of the crawling architecture proposed and for the modules it contains.

3. A set of heuristics to detect Web Spam. In the Web crawling process there are many inadequate web resources that should not be processed, among them are Spam pages. The proposed techniques improve those presented in the state of the art, in both effectiveness and efficiency. They are based on content analysis and try to detect the majority of Web Spam types (Content Spam, Cloaking, Redirection Spam and Link Spam). A method to combine the proposed set of heuristics to be used as a module of a crawling system is presented here, one which allows it to detect and avoid Spam pages.

   The analysis of each of the heuristics, the method of combining them, the final creation of a detection module, and results obtained and its comparison with the presented ones in the state of the art.

4. New techniques to automatically detect Soft-404 pages. This type of web page, as with Spam pages, must not be processed, since the content and links are not relevant. The proposed set of heuristics improves the existing detection systems in both effectiveness and efficiency. These techniques are part of a module of the proposed architecture to detect this type of pages. This will improve the quality of processed pages and the use of resources, as resources which do not process this type of page can be used to process other websites.

5. A system that allows the crawling system to know changes made on different web pages and websites. The system will allow website owners to have contents updated in "real time" by search engines. For this, websites must use the proposed system and then, each of them will be monitored by the system. Thus, the crawling system will know when pages have been changed in "real time". This improves the quality of content indexed, as they will be more up-to-date, which affects the quality of searches and the web user experience. So, the crawling system improves the use of resources, as it will prevent reprocessing pages that have not been modified.

IX.2.2 Obtained Conclusions

This section lists and discusses the main conclusions of this thesis.

CONCLUSION 1: It is possible to design solutions for efficient crawling.

The architecture presented in this thesis considers all necessary tasks for performing an efficient crawling from the point of view of processed content. The present study proposes solutions for those tasks that were not appropriate solutions, in both effectiveness and performance, to be adapted to crawling systems (detection of Web Spam and Soft-404 pages). This thesis also presents new techniques that improve solutions proposed in the state of the art (updating of indexed content). The rest of the tasks involved in the architecture may be implemented using techniques in the literature, as described in section IV.1, where the proposed architecture is described.

The techniques and methods proposed have been implemented and evaluated on different public datasets, with real web pages, belonging to different domains. The results obtained in these experiments were very promising and support the use of the proposed architecture in high performance crawling systems.

CONCLUSION 2: It is possible to automate the process of detection of Web Spam and Soft-404 on the Web.

The architecture proposed in section IV.1 detects Web Spam and Soft-404 pages, using a method to combine a new set of heuristics.

Currently, the Web contains many Spam and Soft-404 pages. The study presented by Ntoulas *et al.* [NM06], indicates that around 70% of pages of .biz domain are Web Spam, 35% of the pages in the .us domain, and 15% and 20% of web pages on the domain .com and .net, respectively. According to the study by Bar-Yossef *et al.* [BYBKT04], 29% of broken links point to Soft-404 websites. These data indicate the serious problem that these pages represent for crawling systems processing the Web.

Detecting and avoiding Spam and Soft-404 pages improves the quality of processed websites and saves resources, which can be used to process other websites. Therefore, the proposed module, improves the quality and quantity of processed web resources.

CONCLUSION 3: It is possible to automate the detection of changes in web pages and websites.

Another major challenge for crawlers, is the updating of processed content. Currently, the repositories of the crawlers are much of the time outdated, since web changes are detected later, and on many other occasions, the crawler visits the web pages and it has not been modified. To avoid this, crawling systems and search engines define a set of policies that help them to decide when is the optimal time to reprocess a website. However, these policies are based on statistical data, and are therefore, likely to fail.

This thesis presents a system (chapter VII) that allows changes in web pages to be detected automatically. The quality of indexed web pages can therefore be improved, and resources can be saved, because those web

pages that have not been modified will not be processed, and thus the crawler may use these resources to process new web pages. So, like the modules for detecting Web Spam and Soft-404, the proposed module improves the quality and quantity of processed web pages.

CONCLUSION 4: The techniques for detecting Web Spam and Soft-404 pages are effective with real web sources.

The heuristics presented in chapters V and VI allow Spam and Soft-404 pages to be detected. Based on content analysis of the website, these heuristics try to detect the different types of existing Spam, and determine whether a web page is a Soft-404 page.

The results in both cases are very promising: all metrics show high values and even, in some cases, reaching 100%. These experiments have been performed on public dataset with real web pages.

CONCLUSION 5: The implementation of the proposed techniques is a solution to the problems of inefficient crawling.

The prototype developed for this thesis, is not a complete implementation of the proposed architecture, however, it can be concluded that the proposed techniques provide an effective solution to the detection problems of Web Spam and Soft-404 pages, and to the process of updating content by crawling systems.

The results obtained in different experiments have provided promising results. These results show that it has managed, on the one hand, to detect and avoid processing of Spam websites and Soft-404 pages, and on the other hand, improve the process of updating the content. The proposed architecture is a solution to the inefficiency problems of the existing crawling systems.

IX.3 Lines of the Future Work

This section describes the lines of current and future work of the author.

IX.3.1 Study of the Web

The basis of the study presented in this thesis was to analyze the Web to find those elements that make the task of the crawling systems more difficult. This knowledge makes it possible to improve the crawlers to address these challenges. Avoiding or solving these problems improves the performance and quality of crawling systems.

Therefore, future work will be crawling the Web and creating new datasets, allowing us to analyze successive years. These successive studies will determine how the Web evolves, both users and website creators. The results of the analysis and patterns of action obtained will help to create and modify the operation policies of crawling systems, and thus improve performance.

On the other hand, it is necessary to make another series of studies, in parallel to the above, to determine the quality of the Web. Check whether the Web is in decline (content quality, broken links, repetitive content, etc.), or, if on the contrary, is improving. These studies will include a section about web security (security protocols used, use of secure technologies, updated systems, etc.) and the dangers it presents to the user.

Finally, in relation to analysis on the Web, the presented improvement policies on high performance crawling systems will be tested, with the idea of proving their validity and improvement in performance of the crawlers.

IX.3.2 Processing of the Client Side Web Technologies

Among the studies carried out about the Web, the processing of client side Hidden Web by the crawling systems was analyzed. It was found that the crawlers had many deficiencies and that there was a large part of the Web that was not being processed.

The first aim is to improve evaluation methods based on the scale. On the other hand, a study analyzing how the different features of a website could affect its indexing and the link extraction process performed by the crawlers is being contemplated. Therefore, this study would determine whether the relevance, the topic and the number of visits to a website affect the way that these website are processed by crawlers.

Finally, due to the results obtained processing the client side Hidden Web by crawling systems, and considering that the crawlers with best results are not opensource, an algorithm that extract links used in the shown technologies without using mini-browsers or interpreters will be designed and developed. This saving of resources will allow this algorithm to be used in a global crawling process. The next step will be to design and develop a crawling system for adapting the algorithm in order to process a larger part of the Hidden Web. This crawler will process a larger and better number of web resources, so the results shown to the user and its experience will improve.

IX.3.3 Techniques to Web Spam Detection

Regarding detection techniques of Web Spam, future study will address the benefits obtained including the module in a web browser, to detect this type of pages and thus help to avoid possible risks.

This study analyzed the use of the proposed Web Spam detection techniques as a module of a crawling system that enables the detection of web spam before processing. A complementary study is to discuss the possibility of generalizing the Web Spam detector to a distributed architecture, based on the Map-Reduce model [DG08], which, currently, is the most widely used in crawling environments.

Another study will examine in detail the computation time of each heuristic, suggesting a larger number of heuristic groups, allowing more operation policies for a crawler system to be proposed. The challenge will be to study the behavior of each proposed policy, analyzing their effectiveness and efficiency in the different

situations a crawler can be found.

Finally, I want to analyse Web Spam pages that will let us know new techniques used by spammers. This will allow us to create new heuristics to detect Web Spam. On the other hand, there is a need to study the possibility of combining the different techniques of Web Spam detection. For example, combining the study presented with analysis of the web graph and the importance of the relationship between websites; recognition automatically generated text [MS99], detection of hidden text (colors, text size, font, etc.) based on analysis of the HTML code and the associated css.

IX.3.4 Techniques to Soft-404 Detection

This thesis presents techniques for detecting Soft-404 pages, a future study will be to differentiate between Soft-404 pages and parking pages. For this, it is necessary to create and a new dataset with three tags, parking pages, Soft-404 pages and normal pages, and create a new set of heuristics that allow the detection of the different types of pages.

On the other hand, as with the techniques used for the detection of Web Spam, I will study the possibility of generalizing the Soft-404 detector to a distributed architecture, based on the Map-Reduce model [DG08], which is currently the most widely used in crawling environments.

IX.3.5 Web Change Detection System on real environments

This thesis presents a system based on a collaborative architecture, which detects in "real time" changes made to web pages. Moreover, this system has been presented as a new module of the efficient crawling architecture proposed.

Future work will be to design and develop the proposed system to be used in existing crawling systems (Nutch [KC04], Heritrix [MKSR04], etc.). Similarly, the possibility of adapting the system to a Map-Reduce architecture will be studied [DG08]. This will allow the use of the system in crawlers based on the same architecture.

The proposed system performs the summary of the website generating a MD5 hash of the different parts of the website. A study will be to change the way the summary is generated, creating a unique hash that summarizes all the content, and moreover, according to the number of bytes different between the stored hash and one current hash, indicates the percentage change of the content [MJDS07]. Thus, the system will be able to know approximately what percentage of the page has changed and decide what the optimal moment to reprocess the page is.

On the other hand, a study to analyze in more detail recrawling policies used by the search engines. The study will analyze the different characteristics of web pages and websites that help to estimate the frequency of

changes, and that allow the creation of updating policies complementary to the data provided by the system. By doing so, it will improve the probability of success (revisiting a modified web page as quickly as possible) when deciding to reprocess a web page.

Finally, I will study the different improvements to the proposed system, such as improving the system security against possible attacks.

Bibliografía

[ABHN05] Mikael Andersson, Anders Bengtsson, Martin Höst, and Christian Nyberg. Web server traffic in crisis conditions. In *In In Proceedings of the Swedish National Computer Networking Workshop, SNCNW*, 2005.

[ACD+03] Einat Amitay, David Carmel, Adam Darlow, Ronny Lempel, and Aya Soffer. The connectivity sonar: detecting site functionality by structural patterns. In *In Proceedings of the Fourteenth ACM Conference on Hypertext and Hypermedia*, pages 38–47. ACM Press, 2003.

[ACGM+01] Arvind Arasu, Junghoo Cho, Hector Garcia-Molina, Andreas Paepcke, and Sriram Raghavan. Searching the web. *ACM Trans. Inter. Tech.*, 1(1):2–43, 2001.

[AKK+09] Susumu Akamine, Yoshikiyo Kato, Daisuke Kawahara, Keiji Shinzato, Kentaro Inui, Sadao Kurohashi, and Yutaka Kidawara. Development of a large-scale web crawler and search engine infrastructure. In *Proceedings of the 3rd International Universal Communication Symposium*, IUCS '09, pages 126–131, New York, NY, USA, 2009. ACM.

[Álv07] Manuel Álvarez. *Arquitectura para Crawling Dirigido de Información Contenida en la Web Oculta*. PhD thesis, Universidade da Coruna, 2007.

[BB96] Leo Breiman and Leo Breiman. Bagging predictors. In *Machine Learning*, pages 123–140, 1996.

[BB98] Krishna Bharat and Andrei Broder. A technique for measuring the relative size and overlap of public web search engines. In *Proceedings of the seventh international conference on World Wide Web 7*, WWW7, pages 379–388, Amsterdam, The Netherlands, The Netherlands, 1998. Elsevier Science Publishers B. V.

[BBC+08] Paolo Boldi, Francesco Bonchi, Carlos Castillo, Debora Donato, Aristides Gionis, and Sebastiano Vigna. The query-flow graph: model and applications. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 609–618, New York, NY, USA, 2008. ACM.

[BCD+06] Luca Becchetti, Carlos Castillo, Debora Donato, Stefano Leonardi, and Ricardo Baeza-Yates. Link-based characterization and detection of web spam. In *In AIRWeb*, 2006.

[BCSV02] Paolo Boldi, Bruno Codenotti, Massimo Santini, and Sebastiano Vigna. Structural properties of the African Web. In *Proceedings of the eleventh international conference on World Wide Web*, Honolulu, Hawaii, USA, May 2002. ACM Press.

[Ber01] Michael K. Bergman. White paper: The deep web. surfacing hidden value. *The Journal of Electronic Publishing*, 7(1):online, Aug. 2001.

[BGMZ97a] Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig. Syntactic clustering of the web. *Comput. Netw. ISDN Syst.*, 29(8-13):1157–1166, September 1997.

[BGMZ97b] Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig. Syntactic clustering of the web. In *Selected papers from the sixth international conference on World Wide Web*, pages 1157–1166, Essex, UK, 1997. Elsevier Science Publishers Ltd.

[BI04] Lennart Björneborn and Peter Ingwersen. Toward a basic framework for webometrics. *Journal of the American Society for Information Science and Technology*, 55:1216–1227, 2004.

[BKM+00] Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. Graph structure in the web. *Computer Networks*, 33(1-6):309 – 320, 2000.

[BLCL+94] Tim Berners-Lee, Robert Cailliau, Ari Luotonen, Henrik Frystyk Nielsen, and Arthur Secret. The world-wide web. *Commun. ACM*, 37(8):76–82, August 1994.

[BNW03] Andrei Z. Broder, Marc Najork, and Janet L. Wiener. Efficient url caching for world wide web crawling. In *Proceedings of the 12th international conference on World Wide Web*, WWW '03, pages 679–689, New York, NY, USA, 2003. ACM.

[BP98] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the seventh international conference on World Wide Web 7*, WWW7, pages 107–117, Amsterdam, The Netherlands, The Netherlands, 1998. Elsevier Science Publishers B. V.

[Bur97] Mike Burner. Crawling towards eternity: Building an archive of the world wide web. *Web Techniques Magazine*, 2(5), May 1997.

[BWN07] Roger Braunstein, Mims H. Wright, and Joshua J. Noble. *ActionScript 3.0 Bible*. Wiley, October 2007.

[BY11] Berthier Baeza-Yates, Ricardo; Ribeiro-Neto. *Modern information retrieval: the concepts and technology behind search*. Addison-Wesley Longman Publishing Co., Inc., 2011.

[BYBKT04] Ziv Bar-Yossef, Andrei Z. Broder, Ravi Kumar, and Andrew Tomkins. Sic transit gloria telae: towards an understanding of the web's decay. In *Proceedings of the 13th international conference on World Wide Web*, WWW '04, pages 328–337, New York, NY, USA, 2004. ACM.

[ByC02] Ricardo Baeza-yates and Carlos Castillo. Balancing volume, quality and freshness in web crawling. In *In Soft Computing Systems - Design, Management and Applications*, pages 565–572. IOS Press, 2002.

[BYC04] Ricardo Baeza-Yates and Carlos Castillo. La web chilena, 2004.

[BYC10] Ricardo Baeza-Yates and Carlos Castillo. Language detection library for java, 2010.

[BYCE07] Ricardo Baeza-Yates, Carlos Castillo, and Efthimis N. Efthimiadis. Characterization of national web domains. *ACM Trans. Internet Technol.*, 7, May 2007.

[BYP06] Ricardo Baeza-Yates and Barbara Poblete. Dynamics of the chilean web structure. *Comput. Netw.*, 50:1464–1473, July 2006.

[BYSJC02] Ricardo Baeza-Yates, Felipe Saint-Jean, and Carlos Castillo. Web structure, dynamics and page quality. In Alberto Laender and Arlindo Oliveira, editors, *String Processing and Information Retrieval*, volume 2476 of *Lecture Notes in Computer Science*, pages 453–461. Springer Berlin / Heidelberg, 2002.

[CC04] Mike Cafarella and Doug Cutting. Building Nutch: Open Source Search: A case study in writing an open source search engine. *ACM Queue*, 2(2), 2004.

[CD11] Carlos Castillo and Brian D. Davison. Adversarial web search. *Found. Trends Inf. Retr.*, 4(5):377–486, May 2011.

[CGM00a] Junghoo Cho and Hector Garcia-Molina. The evolution of the web and implications for an incremental crawler. In *Proceedings of the 26th International Conference on Very Large Data Bases*, VLDB '00, pages 200–209, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[CGM00b] Junghoo Cho and Hector Garcia-Molina. Synchronizing a database to improve freshness. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, SIGMOD '00, pages 117–128, New York, NY, USA, 2000. ACM.

[CGM03a] Junghoo Cho and Hector Garcia-Molina. Effective page refresh policies for web crawlers. *ACM Trans. Database Syst.*, 28(4):390–426, December 2003.

[CGM03b] Junghoo Cho and Hector Garcia-Molina. Estimating frequency of change. *ACM Trans. Internet Technol.*, 3:256–290, August 2003.

[CGmP98] Junghoo Cho, Hector Garcia-molina, and Lawrence Page. Efficient crawling through url ordering. In *COMPUTER NETWORKS AND ISDN SYSTEMS*, pages 161–172, 1998.

[Cha02] Soumen Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan-Kauffman, 2002.

[CM07] Kumar Chellapilla and Alexey Maykov. A taxonomy of javascript redirection spam. In *Proceedings of the 3rd international workshop on Adversarial information retrieval on the web*, AIRWeb '07, pages 81–88, New York, NY, USA, 2007. ACM.

[CN02] Junghoo Cho and Alexandros Ntoulas. Effective change detection using sampling. In *Proceedings of the 28th international conference on Very Large Data Bases*, VLDB '02, pages 514–525. VLDB Endowment, 2002.

[CSGM00] Junghoo Cho, Narayanan Shivakumar, and Hector Garcia-Molina. Finding replicated web collections. *SIGMOD Rec.*, 29(2):355–366, May 2000.

[Dan02] David R. Danielson. Web navigation and the behavioral effects of constantly visible site maps. *Interacting with Computers*, 14(5):601–618, 2002.

[DDH00] Kenneth McDonald Daryl D Harms. *The Quick Python Book*. Manning Publications, 2000.

[DG08] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51:107–113, January 2008.

[Dow01] Allen B. Downey. The structural cause of file size distributions. *SIGMETRICS Perform. Eval. Rev.*, 29(1):328–329, June 2001.

[DP08] Dawei Yin Donghua Pan, Shaogang Qiu. Web page content extraction method based on link density and statistic. In *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference*, pages 1–4, 2008.

[dSVG+99] Altigran S. da Silva, Eveline A. Veloso, Paulo B. Golghe, Berthier Ribeiro-Neto, Alberto H. F. Laender, and Nivio Ziviani. Cobweb ? a crawler for the brazilian web. *String Processing and Information Retrieval, International Symposium on*, 0:184, 1999.

[EC04] Efthimis Efthimiadis and Carlos Castillo. Charting the greek web. In *Proceedings of the Conference of the American Society for Information Science and Technology (ASIST)*, Providence, Rhode Island, USA, November 2004. American Society for Information Science and Technology.

[Fla98] David Flanagan. *JavaScript: The Definitive Guide*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 3rd edition, 1998.

[FMN03] Dennis Fetterly, Mark Manasse, and Marc Najork. On the evolution of clusters of near-duplicate web pages. In *Proceedings of the First Conference on Latin American Web Congress*, LA-WEB '03, pages 37–, Washington, DC, USA, 2003. IEEE Computer Society.

[FMN04] Dennis Fetterly, Mark Manasse, and Marc Najork. Spam, damn spam, and statistics: using statistical analysis to locate spam web pages. In *Proceedings of the 7th International Workshop on the Web and Databases: colocated with ACM SIGMOD/PODS 2004*, WebDB '04, pages 1–6, New York, NY, USA, 2004. ACM.

[FMN05] Dennis Fetterly, Mark Manasse, and Marc Najork. Detecting phrase-level duplication on the world wide web. In *In Proceedings of the 28th Annual International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 170–177. ACM Press, 2005.

[FMNW03] Dennis Fetterly, Mark Manasse, Marc Najork, and Janet Wiener. A large-scale study of the evolution of web pages. In *Proceedings of the 12th international conference on World Wide Web*, WWW '03, pages 669–678, New York, NY, USA, 2003. ACM.

[GGM04] Zoltan Gyongyi and Hector Garcia-Molina. Web spam taxonomy. Technical Report 2004-25, Stanford InfoLab, March 2004.

[GGM05] Zoltán Gyöngyi and Hector Garcia-Molina. Link spam alliances. In *Proceedings of the 31st international conference on Very large data bases*, VLDB '05, pages 517–528. VLDB Endowment, 2005.

[GGMP04] Zoltán Gyöngyi, Hector Garcia-Molina, and Jan Pedersen. Combating web spam with trustrank. In *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30*, VLDB '04, pages 576–587. VLDB Endowment, 2004.

[GJC09] Bastidas Washington Gonzalez Jesus and Abad Cristina. Implementacion y evaluacion de un detector masivo de web spam. 2009.

[GN00] G. Grefenstette and J. Nioche. Estimation of english and non-english language use on the www. In *Proceedings of Content-Based Multimedia Information Access (RIAO)*, pages 237–246, Paris, France, 2000.

[GÖ03] Sule Gündüz and M. Tamer Özsu. A poisson model for user accesses to web pages. In *ISCIS*, pages 332–339, 2003.

[GS05a] Daniel Gomes and Mário J. Silva. Characterizing a national community web. *ACM Trans. Internet Technol.*, 5(3):508–531, August 2005.

[GS05b] A. Gulli and A. Signorini. The indexable web is more than 11.5 billion pages. In *Special interest tracks and posters of the 14th international conference on World Wide Web*, WWW '05, pages 902–903, New York, NY, USA, 2005. ACM.

[HFH+09] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11:10–18, November 2009.

[Hid02] Jose María Gómez Hidalgo. Evaluating cost-sensitive unsolicited bulk email categorization, 2002.

[HLC+06] Yunhua Hu, Hang Li, Yunbo Cao, Li Teng, Dmitriy Meyerzon, and Qinghua Zheng. Automatic extraction of titles from general documents using machine learning. *Inf. Process. Manage.*, 42(5):1276–1293, September 2006.

[HMS02] Monika R. Henzinger, Rajeev Motwani, and Craig Silverstein. Challenges in web search engines. *SIGIR Forum*, 36:11–22, September 2002.

[HN99] Allan Heydon and Marc Najork. Mercator: A scalable, extensible web crawler. *World Wide Web*, 2(4):219–229, April 1999.

[Hol08] Anthony T. Holdener, III. *Ajax: the definitive guide*. O'Reilly, first edition, 2008.

[JLW98] Jr, Zhen Liu, and Richard R. Weber. Optimal robot scheduling for web search engines. *Journal of Scheduling*, (1):15–29, 1998.

[JS03] Bernard J Jansen and Amanda Spink. An analysis of web documents retrieved and viewed, 2003.

[KB03] James Kalbach and Tim Bosenick. Web page layout: A comparison between left- and right-justified site navigation menus. *J. Digit. Inf.*, 4(1), 2003.

[KC04] Rohit Khare and Doug Cutting. Nutch: A flexible and scalable open-source web search engine. Technical report, 2004.

[KG09] J Prasanna Kumar and P Govindarajulu. Duplicate and near duplicate documents detection: A review. *European Journal of Scientific Research*, 32:514–527, 2009.

[KHKHR07] Adrian Kingsley-Hughes, Kathie Kingsley-Hughes, and Daniel Read. *VBScript Programmer's Reference*. Wrox Press Ltd., Birmingham, UK, UK, 3rd edition, 2007.

[Kle99] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, September 1999.

[KMSB08] Aaron Kimball, Sierra Michels-Slettvet, and Christophe Bisciglia. Cluster computing for web-scale data processing. *SIGCSE Bull.*, 40:116–120, March 2008.

[KNA09] S. Kharazmi, A. F. Nejad, and H. Abolhassani. Freshness of web search engines: Improving performance of web search engines using data mining techniques. In *Internet Technology and Secured Transactions, 2009. ICITST 2009. International Conference for*, pages 1–7, November 2009.

[Koh95] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. pages 1137–1143. Morgan Kaufmann, 1995.

[Lew08] Dirk Lewandowski. A three-year study on the freshness of web search engine databases. *J. Inf. Sci.*, 34:817–831, December 2008.

[LG00] Steve Lawrence and C. Lee Giles. Accessibility of information on the web. *Intelligence*, 11(1):32–39, April 2000.

[LKK+09] Taehyung Lee, Jinil Kim, Jin Wook Kim, Sung-Ryul Kim, and Kunsoo Park. Detecting soft errors by redirection classification. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 1119–1120, New York, NY, USA, 2009. ACM.

[LRNdS02] Alberto H. F. Laender, Berthier Ribeiro-Neto, and Altigran S. da Silva. Debye - date extraction by example. *Data Knowl. Eng.*, 40(2):121–154, February 2002.

[MG09] João Miranda and Daniel Gomes. How are web characteristics evolving? In *Proceedings of the 20th ACM conference on Hypertext and hypermedia*, HT '09, pages 369–370, New York, NY, USA, 2009. ACM.

[Mic11] Microsoft. The official microsoft asp.net site, 2011. [Online; accessed 18-February-2011].

[MJDS07] Gurmeet Singh Manku, Arvind Jain, and Anish Das Sarma. Detecting near-duplicates for web crawling. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 141–150, New York, NY, USA, 2007. ACM.

[MKSR04] G. Mohr, M. Kimpton, M. Stack, and I. Ranitovic. Introduction to heritrix, an archival quality web crawler. In *4th International Web Archiving Workshop (IWAW04)*, 2004.

[MPZ+05] Marco Modesto, álvaro Pereira, Nivio Ziviani, Carlos Castillo, and Ricardo Baeza-Yates. Um novo retrato da Web Brasileira. In *Proceedings of XXXII SEMISH*, pages 2005–2017, São Leopoldo, Brazil, 2005.

[MS99] Christopher D. Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT Press, Cambridge, MA, USA, 1999.

[Naj] M. Najork. System and method for identifying cloaked web servers. Jul 10 2003.

[Naj09] Marc Najork. Web spam detection. In Ling Liu and M. Tamer Özsu, editors, *Encyclopedia of Database Systems*, pages 3520–3523. Springer US, 2009.

[Net01] Distribution of languages on the internet. `http://www.netz-tipp.de/languages.html`, 2001. [Online; accessed 31-January-2012].

[NM06] Alexandros Ntoulas and Mark Manasse. Detecting spam web pages through content analysis. In *In Proceedings of the World Wide Web conference*, pages 83–92. ACM Press, 2006.

[OP08] Christopher Olston and Sandeep Pandey. Recrawl scheduling based on information longevity. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 437–446, New York, NY, USA, 2008. ACM.

[Ous94] John K. Ousterhout. *Tcl and the Tk Toolkit*. Addison-Wesley Professional, 1 edition edition, 1994.

[PAC11] Víctor M. Prieto, Manuel Álvarez, and Fidel Cacheda. Evolucion de la web espanola y sus implicaciones en crawlers y buscadores. In Paolo Rosso Rafael Berlanga, editor, *II Congreso Español de Recuperación de Información*, pages 308–315. SciTePress, 2011.

[PAC13a] Víctor M. Prieto, Manuel Álvarez, and Fidel Cacheda. Analysis and detection of soft-404 pages. In *Third International Conference on Innovative Computing Technology (INTECH 2013)*, page 440. Institute of Electrical and Electronics Engineers, 2013.

[PAC13b] Víctor M. Prieto, Manuel Álvarez, and Fidel Cacheda. Saad, a content based web spam analyzer and detector. *Journal of Systems and Software*, 2013.

[PALGC11] Víctor M. Prieto, Manuel Álvarez, Rafael López-García, and Fidel Cacheda. Web oculta del lado cliente: Escala de crawling. In *X Jornadas de Ingeniería Telemática (JITEL 2011)*, pages 308–315. SciTePress, 2011.

[PALGC12a] Víctor M. Prieto, Manuel Álvarez, Rafael López-García, and Fidel Cacheda. Analysing the effectiveness of crawlers on the client-side hidden web. In Juan M. Corchado Rodríguez, Javier Bajo Pérez, Paulina Golinska, Sylvain Giroux, and Rafael Corchuelo, editors, *PAAMS (Workshops)*, volume 157 of *Advances in Soft Computing*, pages 141–148. Springer, 2012.

[PALGC12b] Víctor M. Prieto, Manuel Álvarez, Rafael López-García, and Fidel Cacheda. Analysis and detection of web spam by means of web content. In Michail Salampasis and Birger Larsen, editors, *IRFC*, volume 7356 of *Lecture Notes in Computer Science*, pages 43–57. Springer, 2012.

[PALGC12c] Víctor M. Prieto, Manuel Álvarez, Rafael López-García, and Fidel Cacheda. Architecture for a garbage-less and fresh content search engine. In *KDIR*, pages 371–374. SciTePress, 2012.

[PALGC12d] Víctor M. Prieto, Manuel Álvarez, Rafael López-García, and Fidel Cacheda. A scale for crawler effectiveness on the client-side hidden web. *Comput. Sci. Inf. Syst.*, 9(2):561–583, 2012.

[PBMW98] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[PSW08] Jakub Piskorski, Marcin Sydow, and Dawid Weiss. Exploring linguistic features for web spam detection: a preliminary study. In *Proceedings of the 4th international workshop on Adversarial information retrieval on the web*, AIRWeb '08, pages 25–28, New York, NY, USA, 2008. ACM.

[Qui93] J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[Qui96a] J. R. Quinlan. Bagging, boosting, and c4.5. In *In Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 725–730. AAAI Press, 1996.

[Qui96b] J. R. Quinlan. Improved use of continuous attributes in c4.5. *Journal of Artificial Intelligence Research*, 4:77–90, 1996.

[RBYL05] Carlos Castillo Ricardo Baeza-Yates and Vicente Lopez. Characteristics of the web of spain, 2005.

[RGM01] Sriram Raghavan and Hector Garcia-Molina. Crawling the hidden web. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, pages 129–138, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[Sch97] W. C. Schmidt. World-wide web survey research: Benefits, potential problems, and solutions. *Behavior Research Methods, Instruments, and Computers*, 29:274–279, 1997.

[SDHH98] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approach to filtering junk e-mail, 1998.

[sea12] The search engine land. http://searchengineland.com/yahoos-transition-to-bing-organic-results-complete-49228, 2012. [Online; accessed 20-October-2012].

[SPnK+00] Surasak Sanguanpong, Punpiti Piamsa-nga, Somnuk Keretho, Yuen Poovarawan, and Suthiphol Warangrit. Measuring and analysis of the thai world wide web. In *Proceeding of the Asia Pacific Advance Network*, pages 225–230, 2000.

[SY01] Torsten Suel and Jun Yuan. Compressing the graph structure of the web. In *Proceedings of the Data Compression Conference*, DCC '01, pages 213–, Washington, DC, USA, 2001. IEEE Computer Society.

[SZG07] Yang Sun, Ziming Zhuang, and C. Lee Giles. A large-scale study of robots.txt. In *In WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 1123–1124. ACM Press, 2007.

[TK98] H. M. Taylor and S. Karlin. *An Introduction To Stochastic Modeling*. Academic Press; 3 edition, 1998.

[TW03] Mike Thelwall and David Wilkinson. Graph structure in three national academic webs: power laws with anomalies. *J. Am. Soc. Inf. Sci. Technol.*, 54(8):706–712, June 2003.

[w3d11] The w3 consortium the document object model. http://www.w3.org/DOM/, 2011. [Online; accessed 18-February-2011].

[w3W11] W3Techs - World Wide Web Technology Surveys. http://w3techs.com/, 2011. [Online; accessed 22-March-2011].

[WCP07] Steve Webb, James Caverlee, and Calton Pu. Characterizing web spam using content and http session analysis, 2007.

[WCP08] Steve Webb, James Caverlee, and Calton Pu. Predicting web spam with http session information. In *Proceedings of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 339–348, New York, NY, USA, 2008. ACM.

[WD05a] Baoning Wu and Brian D. Davison. Cloaking and redirection: A preliminary study, 2005.

[WD05b] Baoning Wu and Brian D. Davison. Identifying link farm spam pages. In *Special interest tracks and posters of the 14th international conference on World Wide Web*, WWW '05, pages 820–829, New York, NY, USA, 2005. ACM.

[Web06] Steve Webb. Introducing the webb spam corpus: Using email spam to identify web spam automatically. In *In Proceedings of the 3rd Conference on Email and AntiSpam (CEAS) (Mountain View*, 2006.

[web11a] BuiltWith Web Technology Usage Statistics. http://trends.builtwith.com/, 2011. [Online; accessed 22-March-2011].

[web11b] Web Copier Pro Web page. `http://www.maximumsoft.com/products/wc_pro/overview.html`, 2011. [Online; accessed 18-February-2011].

[web11c] Webb Spam Corpus, 2011. [Online; accessed 15-September-2011].

[Wik11] Wikipedia. Applet — wikipedia, the free encyclopedia, 2011. [Online; accessed 18-February-2011].

[wik13] Usage share of operating systems. http://en.wikipedia.org/wiki/Usage_share_of_operating_systems, 2013. [Online; accessed 14-February-2013].

[Win72] R. L Winkler. *An Introduction to Bayesian Inference and Decision.* Probabilistic Publishing; 2nd edition, 1972.

[yah11a] Web Spam Detection - Resources for Research on Web Spam, 2011. [Online; accessed 15-September-2011].

[Yah11b] Yahoo! Web spam challenge, 2011. [Online; accessed 18-September-2011].