Documentos duplicados y casi duplicados en el Web: detección con técnicas de *hashing* borroso

Duplicate and near-duplicate documents in the web: detection by means of fuzzy-hash techniques

Carlos G. FIGUEROLA (1), Raquel GÓMEZ DÍAZ (2), José L. ALONSO BERROCAL (3), Ángel F. ZAZO RODRÍGUEZ (4)

Universidad de Salamanca, Facultad de Traducción y Documentación, Francisco de Vitoria, 6-16, 37008 Salamanca (España)
(1) figue@usal. (2) rgomez@usal.es. (3) berrocal@usal.es. (4) zazo@usal.es

Resumen

La detección de los duplicados en la web es importante porque permite aligerar las bases de datos documentales y mejorar la eficiencia de los motores de búsqueda y la precisión de los análisis cibermétricos y los estudios de minería web, etc. Sin embargo, las técnicas estándar de hashing aplicadas habitualmente sólo detectan duplicados exactos, a nivel de bits, mientras que muchos de los duplicados que encontramos en el mundo real no son exactamente iguales, por cambios en el formato, las cabeceras, las etiquetas META o las plantillas de visualización. La solución obvia es comparar las conversiones a texto plano de todos esos formatos, pero esas conversiones nunca son idénticas, debido al diferente tratamiento que hacen los conversores de los diversos elementos de formato. Se presenta la posibilidad de utilizar fuzzy-hashing para producir huellas digitales de dos documentos que se pueden comparar para proporcionar una estimación de la cercanía o distancia entre los dos documentos. Basado en el concepto de rolling-hash, el fuzzy hashing se utiliza con éxito en tareas de seguridad informática como identificación de programas maliciosos, correo basura, detección de virus, etc. Hemos añadido capacidades de fuzzy-hashing a un crawler y hemos llevado a cabo diversas pruebas que nos han permitido estimar umbrales útiles de similitud o parecido entre documentos, así como obtener datos interesantes sobre la cantidad y distribución de documentos duplicados en servidores web.

Palabras clave: World Wide Web. Detección de duplicados. Fuzzy hashing

Abstract

The detection of duplicates in the web is important because it allows to lighten databases and improve the efficiency of information retrieval engines and the precision of cybermetric analysis, web mining studies, etc. Standard hash techniques used to detect these duplicates only detect exact ones, at the bit level. However, many of the duplicates found in the real world are not exactly alike and have the same content, but different formats, headers, meta tags or style sheets. The obvious solution is to compare plain text conversions of all these formats, but these conversions are never identical, because of the different treatments that the converters give to the various formatting elements (treatment of textual characters, diacritics, spacing, paragraphs…). In this article, we introduce the possibility of using fuzzy-hashing to produce fingerprints of files (or documents, etc..) that can be compared to estimate the closeness or distance between two files, documents, etc. Based on the concept of "rolling hash", the fuzzy hashing has been used successfully in computer security tasks, such as identifying malware, spam, virus scanning, etc. We have added capabilities of fuzzy hashing to a slight crawler and have made several tests in a heterogeneous network domain, consisting of multiple servers with different software, static and dynamic pages, etc. These tests allowed us to measure similarity thresholds and to obtain useful data about the quantity and distribution of duplicate documents on web servers.

Keywords: World Wide Web. Duplicate detection. Fuzzy hashing.

1. Introducción

El web es el depósito de documentos de mayor tamaño disponible. Incluso de una manera intuitiva es fácil encontrar abundante información replicada en cualquier navegación que se efectúe. Ya a finales de los años 90, Bharat y Broder (Bharat ,1999) citaban estudios en los que se muestra que más del 30 % de las páginas web exploradas por los *crawlers* eran duplicados. Aunque en su estudio identificaron sólo el 9.4 % de las páginas duplicadas, no es menos cierto que estos autores se centraron solamente en la duplicación por copia explícita o *mirroring*. En cualquier caso, se trata de cantidades importantes que hacían pensar ya entonces en la existencia de un alto nivel de replicación de contenidos web, por diversas razones.

La duplicación de documentos entorpece el trabajo de los *crawlers* y ralentiza la indización de documentos, además de ser una dificultad

añadida para los usuarios que necesitan revisar los documentos recuperados tras una búsqueda. Chowdury y otros (Chowdury, 2000) han apuntado, además, que el problema de la duplicación de documentos incide no sólo en la velocidad de la recuperación, sino también en los cálculos de los pesos de los términos, que se basan, entre otras cosas, en sus frecuencias de aparición.

En muchos casos, sin embargo, no se trata de duplicados exactos, sino de casi duplicados: documentos que tienen prácticamente el mismo contenido, aunque difieren al algunos detalles. Yerra y Ng (Yerra, 2005) citan algunos casos, a modo de ejemplo: documentos en diferentes versiones, documentos pequeños ensamblados en un documento grande, y lo contrario: documentos grandes divididos en partes más pequeñas; diferentes visualizaciones del mismo contenido… A todo ello se pueden añadir algunas causas más: cabeceras específicas introducidas por servidores o aplicaciones web a documentos idénticos, contadores de visitas, documentos que incorporan la fecha actual, su propio URL, trayectorias relativas, etc.

Debido a ello, es importante contar con técnicas y herramientas útiles que permitan detectar documentos duplicados y casi duplicados; esta necesidad se plantea en diversos contextos, y también en el que a nosotros nos interesa: el de la recuperación de la información en el web.

Este trabajo pretende mostrar algunos resultados preliminares en la aplicación en este tipo de tareas de técnicas conocidas como *fuzzy-hash*. Este trabajo está organizado como sigue: en la sección 2 se hace un breve repaso a las técnicas más usuales para medir el parecido o similitud entre documentos, y su aplicación en la detección de documentos casi duplicados. En la sección 3 se describe el *fuzzy-hash* o *Context Triggered Piece Hashing*. En la sección 4 se exponen las pruebas experimentales llevadas a cabo y se comentan sus resultados. Finalmente, se ofrecen unas conclusiones y líneas de trabajo futuro.

2. Medidas de similitud entre documentos

Hay diversos sistemas que permiten estimar el parecido o similitud entre dos documentos. Varios de ellos derivan del cálculo de similitud entre vectores, aplicable cuando los documentos son representados mediante vectores: coseno, Dice, Jaccard y otros (Tan, 2006). Pero no son utilizables en este caso; fueron diseñados para estimar similitud semántica; el propio modelo vectorial en que se apoya su uso parte de la independencia entre los términos, sin tomar en cuenta la posición de unos respecto a otros. Y, en todo caso, su aplicación tiene unos requerimientos de procesamiento inasumibles en este contexto (Chowdury, 2004).

Otros sistemas derivan de lo que se conoce como distancia de edición, es decir, el número de operaciones necesarias para transformar una secuencia de caracteres en otra (Navarro, 2001). Esto incluye medidas de similitud entre secuencias de caracteres como la de Hamming (Hamming, 1950), Lenvensthein (Levenshtein, 1966; Soukoreff, 2001) o Damerau (Damerau, 1964), entre otras varias. Su principal problema es que están concebidas para secuencias de caracteres cortas; uno de sus usos más frecuentes, por ejemplo, es la corrección de errores ortotipográficos, sugiriendo palabras cercanas. Aplicadas con cadenas muy largas, como pueden ser los documentos completos, resultan inaplicables, pues necesitan demasiado tiempo.

La solución más empleada a la hora de detectar documentos duplicados es el *hashing*, esto es, la obtención de una huella digital que, gracias a los algoritmos utilizados, es de pequeño tamaño, pero asegura con bastante fiabilidad que no se producen dos huellas iguales para documentos que son distintos. El *hashing* tienen diversas aplicaciones, desde facilitar búsquedas rápidas en estructuras que utilizan como clave o referencia el *hash* de los datos almacenados, hasta su uso criptográfico, por ejemplo, en aplicaciones de firma digital o como garantía de no alteración de un documento. Algunos de los algoritmos más conocidos son MD5 o SHA; MD5 es utilizado con frecuencia en las descargas de software libre para comprobación de la integridad de los ficheros originales y es fácil encontrar utilidades en los sistemas operativos y en los lenguajes de programación que permiten obtener huellas mediante estos algoritmos.

Sin embargo, el *hashing* convencional, que funciona muy bien para los objetivos para los que fue ideado, tiene inconvenientes en la detección de documentos duplicados. En efecto, la menor diferencia entre dos documentos, incluso a nivel de bits, produce huellas completamente diferentes, sin que esta diferencia sea proporcional a la diferencia real entre tales documentos. Así, un algoritmo como MD5 certifica de forma eficiente y muy segura que dos documentos son idénticos; pero no es válido para detectar documentos que, pudiendo considerarse iguales en la práctica, contengan la más mínima diferencia.

3. Fuzzy-hashing

Fuzzy-hash o *Context Triggered Piece Hashing* (CTPH) se basa en el trabajo de A. Tridgell (2001), que, después de utilizar técnicas de *hash* en el desarrollo de *rsync* (una conocida aplicación para la transferencia incremental de ficheros, (http://rsync.samba.org/) aplicó *fuzzy hash* en otra aplicación (a la que llamó *spamsum*) dedicada a detectar mensajes basura en el correo electrónico.

El *fuzzy-hash* se apoya en lo que se conoce como *rolling hash*. Supongamos una ventana de *n* caracteres de tamaño que se va desplazando a lo largo de un documento, cada vez un carácter. Para cada posición se calcula una huella convencional, que se recalcula de manera muy rápida para la siguiente posición de la ventana, hasta llegar al final del documento. *Rolling hash* se aplica en en la búsqueda de subcadenas, entre otras cosas. El *fuzzy-hash* aplica un *rolling hash* que avanza a lo largo de un documento hasta llegar a determinados puntos, conocidos como *trigger points*. Cuando alcanza un *trigger point*, el *rolling hash* calculado hasta entonces es almacenado y se reinicia, hasta el siguiente *trigger point*, y así sucesivamente hasta el final del documento. Tanto el tamaño *n* de la ventana del *rolling hash* como los *trigger points* se obtienen por diversos procedimientos en función del tamaño del documento.

Al final del proceso se tienen una serie de huellas correspondientes a otros tantos segmentos del documento original. Es posible entonces obtener una signatura corta de cada huella y concatenarlas en una secuencia de caracteres. Si modificamos el documento original, alguno o algunos de los segmentos (los correspondientes a la modificación) producirán *rolling hashes* diferentes, pero no el resto. Así la secuencia final de caracteres obtenida diferirá en los correspondientes a los segmentos afectados por la modificación, pero no en el resto. De esta manera, la cantidad de caracteres diferentes en el *fuzzy-hash* final es proporcional a la envergadura de las modificaciones introducidas en el documento (Kornblum, 2010).

El fuzzy hashing es muy utilizado en tareas relacionadas con la seguridad informática (análisis forense, detección de virus polimórficos, detección de programas maliciosos, etc.) (Kornblum, 2006).

4. Pruebas efectuadas

Hemos efectuado diversas pruebas tendentes a evaluar la utilidad del *fuzzy hashing* en la detección de documentos duplicados en el web. El contexto de tales pruebas es la Recuperación de Información en el web. En este contexto, un *crawler* recorre la red recopilando e indizando documentos. Básicamente, el *crawler* explora una página web, la indiza y también extrae los enlaces que ésta contiene, que son almacenados. Una vez indizada una página, el *crawler* obtiene una nueva dirección a explorar de entre las almacenadas. Naturalmente, el *crawler* se cuida de no explorar la misma dirección, es decir, no almacena direcciones que ya han sido visitadas, o que ya están almacenadas previamente. Naturalmente también, es posible limitar los recorridos del *crawler* a URLs que pertenezcan a determinados dominios o que reúnan determinadas características, como es posible también circunscribir la indización a documentos en determinados formatos, etc. (Figuerola, 2006).

En las pruebas hemos utilizado nuestro propio *crawler*, diseñado para formar parte de nuestro motor de búsqueda *VII* (Figuerola, 2010). Le hemos añadido la capacidad de detectar duplicados exactos, mediante un algoritmo MD5, y también la de detectar documentos muy similares mediante *fuzzy hash*, utilizando el módulo *ssdeep* para el lenguaje de programación *python* (Milemko, 2010), una adaptación para este lenguaje basado en la librería *ssdeep* de Kornblum (Kornblum, 2010).

En todos los casos, tanto el hash MD5 como el *fuzzy hash* se obtienen no de la página o documento original, sino de su conversión a texto plano. De esta manera obviamos diferencias derivadas simplemente del formato. De otro lado, como se ha dicho antes, el *crawler* sólo recorre URLs diferentes, es decir, nos referimos a documentos duplicados o casi duplicados con direcciones de red diferentes.

Al tratarse de pruebas preliminares hemos efectuado exploraciones en el interior de determinados servidores web o de determinados subdominios. Los duplicados encontrados, en consecuencia, están siempre en el mismo servidor o, en todo caso, dentro del mismo subdominio.

Tras diversas pruebas, establecimos un umbral de similitud en 0.9 para el *fuzzy hash*, de manera que los documentos con un resultado superior a este umbral se han considerado duplicados. Algunos resultados están representados en los gráficos adjuntos.

Lo primero que hay que constatar es la abundancia de documentos duplicados, en todos los casos. Buena parte de ellos son duplicados exactos (su conversión a texto plano), detectados con MD5. Pero, de forma variable, según los casos, hay una cantidad importante de do-

cumentos no detectados como duplicados por MD5, pero que gracias al *fuzzy hashing* pueden considerarse como duplicados a efectos prácticos.

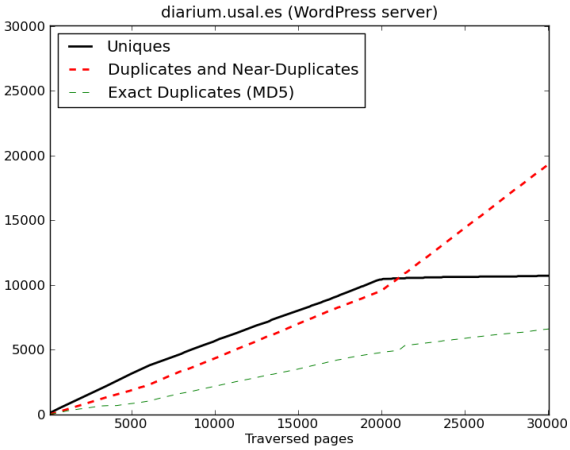

Figura 1. Resultados en el servidor WordPress

En muchos casos, la cantidad de duplicados es incluso superior a la de documentos únicos. En algunos gráficos se observa claramente cómo, según avanza el recorrido del *crawler* a través de un servidor, el número de páginas o documentos únicos no duplicados crece hasta alcanzar un techo, que no se sobrepasa o se sobrepasa con escasa pendiente. El número de duplicados y el de casi duplicados también crece según se avanza en el recorrido por el web, pero no se detiene y supera, antes o después, a los documentos únicos.

En otras palabras, el *crawler* obtiene todos los documentos únicos, pero no se detiene, pues sigue obteniendo URLs que, siendo nuevos y diferentes de los ya explorados, conducen a páginas o documentos que son duplicados de otros ya visitados.

Esto significa que, con bastante aproximación, podría suspenderse el recorrido del *crawler* bastante antes de haber explorado la totalidad de direcciones recopiladas, en la confianza de que, a partir de determinado punto, pocas o ninguna página con contenido nuevo va a ser recorrida.

De otro lado, se han efectuado pruebas también con un subdominio completo; este subdominio es heterogéneo, en el sentido de que está integrado por varios servidores (unos 200) de diverso tamaño, unos con páginas estáticas y otros dinámicos, gestionados por una gran variedad de gestores de contenidos o aplicaciones web ad-hoc. El tamaño, en número total de páginas de ese subdominio es difícil de precisar; Google arroja un resultado de 1.500.000 páginas, aunque es sabido que las cifras que proporcionan los buscadores deben observarse con muchas reservas (Bar-Ilan, 2005). Lo que más llama la atención es que, después de explorar las primeras 73,000 páginas prácticamente no se encuentran ya documentos únicos; y que, de éstos, sólo se obtienen algo más de 46,000. La cantidad de documentos duplicados exactos (detectados mediante MD5) es notable, pero se estabiliza también a partir de un determinado punto. Lo que hace aumentar a partir de ese punto la cantidad de duplicados son los documentos casi duplicados, detectados a través de *fuzzy hash*.

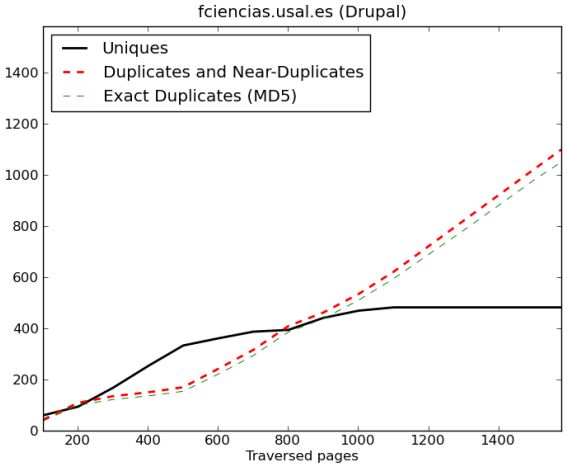

Figura 2. Resultados en el servidor Drupal

En este punto pueden plantearse varios interrogantes. En primer lugar, las causas de la abundante duplicación de documentos; teniendo en cuenta que el *crawler* sólo sigue URLs únicos. Parte de tales duplicaciones pudiera venir de redirecciones y alias en los nombres de los servidores. Pero, a la espera de confirmar esta hipótesis con datos rigurosos, parece que esta causa sólo podría explicar una parte de las duplicaciones exactas. Otra parte puede tener origen en la forma en que los gestores de contenidos y aplicaciones web generan enlaces hacia sus componentes, que podrían tomar formas diferentes en lo que se refiere al URL, pero que apuntan al mismo sitio. En este sentido, es interesante preguntarse por diferentes comportamientos en función de los programas concretos utilizados en cada sitio web; podemos suponer que, tal vez, unos gestores de contenido web generan más duplicados que otros. Es probable, también, que determinados usos del web generen un número mayor de replicaciones, ya sean exactas o ya sean generando ver-

siones del mismo documento. Esto pudiera producirse con sitios web con orientación social (web 2.0); y es probable también que esta clase de sitios tengan mayor vinculación con determinados gestores de contenido, orientados, precisamente, a la Web 2.0

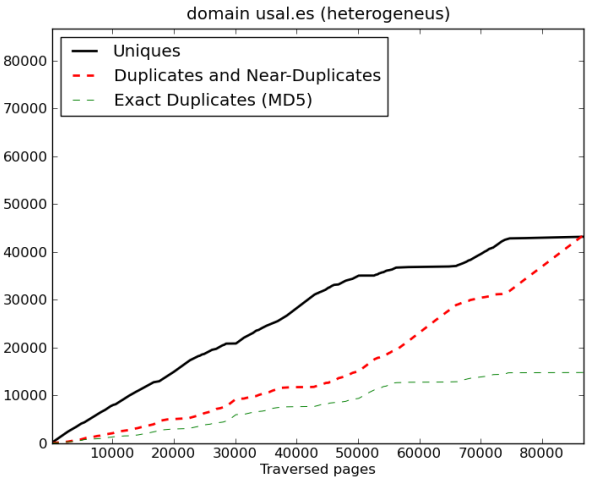

Figura 3. Resultados en el dominio usal.es

¿La manera de recorrer el web por parte del *crawler* influye en el hecho de encontrar más o menos duplicados? ¿Podemos descubrir caminos que nos hagan explorar antes los documentos únicos que contiene un sitio o un dominio web?

De otro lado, si, como parece, hay un nivel importante de documentos duplicados en el web, este hecho debiera afectar a los resultados de los buscadores web. De hecho, muchos de ellos aplican diversos métodos para detectar y eliminar duplicados y casi duplicados de los resultados de las búsquedas, con más o menos éxito.

Éste es el caso de Google, que tiene incluso una patente sobre un sistema para detectar duplicados (Pugh, 2003), y que aplica tras una búsqueda, antes de presentar al usuario la lista de documentos encontrados.

Yahoo también dispone de un sistema de detección de duplicados, como cabe desprender de las posibilidades que su API ofrece a los programadores, aunque no está, hasta donde nosotros sabemos, debidamente documentado.

Por lo que se refiere a Google, hemos efectuado algunas pruebas consistentes en realizar 50 consultas o búsquedas estándar. Se trata, en concreto, de las consultas propuestas en la tarea de recuperación web del TREC 2009 (Clarke, 2009). Para cada una de esas consultas hemos examinado los primeros 750 documentos recuperados, aplicando las técnicas antes descritas de *fuzzy-hash* para detectar duplicados y casi duplicados.

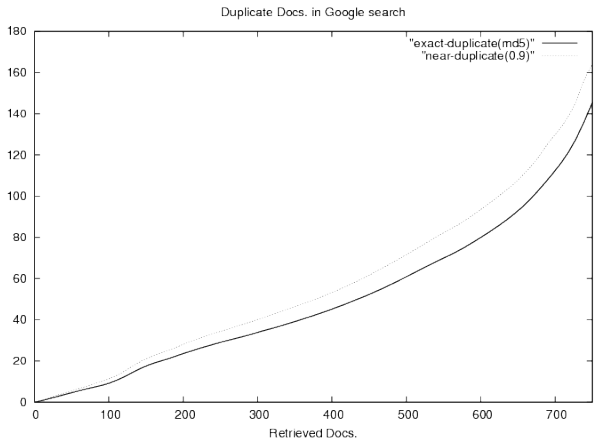

Figura 4. Resultados en Google

Es importante tener en cuenta que estas técnicas las hemos aplicado en adición a las propias que Google aplica por su cuenta. Los resultados quedan reflejados en el gráfico adjunto, que muestra un nivel notable de documentos duplicados, aún después de haber eliminado Google los que él mismo ha sido capaz de detectar.

5. Conclusiones

Hemos mostrado una técnica, conocida como *fuzzy-hashing* que permite detectar documentos duplicados y casi duplicados. Esta técnica ha sido aplicada implementándola en un *crawler* que navega autónomalmente por el web recorriendo (y, eventualmente, indizando) documentos.

El *fuzzy-hashing* es una huella criptográfica real, similar a otros sistemas de *hashing*: se calcula de una forma muy rápida y el riesgo de colisiones (dos documentos diferentes que producen la misma huella) es muy bajo. Sin embargo, la diferencia entre los *fuzzy hash* de dos documentos es proporcional a la diferencia entre dichos documentos. Esta características nos permite detectar documentos casi duplicados fácilmente.

Los resultados de las pruebas efectuadas sugieren que el web contiene una cantidad significativa de documentos duplicados o casi duplicados. Sugieren también que, para un determinado nivel de páginas exploradas por un *crawler* en un sitio web, se alcanza un punto en el que ya no se encuentran páginas nuevas que no hayan sido exploradas previamente.

Referencias

Bar-Ilan, J. (2005). Expectations versus reality sarch engine features needed for web research at mid 2005. // Cybermetrics 9:1 (2005).

Bharat, K.; Broder, A. (1999). Mirror, mirror on the web: A study of host pairs with replicated con-tent. // Computer Networks. 31:11-16 (1999) 1579-1590.

Chowdhury, A. (2004). Duplicate data detection. http://gogamza.mireene.co.kr/wpcontent/uploads/1/XbsrPeUgh6.pdf (2011-01-13).

Chowdhury, A.; Frieder, O.; Grossman, D.; McCabe, M. (2002). Collection statistics for fast duplicate document detection. // ACM Transactions on In-formation Systems (TOIS) 20:2, 171-191 (2002) http://citeseerx.ist.psu.edu/viewdoc/download/doi=10.1.1.5.373&rep=rep1\&type=pdf (2011-01-13).

Clarke C.L.; Crasswell, N.; Soboroff, I. (2009). Overview of the TREC 2009 Web Track // Proceedings of the 18th Text REtrieval Conference, Gaithersburg, Maryland, 2009. 1-9

Damerau, F. (1964). A technique for computer detec-tion and correction of spelling errors. // Communications of the ACM. 3, 171-176.

Figuerola, C. G.; Alonso Berrocal, J. L.; Zazo Rodríguez, A. F.; Rodriguez Vázquez de Aldana, E. (2006). Diseño de spiders. // Tech. Rep. DPTOIA-IT-2006-002 (2006).

Figuerola, C. G.; Gómez Díaz, R.; Alonso Berrocal, J. L.; Zazo Rodríguez, A. F. (2010). Proyecto 7: un motor de recuperación web colaborativo. // Scire: Representación y Organización del Conocimiento. 16, 53-60 (2010).

Hamming, R. (1950). Error detecting and error correcting codes. // Bell System Technical Journal. 29:2, 147-160.

Kornblum, J. (2006). Identifying almost identical files using context triggered piecewise hashing. // Digital investigation. 3, 91-97.

Kornblum, J. (2010). Beyond fuzzy hash. // US Digital Forensic and Incident Response Summit 2010 (2010). http://computer-foren-sics.sans.org/community/summits/2010/files/19-beyond-fuzzy-hashing-kornblum.pdf (2011-01-13).

Kornblum, J. (2010). Fuzzy hashing and sseep. http://ssdeep.sourceforge.net/ (2011-01-13).

Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions, and reversals. // Soviet Physics Doklady. 10:8, 707-710.

Milenko, D. (2010). ssdeep 2.5. python wrapper for ssdeep library. http://pypi.python.org/pypi/ssdeep (2011-01-13).

Navarro, G. (2001). A guided tour to approximate string matching. // ACM computing surveys (CSUR). 33:1, 31-88.

Pugh, W. Y; Henzinger, M.H. (2003). Detecting Duplicate and Near Duplicate Files. United Sates Patent 6.658.423.

Soukoreff, R., MacKenzie, I. (2001). Measuring errors in text entry tasks: an application of the levenshtein string distance statistic. // CHI'01 ex-tended abstracts on Human factors in computing systems. 319-320. ACM.

Tan, P.; Steinbach, M.; Kumar, V.; et al. (2006). Introduction to data mining. Pearson Addison Wesley: Boston (2006).

Tridgell, A. (2002). Spamsum overview and code. http://samba.org/ftp/unpacked/junkcode/spamsum (2011-01-13).

Tridgell, A., Mackerras, P.(2004). The rsync algorithm. http://dspace-prod1.anu.edu.au/bitstream/1885/40765/2/TR-CS-96-05.pdf (2011-01-13).

Yahoo! (2011). Yahoo Developer Network. http://developer.yahoo.com (2011-01-13).

Yerra, R.; Ng, Y. (2005). Detecting similar html documents using a fuzzy set information retrieval approach. // 2005 IEEE International Conference on Granular Computing. 2, 693-699. IEEE (2005).