US006286006B1

(12) **United States Patent** (10) **Patent No.:** **US 6,286,006 B1**
Bharat et al. (45) **Date of Patent:** **Sep. 4, 2001**

(54) **METHOD AND APPARATUS FOR FINDING MIRRORED HOSTS BY ANALYZING URLS**

(75) Inventors: **Krishna A. Bharat**, Santa Clara; **Andrei Broder**, Menlo Park; **Steven C. Glassman**, Mountain View; **Jeffrey Dean**; **Monika R. Henzinger**, both of Menlo Park, all of CA (US)

(73) Assignee: **Alta Vista Company**, Palo Alto, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/307,320**

(22) Filed: **May 7, 1999**

(51) **Int. Cl.**^{7} **G06F 17/30**
(52) **U.S. Cl.** **707/100**; 707/5; 707/10; 709/203; 709/230
(58) **Field of Search** 707/5, 100, 10; 709/230, 203

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,895,470	4/1999	Pirolli et al.	707/102
5,909,677	* 6/1999	Broder et al.	707/3
5,935,207	* 8/1999	Logue et al.	709/219
5,978,797	* 11/1999	Yianilos	707/3
5,991,714	* 11/1999	Shaner	704/9
6,105,019	* 8/2000	Burrows	707/2
6,119,124	* 9/2000	Broder et al.	707/103
6,125,395	* 9/2000	Rosenberg et al.	709/228
6,138,113	* 10/2000	Dean et al.	707/2
6,154,744	* 11/2000	Kenner et al.	707/10
6,167,398	* 12/2000	Wyard et al.	707/5

FOREIGN PATENT DOCUMENTS

WO-00/68838 * 11/2000 (WO) G06F/17/30

OTHER PUBLICATIONS

Roy E. Kimbrell, "Searching for Text? Send an N$_{13}$gram", pp. 3–18, 1988, Mcgraw–Hill, Inc.*

Andrei Z. Broder et al., SRC Technical Note "Syntactic Clustering of the Web", Jul. 25, 1997, 7–page document.

Gerard Salton, et al., "Term–Weighting Approaches in Automatic Text Retrieval, " Information Processing Management, vol. 24, No. 5, Copyright© 1988, Nov. 1987, 11–page document.

Narayanan Shivakumar et al., "Finding near–replicas of documents on the web," Department of Computer Science, Stanford, CA 94305, three–page document.

Karen Sparck Jones and Peter Willett, "Readings in Information Retrieval, " Morgan Kaufman Publishers, Jul. 1997, Copyright© 1997, 589–page book.

Skivakumar, N., Garica–Molina, H., "Finding near–replicas of documents on the web", Proceeding of Workshop on Web Databases, Mar. 1998.

Broder, et al., "Syntactic Clustering of the Web", Computer Networks and ISDN Systems, NL, North Holland Publishing, Amsterdam, vol. 29, No. 8–13, Sep. 1, 1997.

Salton, G., et al., "Term–Weighting Approaches in Automatic Text Retrieval", Information Processing & Management, Gb, vol. 24, No. 5, 1998, pp. 513–523.
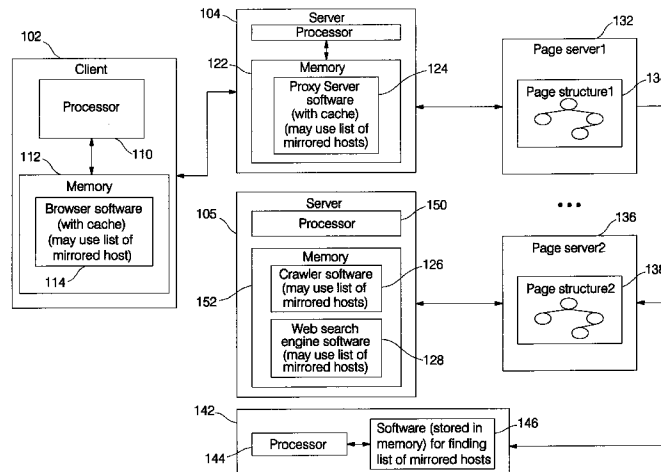
* cited by examiner

Primary Examiner—John Breene
Assistant Examiner—Debbie M Le
(74) Attorney, Agent, or Firm—Fenwick & West LLP

(57) **ABSTRACT**

A method and apparatus that detects mirrored host pairs using information about a large set of pages, including URLs. The identities of the detected mirrored hosts are then saved so that browsers, crawlers, proxy servers, or the like can correctly identify mirrored web sites. The described embodiments of the present invention look at the URLs of pages hosts to determine whether the hosts are potentially mirrored.

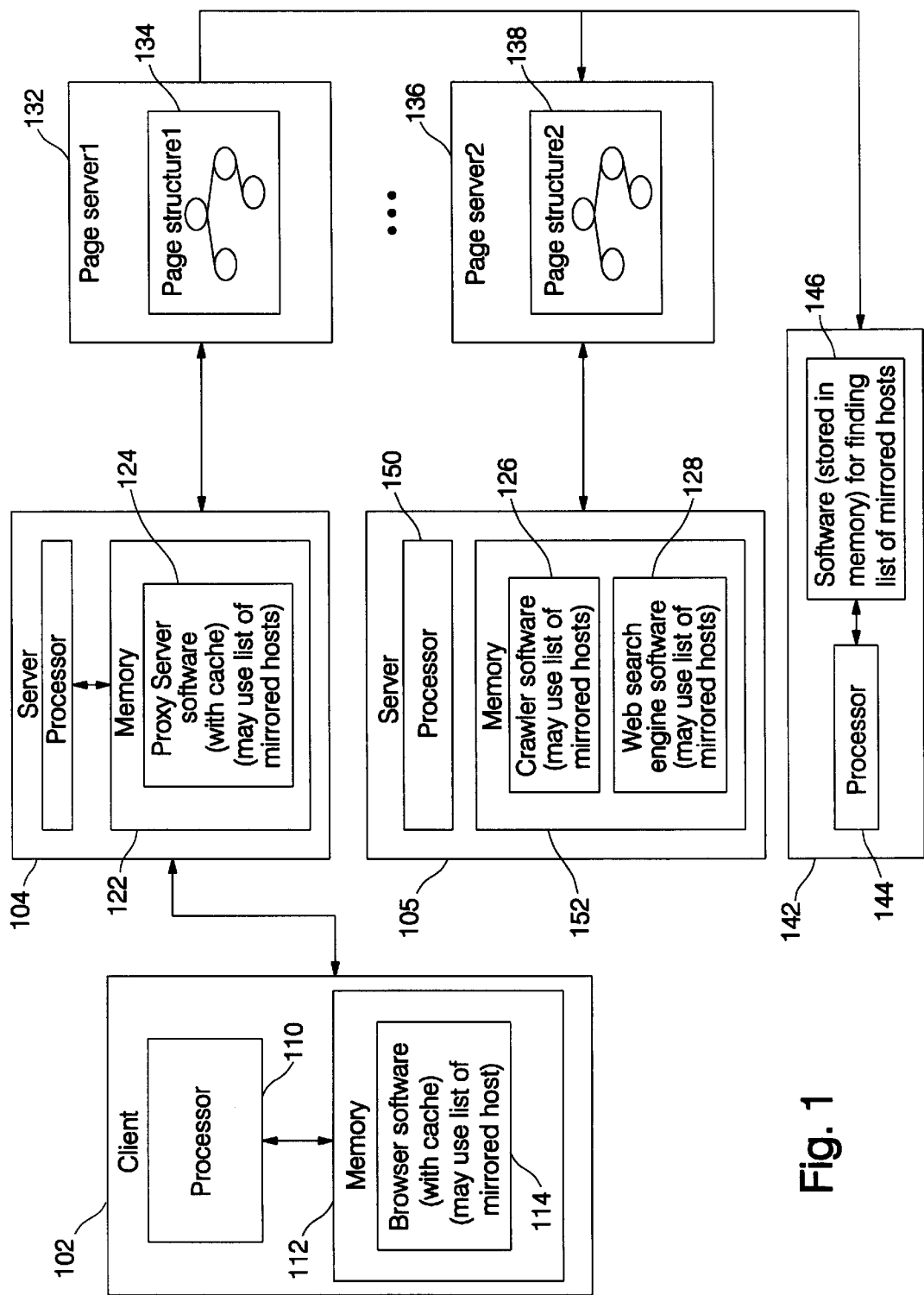**21 Claims, 29 Drawing Sheets**

Fig. 1

www.host2.edu

root

a

b

c

d

www.host2.edu/a
www.host2.edu/b
www.host2.edu/b/c
www.host2.edu/b/d

www.host1.org

root

a

b

c

d

www.host1.org/a
www.host1.org/b
www.host1.org/b/c
www.host1.org/b/d

Fig. 2(a)

www.host1.org

www.host1.org/a
www.host1.org/b
www.host1.org/b/c
www.host1.org/b/d

www.host2.edu

www.host2.edu/a
www.host2.edu/b
www.host2.edu/b/c
www.host2.edu/b/d
www.host2.edu/b/e

Fig. 2(b)

```
                    ┌─────────────┐
                    │    Start    │
                    └──────┬──────┘
                           │
                           ▼
              ┌──────────────────────┐
              │   Put address of a   │
     302 ～    │   web page in the    │
              │  uncrawled address   │
              │        queue         │
              └──────────┬───────────┘
                         │
                         ▼
                        ╱╲
          304 ～       ╱    ╲
     ┌─────┐         ╱Addresses in╲
     │ End │◄───────╱ uncrawled address╲
     └─────┘    N   ╲    queue?    ╱
                     ╲          ╱
                      ╲        ╱
                       ╲  │  ╱
                        ╲ │ ╱
                          │ Y
                          ▼
              ┌──────────────────────┐
              │   Crawl down next    │
              │ uncrawled address to a│
     306 ～    │    page.  Collect    │
              │  information about this│
              │        page.         │
              └──────────┬───────────┘
                         │
                         ▼
     308 ～   ┌──────────────────────┐
              │  Collect all addresses│
              │ that are linked to on that│
              │  page;  Add these to the│
              │  queue (unless they are│
              │   mirror pages and   │
              │   already crawled)   │
              └──────────────────────┘
```

Fig. 3(a)
Exemplary
Crawler Software
using List of
Mirrored hosts

Start

322～ Receive address of page to fetch (received from client system)

326～ Addressed page (or its mirror) in the proxy cache?

328～ Y → Retrieve page (or cached mirrored page) from cache; return cached page (or cached mirrored page) to client

324～ N → Fetch and cache addressed page; return fetched page to client

End

Fig. 3(b)

Exemplary Proxy Server Software using List of Mirrored hosts

Start

342 ~ Receive address of page to fetch (for example, entered by user or from a clicked link)

346 ~ Addressed page (or its mirror) in the browser cache?

N → 344 ~ Fetch and cache addressed page; Display fetched page

Y → 348 ~ Retrieve page (or cached mirrored page) from cache; display cached page (or cached mirrored page); Display retrieved page

End

Fig. 3(c)
Exemplary
Browser Software
using List of
Mirrored hosts

Start

362 — Receive search term(s) and review indices to identify potential pages having these search terms

364 — Eliminate mirrors (do not show duplicate hosts in search results)

366 — List alternate sources for certain hosts (i.e., list mirrors as being mirrors)

368 — List mirror hosts as "alternative hosts" for broken links

End
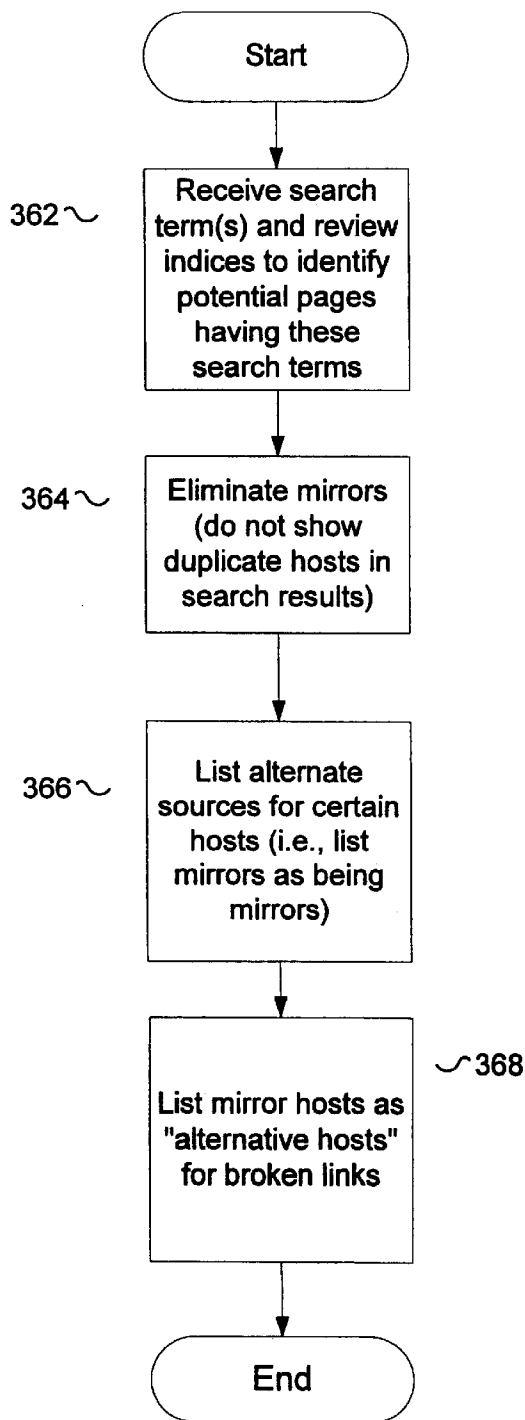
Fig. 3(d)
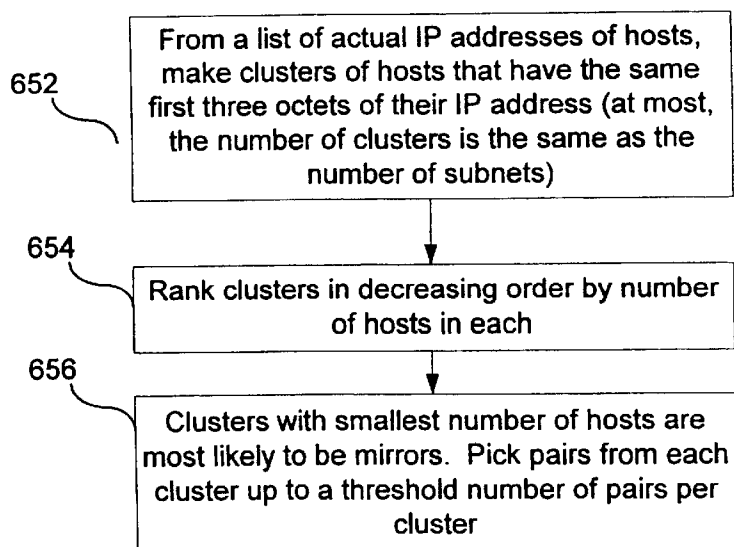Exemplary
Search Engine
Software using
List of Mirrored
hosts

402

Stage I (Candidate Pair Detection):

Provide information about a large set of pages, including URLs, host IP addresses, and/or connectivity information, such as links at the page and/or host level. This could be from a web crawl. Sample a subset of this information and compute a similarity between members of the subset. Output lists of host pairs <host1, host2> that are potential mirrors, ordered by likelihood that the hosts in the host pairs are mirrors of each other.

Uses one or more of methods such as the following to determine host pairs:

IP address-based;
URL-based
page connectivity-based;
host connectivity-based

404

Stage II (host Pair classification):

Process the list of candidate host pairs. Access some of the hosts of the host pairs to determine whether the hosts are indeed mirrors

Fig. 4

XXX.XXX.XXX.XXX

First octet   Second octet   Third octet   Fourth octet

Subnet

Fig. 5

Format of an IP Address of a host

602

From a list of actual IP addresses of hosts, make clusters of hosts that have the same IP address (at most, the number of clusters is the same as the number of IP addresses)

604

Rank clusters in increasing order by number of hosts in each

606

Clusters with smallest number of hosts are most likely to be mirrors. Pick pairs from each cluster up to a threshold number of pairs per cluster

Fig. 6(a)
A First IP Address-based Method
(Compare entire IP address of hosts)
(Stage I)

652

From a list of actual IP addresses of hosts, make clusters of hosts that have the same first three octets of their IP address (at most, the number of clusters is the same as the number of subnets)

654

Rank clusters in decreasing order by number of hosts in each

656

Clusters with smallest number of hosts are most likely to be mirrors. Pick pairs from each cluster up to a threshold number of pairs per cluster

Fig. 6(b)
A Second IP Address-based Method
(Compare three octets of IP addresses of hosts)
(Stage I)

http://www.host1.com/ABC/xy/z.html

Access method

host name

Path
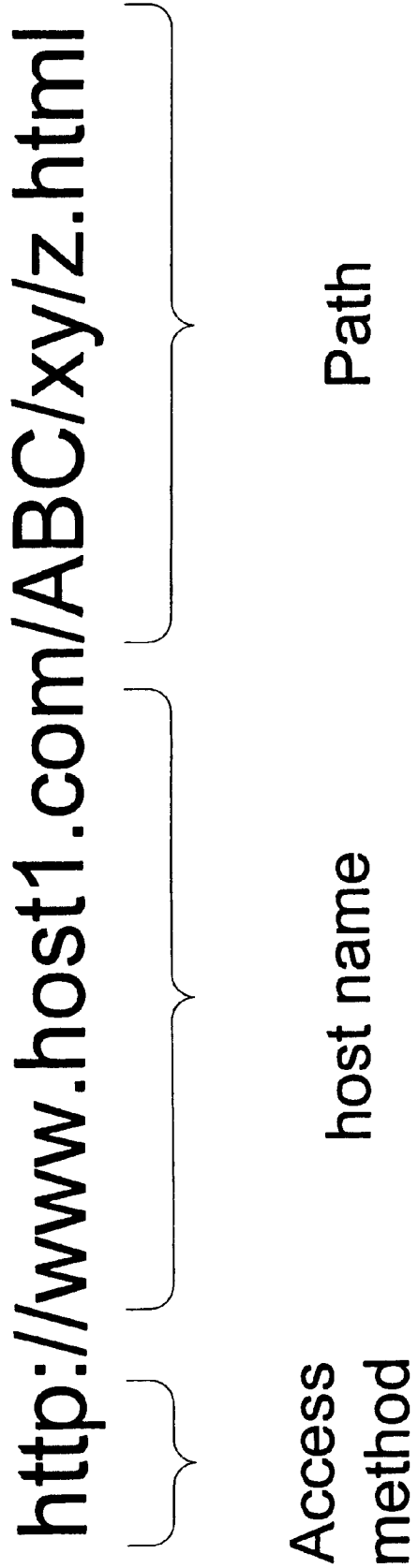
Fig. 7

Example URL of a Page

802

Convert each URL in the list of URLs to tokens by:
1) converting URL to lower case
2) treat any non-alphabetical characters (including numbers) as word breaks

804

Generate "bi-grams" by treating each contiguous pair of tokens in a URL as a "term." Add path depth information to terms

806

Sort list of <term, host> tuples by terms

808

Determine weights of sorted terms in inverse proportion to frequency;

Determine similarity score for every pair of hosts host1 and host2 that share a term. Normalize similarity score

810

Output list of prospective mirrored hosts in decreasing order of probability that the hosts are mirrors, in accordance with their similarity scores

Fig. 8(a)

A URL String-based
Method
(Stage I)

<u>URL</u>: http://a.b.com/cellblock16/inmates/me/
        personal/foo.html

<u>Yields bi-grams:</u> {(cellblock,inmates,0) (inmates,me,1)
(me,personal,2) (personal,foo,3) (foo,html,4)}

<u>Yields host tuples</u> <term,host>:
<(cellblock,inmates,0) ,a.b.com>
<(inmates,me,1), a.b.com>
<(me,personal,2), a.b.com>
<(personal,foo,3), a.b.com>
<(foo,html,4), a.b.com>

<u>URL</u>: http://x.y.com/cellblock17/inmates/me/
        personal/foo.html

<u>Yields bi-grams:</u> {(cellblock,inmates,0) (inmates,me,1)
(me,personal,2) (personal,foo,3) (foo,html,4)}

<u>Yields host tuples</u> <term,host>:
<(cellblock,inmates,0) ,x.y.com>
<(inmates,me,1), x.y.com>
<(me,personal,2), x.y.com>
<(personal,foo,3), x.y.com>
<(foo,html,4), x.y.com>

Fig. 8(b)

<(cellblock,inmates,0) ,a.b.com>
<(cellblock,inmates,0) ,x.y.com>   }   H1 and H2

<(inmates,me,1), a.b.com>
<(inmates,me,1), x.y.com>   }   H1 and H2

<(me,personal,2), a.b.com>
<(me,personal,2), x.y.com>   }   H1 and H2

<(personal,foo,3), a.b.com>
<(personal,foo,3), x.y.com>   }   H1 and H2

<(foo,html,4), a.b.com>
<(foo,html,4), x.y.com>   }   H1 and H2

Fig. 8(c)

host Address = www.uvw.edu

Page address = www.uvw.edu/pqr

[no links]

Page address = www.uvw.edu/abc

[no links]

host Address = www.xyz.com

Page address = www.xyz.com/ghi
www.uvw.edu/pqr

Page address = www.xyz.com/ghi/ijk
www.uvw.edu/abc

Page address = www.xyz.com/ghi/lmn
www.rst.org/abc

host Address= www.rst.org

Page address = www.rst.org/def
www.uvw.edu
www.jkl.edu

Page address = www.rst.org/abc
www.rst.org/def

Fig. 9(a)
Example of
Connectivity at the
Page Level

Pages	Page Level Links
www.xyz.com	--
www.xyz.com/ghi	www.uvw.edu/pqr
www.xyz.com/ghi/ijk	www.uvw.edu/abc
www.xyz.com/ghi/lmn	www.rst.org/abc
www.uvw.edu	--
www.uvw.edu/abc	--
www.rst.org	--
www.rst.org/abc	www.rst.org/def
www.rst.org/def	www.uvw.edu jkl.edu

Fig. 9(b)
Connectivity at
the Page Level

Hosts	Host Level Connections	Frequency
www.xyz.com	www.uvw.edu	2
	www.rst.org	1
www.uvw.edu	--	--
www.rst.org	www.uvw.edu	1
	www.jkl.edu	1

Fig. 9(c)
Connectivity at the Host Level

1002   Obtain list of host pairs

1004   Loop for each host pair from the URL-based method, consider paths that are known to be present on both hosts

1006   Select n such paths from each host in pair. Select paths whose corresponding pages have the highest outdegree (2*n paths in all)

1007   For each of the 2*n paths of this host pair

1008   (Each chosen path will have a page on each of the 2 hosts. Consider the sets of links output from each the two pages)

Count what percentage of the outgoing links are common to both pages. If more than a certain threshold are common, then the paths are said to "match".

(If a link points to one of the two hosts, first remove the hostname to make it a relative URL)

1010   End inner loop (for 2*n selected paths of this host pair)

Determine what percentage of the 2*n paths for this host pair "match". If above a threshold, allow the host pair to remain (if no, eliminate host pair from ranking)

End outer loop (for each host pair)

Fig. 10(a)

Page level connectivity

www.abc.com/xyz

www.def.com/xyz

www.lmn.com

www.example.com

www.lmn.com

www.example.com

www.lmn.com

www.example.com

Fig. 10(b)

Example of Matches at
the Page Level
(current path of 2*n paths
is www.xyz) (path
www.xyz matches 100% in
each case)

www.abc.com/

www.abc.com/xyz

www.example.com

www.def.com/xyz

www.def.com/xyz

www.example.com

www.example.com

Fig. 10(c)

Example of Matches at
the Page Level
(current path of 2*n paths
is www.xyz) (path
www.xyz matches 100% in
each case)

1102

For each host we compute a set of terms, each with an associated frequency. Each term is the name of a host. This is done as follows:

If host1 has n pages which link to some page on host2, then host2 is a term for host1 with a term-frequency of n.

1104

For each host keep only 10 log p terms with highest term-frequency, where p is the number of pages from the host in the input set

1106

Compute a weight for each term t based on in(t), which is the number of hosts which have t as a term. If in(t) <=25 then term-weight is 1, otherwise term-weight is MIN(1/5,200/in(t))

1108

Use term vector matching to determine the likelihood of a pair of hosts being mirrors

Fig. 11

Host level connectivity
(hconn1)

For each host, compute a set of terms, each with an associated frequency.

Each term is the name of a host. This is done as follows:

If host1 has n pages which link to some page on host2, then host2 is

a term for host1 with a term-frequency of n.

For each host keep only 10 log p terms with highest term-frequency, where

p is the number of documents from the host in the input set

Compute a weight for each term t as in hconn1 (see step 1106 of Fig. 11)

Further multiply each term weight by 1+log(maxin/in(t))

where in(t) is the number of hosts that have t as a term and

where maxin is the highest value of in(t) amongst all terms of all hosts

Use term vector matching to determine a likelihood score of a pair

of hosts being mirrors

1202

1204

1206

1208

Fig. 12

Host level connectivity(hconn2)

1302

Obtain top 100,000 host pairs from each of hosts, IP address-based, paths, prefixes, and Hconn (i.e., in parallel)

1304

Sort all host pairs by
number of methods that find each pair
(use best rank obtained by the host pair in any of the methods to break ties during sort)

1306

Use predetermined number of host pairs that received the highest sorted ranking as input to Stage II

Fig. 13(a)

Example of combined approach
(Stage I)

1352

Obtain list of ranked host pairs that are potential mirrors using a URL based method

1354

Use connectivity based method to filter out host pairs less likely to be mirrors (results in output to Stage II)

Fig. 13(b)

Example of combined approach
(Stage I)

1402 — Loop for each host pair <host1, host2>

1404 — Fetch the root page for each host and check for equivalence of content (use threshold, e.g., 50%)

1406 — Did both root pages load?   N   Y

1408 — Do root pages have high syntactic similarity?   N   Y

1410 — Do a set of paths sampled from host1 and host2 yield a predetermined number of highly similar documents (see Fig. 14(b))   N   Y

1412 — Identify current host pair as mirrored

1416 — Identify current host pair as not mirrored

1418 — End loop for current host pair <host1, host2>

Fig. 14(a)

Test to determine whether each ranked pair is a mirrored pair (Stage II)

1452 — Sort all paths of host1 and host2 lexicographically

1454 — Sample 15 paths uniformly from each sorted list of paths (30 total)

1456 — Randomly permute both lists of paths

1458 — Test paths alternatively from the two permuted lists (attempt to access each tested path on both hosts)

(A) 1460 — Path failed on both hosts?

1462 — Ignore this path (continue testing) — (A)

1464 — Path succeeds on both hosts, but pages are not highly similar?

1466 — Declare a mismatch between these hosts and move on to next host pair (not highly similar)

1468 — Path fails on one host but not the other?

1470 — Wait and retest later. If path fails on one host but not the other a second time, declare a mismatch between these hosts (not highly similar)

1472 — Both succeed and fetched pages are highly similar

1474 — Tenth path match for these hosts?

1476 — hosts are mirrors

1478 — Make note of path match (continue testing this host pair) — (A)

Fig. 14(b) Whether a set of paths sampled from host1 and host2 yield highly similar documents (Stage II)

if MIRROR(A,B) AND MIRROR(B,C) THEN MIRROR(A,C)

if MIRROR(A,B) AND MISMATCH(B,C) THEN MISMATCH(A,C)

Fig. 15

Exemplary pooling Evaluations
(Stage II)

Optional IP address check here

1602 — Loop for each host pair <host1, host2>

1604 — Fetch the root page for each host and check for equivalence of content (use threshold, e.g., 50%)

1606 — Potential outcomes:
SF - source failure (root fetch failed)
FM - fingerprint match. Content is byte-wise identical
FS - Full similarity. The documents are 100% equivalent after removal of HTML tags
HS - high similarity. Common content is above a threshold for high similarity by some syntactic similarity measure (e.g., 50% threshold)
NS - path is valid, but no syntactic similarity

1608 — Make 18 more web page comparisons (9 random pages sampled from each host). The host that the web page belongs to is the source. The other host is the target.
Attempt to access and compare the 18 web pages will results in
SF - source failure (could not fetch page from source)
TF - target failure (could not fetch page from target)
FM - fingerprint match, Content is byte-wise identical
FS - Full similarity. The documents are 100% equivalent after removal of HTML tags
HS - high similarity. Common content is above a threshold for high similarity by some syntactic similarity measure (e.g., 50% threshold)
NS - path is valid, but no syntactic similarity

1610 — Classify each host pair as one of level 1-5 (or mismatch) (see Fig. 16(b))

1612 — End loop for current host pair <host1, host2>

FIG. 16(a)   Test to classify each host pair into one of several mirroring categories (Alternate Stage II)

1652 — All tests show SF or FM and not all are SF?

1654 — Level 1 - all paths are replicated and content is identical

1656 — All tests show SF or FM or FS and not all are SF?

1656 — Level 2 - all paths are replicated and content is equivalent

1660 — All tests show SF or FM or FS or HS and not all are SF?

1662 — Level 3 - all paths are replicated and content is highly similar

1664 — Not all tests are SF or TF or NS?

1666 — Level 4 - some paths are replicated. For replicated paths, content is similar

1668 — No test shows TF and not all are SF?

1670 — Level 5 - all paths are replicated but content may not be similar

1672 — All tests show TF, SF, or NS?

1674 — Mismatch. The hosts are not mirrors

Fig. 16(b)

Classification of a host pair (from Fig. 16(a))

Count number of hosts that have each IP address.

1702

Do more than a threshold number of hosts have a particular IP address?

1704

Y

Too many hosts have this IP address. They cannot be mirrors. Classify as mismatches

1706

N

Classify as mirrors

1708

Fig. 17

Optional IP address check at classification step (Stage II)

1

METHOD AND APPARATUS FOR FINDING MIRRORED HOSTS BY ANALYZING URLS

RELATED APPLICATION

U.S. patent application Ser. No. 09/307,153 now pending of Bharat et al., entitled, "Method and Apparatus for Finding Mirrored Hosts by Analyzing Connectivity and IP Addresses," which is incorporated herein by reference and which is filed concurrently herewith.

U.S. patent application U.S. Pat. No. 5,909,766 of Broder et al., entitled, "Method for Determining the Resemblance of Documents," which is incorporated herein by reference and which was filed on Jun. 18, 1996.

U.S. patent application U.S. Pat. No. 6,119,124 of Broder et al., entitled, "Method for Clustering Closely resembling Data Objects," which is incorporated herein by reference and which was filed on Mar. 26, 1998.

BACKGROUND OF THE INVENTION

The present invention relates generally to a method and apparatus for finding mirrored hosts and, specifically, to a method and apparatus for finding mirrored hosts by analyzing connectivity and naming structures of the host and of the web pages of the host.

In recent years, the World Wide Web ("the web") has grown hugely in popularity and use. Currently, almost any type of information can be found on the web if one knows where to look. Knowing where to look has increasingly become problematic because the number of web sites that make up the web have grown at an astounding rate since the early 1990s. In recent years, increasingly sophisticated software, such as search engines and web browsers have been developed that allow users of the web to locate information in the web. Other software, such as proxy servers improve the speed and security of web usage.

A web crawler is a software program that fetches a set of pages from the web by following hyperlinks between the pages. Search engines, such as Compaq Computer Corporation's Alta Vista search engine, employ crawlers to build the web page indexes used by the search engine. Web browsers are applications that fetch and display web pages to a user. Proxy servers (proxies) fetch web pages from web server systems on behalf of web browsers. For efficiency reasons, proxies and browsers sometimes cache web pages (that is, store their content locally). Thus, if a cached page is requested a second time, it can be retrieved from a local cache.

A page in the web is accessed by its web address, also called a Uniform Resource Locator (URL). The URL of a web page has three parts: 1) an access type (such as "http") 2) a host name, which identifies the host on which the page is stored, and 3) a path, which specifies a location within the host. A web site is made up of one or more web pages. As shown in FIG. 7, the format of a URL of a web page looks like:

<access type>://<host>/<path>

where <host> is the name of the web server that stores the web site and <path> is the path of the page within the web server.

It has become increasingly common to duplicate all or part of certain popular web sites. For example, download hosts for certain popular software are often "mirrored" so that users can obtain the same downloadable software from any one of the mirrored hosts. Mirroring is the systematic

2

replication of content across hosts. Mirroring happens when distinct hosts provide access to copies of the same data. Because mirrored hosts allow users to obtain the same information from any of the mirrored hosts, mirroring helps avoid bottlenecks at popular hosts. Hosts are mirrored for a variety of other reasons. Mirrored hosts may have identical page structures or they may contain only certain pages and page structures that are identical. In this document, two separate tests are used when determining mirrors. In a first test, two hosts, A and B are "mirrors" if and only if for every document on host A there is a highly similar document on B with the same path and vice versa. A second test categorizes pairs of hosts according to a plurality of mirroring categories, where the categories represent degrees of miroring. The two hosts do not have to be exactly matched in structure and/or content to be mirrored hosts.

Crawlers, search engines, and proxy servers all fetch large numbers of pages on the web. If these programs could detect mirroring in hosts, they could refrain from fetching content from all but one of the mirrored hosts, thus reducing the number of pages fetched and improving their overall performance. Given a large list of URLs encountered on the web (such as a list collected by a crawler of the list of URLs viewed by a central proxy of a large Internet Service Provider) it is desirable to be able to determine which hosts are mirrored. Some specific examples are provided below.

Often search engines index only one copy of a mirrored page. In the process, they may fetch replicas and discard them. If mirroring information were available, a search engine could avoid fetching replicas from known mirrored hosts. The search engine could also distribute fetches of the remaining pages between the mirrors for load balancing, or choose the best mirror in terms of response time.

Proxy servers and web browsers maintain cached copies of downloaded pages to avoid re-fetching. The effectiveness of such caches can be increased if mirroring information is available. When a URL needs to be fetched, the cache is first checked. If a requested page has not yet been fetched, but it is determined that a page from a mirrored host with the same path has been fetched and is available in the cache, the cached mirror page can be used instead of fetching the requested page.

Thus, the ability to identify mirrored hosts would improve the speed and efficiency of operation of software accessing the world wide web.

Certain conventional web crawling software are able to identify some mirrored web sites by using Domain Name Server (DNS) lookup. When a crawler fetches a URL, it needs to first convert the hostname of the URL to a corresponding Internet Protocol (IP) address to establish a network connection. Such lookups are done using a service known as DNS. A DNS lookup returns one or more IP address for each hostname. Crawlers usually treat hosts that have an IP addresses in common as mirrors to avoid redundant fetching. This method does not always identify all mirrored hosts and may mis-identify some hosts as mirrored that are not mirrored. For example, a "virtual host" is a host that hosts more than one web site but has a single IP address. The web sites hosted by a virtual host web server, while all having the same IP address are not necessarily mirrors. Similarly, not all mirrored hosts share a common IP address. In addition, some hosts may have more than one IP address. Thus, IP matching alone is not always sufficient to prove that two hosts are mirrors of each other.

SUMMARY OF THE INVENTION

The described embodiment of the present invention addresses not only the problem of finding identical mirror

hosts, but also the problem of finding hosts that are not completely identical, but contain a significant amount of shared content. This information is useful in understanding the composition of the web and the collaborations ongoing between principals on the web.

The described embodiment of the invention detects mirrored host pairs using information about a large set of pages, including URLs. The identities of the detected mirrored hosts are then saved so that browsers, crawlers, proxy servers, or the like can correctly identify mirrored web sites. The described embodiments of the present invention look at the URLs of pages hosts to determine whether the hosts are potentially mirrored.

In accordance with the purpose of the invention, as embodied and broadly described herein, the invention relates to a method of determining mirrored web hosts, comprising: receiving information about the addresses of a plurality of web sites stored on a plurality of hosts; determining a plurality of terms of the URLs associated with every host; weighting the terms in inverse proportion to frequency; determining a similarity score for host pair in accordance with the weighted terms; and outputting a list of potential pairs of mirrored hosts in accordance with their similarity scores.

Advantages of the invention will be set forth in part in the description which follows and in part will be obvious from the description or may be learned by practice of the invention. The objects and advantages of the invention will be realized and attained by means of the elements and combinations particularly pointed out in the appended claims and equivalents.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate several embodiments of the invention and, together with the description, serve to explain the principles of the invention.

FIG. 1 shows an example having a client system, proxy server software, web crawler software, web search engine software, browser software, and a web server host in configuration in which the present invention can be employed, along with software for determining a list of mirrored hosts in accordance with the present invention.

FIG. 2(a) shows an example of first and second mirrored hosts, where the structure of the web sites is duplicated exactly.

FIG. 2(b) shows an example of first and second mirrored hosts, where the structure of the web sites is not duplicated exactly.

FIG. 3(a) is a flow chart showing an example of how web crawler software might use a list of mirrored hosts to improve its operation.

FIG. 3(b) is a flow chart showing an example of how proxy server crawler software might use a list of mirrored hosts to improve its operation.

FIG. 3(c) is a flow chart showing an example of how browser software might use a list of mirrored hosts to improve its operation.

FIG. 3(d) is a flow chart showing an example of how search engine software might use a list of mirrored hosts to improve its operation.

FIG. 4 is a flow chart showing an overview of a method of finding mirrors in accordance with the present invention.

FIG. 5 is an example of an IP address format.

FIGS. 6(a) and 6(b) are flow charts showing methods of using IP addresses to detect mirrors.

FIG. 7 is an example of a format of a URL (Uniform Resource Locator).

FIG. 8(a) is a flow chart showing methods of using URLs to detect mirrors.

FIG. 8(b) shows an example of determining tuples in accordance with a URL.

FIG. 8(c) shows an example of determining host pairs n accordance with the tuples of FIG. 8(b).

FIG. 9(a) shows an example of a plurality of connected web sites.

FIG. 9(b) describes the page level connectivity of the web sites of FIG. 9(a).

FIG. 9(c) describes the host level connectivity of the web sites of FIG. 9(a).

FIG. 10(a) is a flow chart showing a method of using connectivity at a page level to detect mirrors.

FIG. 10(b) shows an example of connectivity matches at the page level.

FIG. 10(c) shows another example of connectivity matches at the page level.

FIG. 11 is a flow chart showing a first method of using connectivity at a host level to detect mirrors.

FIG. 12 is a flow chart showing a second method of using connectivity at a host level to detect mirrors.

FIG. 13(a) is a flow chart showing an example of a method that combines more than one method of detecting mirrors in a parallel order.

FIG. 13(b) is a flow chart showing an example of a method that combines more than one method of detecting mirrors in a serial order.

FIGS. 14(a) and 14(b) are flow charts showing a method that takes a list of potential pairs of mirrors and examines mirror content and linkage to determine whether each pair represents mirrored hosts.

FIG. 15 is a flow chart representing additional details of FIG. 14.

FIGS. 16(a) and 16(b) are flow charts showing a method that takes a list of potential pairs of mirrors and examines mirror content and linkage to classify the mirror pairs as one of several types of mirrors or non-mirrors.

FIG. 17 is a flow chart representing additional details of FIG. 16.

DETAILED DESCRIPTION OF EMBODIMENTS

Reference will now be made in detail to several embodiments of the present invention, examples of which are illustrated in the accompanying drawings. Wherever practicable, the same reference numbers will be used throughout the drawings to refer to the same or like parts.

1. General Discussion

FIG. 1 shows an example in which a client system 102, a proxy server 104, another server 105, and a plurality of web servers 132, 136 in a configuration in which the present invention can be employed. Proxy server 104 includes proxy server software 124. Server 105 includes crawler software 126 and web search software 128. In such a system, search engine software 128 uses web crawler software (such as crawler 126) to perform analysis of a large number of hosts on web server systems, such as web server hosts 132,136. (Hosts are often also called "web hosts" or "web sites." Herein the two terms are used interchangeably). As discussed above, web crawler software 126 can use its knowledge of which hosts are mirrored to increase its crawling and

caching efficiency. The information returned by the crawler **126** is used by the search engine to determine further information about mirrored hosts in order to improve the data it returns to the client. (Web crawlers can also be used by software other than search engines). Information about mirrored hosts, once determined, can also be sent to browser software **114** and/or proxy server software **124** to increase their caching efficiency.

FIG. **1** also shows a system **142** having a processor **144** and software **146** in memory that determines whether various hosts on the system are mirrored in accordance with the present invention. FIG. **1** also includes a plurality of web sites hosted by web servers **132, 136** having different IP addresses. Although not shown, more than one web site/host can share an IP address.

FIG. **2**(*a*) shows an example of first and second mirrored web sites, where the structure of the web sites is duplicated exactly. As can be seen in the figure, each and every link in on a www.host1.org host is duplicated on the www.host2.edu host. The LRLs of the pages are not identical (and the IP addresses may not be identical) but the structure of the two hosts are exact mirrors of each other. Identical structures do not necessarily mean that the page content is also identical, although, as discussed below, a certain amount of similarity is required for the hosts to be mirror hosts.

In contrast, FIG. **2**(*b*) shows an example of first and second mirrored web sites, where the structure of the web sites is not duplicated exactly. In the example, a new page "www.host2.edu/b/e" has been added under page "www.host2.edu/b" but the structure is otherwise the same. As discussed in connection with FIG. **2**(*a*), the content of the pages with identical paths may or may not be the same

2. Examples of Why it is Useful to Identify Mirror Hosts

FIG. **3**(*a*) is a flow chart showing an example of how web crawler software might use a list of mirrored hosts to improve its operation. The list of mirrored hosts is generated, for example, by software **146** of FIG. **1**. In step **302**, the web crawler puts an address (URL) of an uncrawled web page in a queue. The crawler then fetches this page and collects information about the page. (step **306**). The crawler then collects all addresses (URLs) that are linked to on that page (step **308**). Normally, these URLs would be added to the queue and crawled in turn. In this situation, however, if the crawler can determine that a page is a mirror page of a page recently crawled, there is no need to crawl the page again. Thus, links are added to the queue unless they are mirror pages and already crawled. The crawler then fetches the next page address from the queue and repeats the process. The crawler preferably uses a lists of mirrored hosts that has been previously determined in accordance with the present invention. Alternately, the crawler may determine mirror hosts "on the fly" as it is crawling the sites (as soon as it accumulates enough information to be able to identify mirror sites form the information).

FIG. **3**(*b*) is a flow chart showing an example of how proxy server crawler software might use a list of mirrored hosts to improve its operation. In step **322** the proxy server receives from a client an address (URL) of a page to fetch. If, in step **326**, the requested page (or a mirror of the page) is already in the proxy cache, the proxy server retrieves the page from its cache and returns it to the client (step **328**). Otherwise, the proxy server fetches the page from the host server and returns the page to the client (step **324**). The proxy server preferably uses a lists of mirrored hosts that has been previously determined in accordance with the present invention.

FIG. **3**(*c*) is a flow chart showing an example of how browser software might use a list of mirrored hosts to improve its operation. In step **342** the browser receives from a user an address (URL) of a page to display. If, in step **346**, the requested page (or a mirror of the page) is already in the browser cache, the browser retrieves the page from its cache and displays it (step **348**). Otherwise, the browser fetches the page from the host server and then displays the page (step **344**). The browser preferably uses a list of mirrored hosts that has been previously determined in accordance with the present invention.

FIG. **3**(*d*) is a flow chart showing an example of how search engine software might use a list of mirrored hosts to improve its operation. In step **362**, the search engine receives search term(s) and reviews its previously determined search index to identify potential pages having these search terms. Step **364** eliminates mirror hosts so that mirrors hosts are not shown in the search results. Alternatively, mirror hosts could be eliminated when the index of the search engine are created. Step **366** augments the search results to identify certain hosts as having mirrors (and possibly giving the URL of the mirror hosts). Step **368** identifies which search results links are broken links and attempts to replace the broken links with links to a mirror host of the broken link. It should be understood that the steps of FIG. **3**(*d*) are exemplary only and that other search engines can use the present invention, which determines mirrors hosts, in any appropriate manner. The search engine preferably uses a lists of mirrored hosts that has been previously determined in accordance with the present invention.

3. Determining a list of mirrored hosts

FIG. **4** is a flow chart showing an overview of a method of finding mirrored hosts ("mirrors") in accordance with the present invention. A first step **402** (called a "Stage I" step) creates a list of potential mirrored hosts. A second step **404** (called a "Stage II" step) analyzes the structure and content of parts of the hosts in the list of potential mirrored hosts to determine whether the potential mirrored hosts are actually mirrored.

Step **402** (Stage I) step receives as input information about a large set of pages, including URLs, host IP addresses, and/or connectivity information, such as hyperlinks at the page and/or host level. This information about pages could be gathered, for example, by web crawler software or could be gathered or generated by any other appropriate methods. Step **402** samples a subset of this information and computes a similarity between members of the subset. Step **402** outputs lists of host pairs <host1, host2> that are potential mirrors, ordered by likelihood that the hosts in the host pairs are mirrors of each other.

Step **402** uses one or more methods, including but not limited to one or more of the following methods to determine host pairs:

IP address based;

URL string based;

Page connectivity based;

Host connectivity based;

Parallel combinations of the above; and

Serial combinations of the above.

Each of these methods is discussed below. The methods used to determine lists of host pairs also can be combined. An example of such as combination method is described in connection with FIG. **13** below.

Step **404** (Stage II step) processes the list of candidate host pairs to determine whether they are actually mirrors.

Details of both step **402** and step **404** are provided below. Note that the methods described herein are probabilistic in nature. Hence there is a small probability that mirrored hosts will not be identified as such, or that some hosts will be declared mirrors, although they are not.

a. IP address based methods for determining a list of potential mirrors

FIG. **5** is an example of an IP (Internet Protocol) address format in the TCP/IP protocol. An IP address is made up of four "octets." Each octet of an IP address can have a value from 0 through 255 (one byte). If exactly two hosts translate into the same IP address, it is likely that they are mirrors accessing the same server, and that the two hostnames are just aliases. In contrast, a match on the first three octets signifies web servers on a same IP subnet. Such web servers are usually part of the same organization, and are often mirrored hosts, replicated to handle heavy traffic.

When many hosts resolve to the same IP address, however, it is often indicative of a web server that is a "virtual host," i.e., that is acting as a server to more than one web site. For example, certain "big IP" servers use the same IP address for many thousands of distinct web sites. These considerations led to the methods of FIGS. **6**(*a*) and **6**(*b*).

FIGS. **6**(*a*) and **6**(*b*) are flow charts showing methods of using IP addresses to detect mirrors. It will be understood that all methods discussed herein are preferably implemented as software instructions stored in a memory and executed by one or more appropriate processors. The steps of Stage I and Stage II are preferably performed by software **146** of FIG. **1** or by some similar or equivalent software.

It will also be understood that any constant numerical values used herein are provided for the purpose of example only and are not to be interpreted in a limiting sense. The methods of the present invention can be used with any appropriate constant values and any appropriate threshold values and/or weights.

In FIG. **6**(*a*), step **602** inspects the IP addresses in the input information and clusters or groups together hosts that have the same IP address. Thus, the largest number of possible IP address clusters that can be found corresponds to the largest possible number of IP addresses (i.e., one host per IP address cluster). The clusters are ranked according to the number of hosts in each. Clusters with a smallest number of members (greater than one) arc most likely to be mirrors.

The method of FIG. **6**(*b*) is performed similarly to the method of FIG. **6**(*a*) except that, instead of considering an entire IP address, the method considers only the first three octets of an IP address.

b. URL based methods for determining a list of potential mirrors

FIG. **7** is an example of a format of a URL (Uniform Resource Locator) used to address resources in the world wide web. A URL includes an access method (such as "http," "ftp," etc.); a host name (for example,www.host1.com); and a path identifying the path on which a particular page is located in the host structure (for example, ABC/xy/z/hmtl). Paths can be any arbitrary reasonable length.

URL strings provide information about a potential mirroring relationship between hosts in two different ways:

1) similar hostnames suggest that hosts belong to the same or related organizations.

2) similar paths indicate a possible replication of directories. This is because paths usually reflect the host's directory structure. The following embodiment uses "term vector matching" to compute the likelihood that a pair of hosts are mirrors. Term vector matching is described in G. Salton and C. Buckley, "Term Weight-

ing Approaches in Automatic Text Retrieval," Information Processing and Management, 24(5), pages 513–523, 1988, which is herein incorporated by reference. Based on the type of term used, various methods can be used, each with an appropriate weighting scheme. The method of FIGS. **8**(*a*)–**8**(*c*) use terms. Possible terms include: bi-grams (see below), host names, paths, and portions of a URL, such as terms preceding a slash ("/") or other terminator character.

FIGS. **8**(*a*)–**8**(*c*) show an example of how to determine host pairs in accordance with a URL. FIG. **8**(*a*) is a flow chart showing a method of using URLs to detect mirrors, using "bi-grams" (also called "shingles") as terms. As described below, bi-grams are pairs of tokens from a URL along with associated depth information. Instead of two tokens, n-grams with more than two tokens can be used as well. For large data sets, such as the initial data set described herein, reducing the number of terms to be considered is a priority. One way to do this is to ignore any hosts that contribute a small number of URLs to the collection. Such hosts are either small in size and hence unlikely to have much mirrored content, or poorly sampled, in which case the samples in the collection may not be very representative. The described method considers only hosts that have at least 100 URLs in the collection of data. A further data reduction is obtained as follows: for every host, we first sort the list of URLs by alphabetical order, then consider only those paths whose strings upon hashing yield a value that is 0 mod m. Any conventional hashing method for strings can be used. For example, m can initially be 5. The aim of this step is to increase the correlation between the selected paths. If a path is selected on one host, it increases the likelihood that the same path is selected on its mirror.

After the first 20 paths are sampled this way, m is doubled, and this doubling is repeated at certain thresholds. The doubling thresholds are picked so that the sample count from a host is sub-linearly proportional to size. This ensures that, on the one hand, more paths are sampled from larger hosts than smaller hosts but, on the other hand, larger hosts do not dominate the list of terms. Alternately, different or no input sample reduction need be performed.

Steps **802** and **804** of FIG. **8**(*a*) generate "bi-grams" for the paths of the input data sampled as described above. Bi-grams are pairs of tokens from a URL (along with associated depth information) and are used as terms in the example method shown. An example of bi-grams is shown in FIG. **8**(*b*). In step **802**, each URL is converted to lower case. Paths may or may not be case-sensitive, but the hostname is. Even so, case is preferably ignored. Any non-alphabetical chacters (such as numbers) are treated as word breaks when choosing tokens. Thus, in FIG. **8**(*b*), a URL of www.a.b.com/cellblock16/inmates/me/personal/foo.html will yield tokens of (cellblock, inmates, me, personal, foo, and html). In step **804**, bi-grams are generated by treating every continuous pair of tokens as a term and by adding depth information concerning the level of the term in the path. For example, the example of FIG. **8**(*b*) generates the following set of bi-grams for URL http://ab.com/cellblock16/inmates/me/personal/foo.html:
{(Cellblock, inmates,**0**), (inmates, me,**1**), (me, personal, **2**), (personal, foo,**3**), and (foo, html,**4**)}. The depth information is useful when trying to find mirror hosts that share the same path structure. By fragmenting the path, we also associate depth information with the path.

In addition, the set of bi-grams may be reduced by the following three optional steps:

1) eliminate commonly used tokens such as htm, html, tst, main, index, home, bin, cgi

2) avoid URLs that have tokens such as nph, dynaweb, zyview, etc. which are characteristic of web sites created automatically by tools. Such hosts lead to spurious matching because they use a standard naming scheme.

3) eliminate path terms that do not occur at least twice on the host. This eliminates leaf patterns in the path, unless they are part of a pattern.

Step **806** groups the bi-grams as terms in <terms,host> tuples and sorts the list of tuples by the first element of the tuple (the term). For all terms, this causes all hosts that contain that term to be listed contiguously. This allows the list to be processed in a single pass in order to compute the similarity between pairs of hosts that share common terms.

It will be understood that any appropriate terms can be used in the above step, including but not limited to: host names, paths, and portions of a URL, such as terms preceding a slash ("/") or other terminator character.

Step **808** determines a similarity score for every pair of hosts, H1 and H2, in the sorted list of tuples. The similarity score for a pair of hosts need not be explicitly maintained unless it is non-zero.

The method updates the similarity score for all pairs of hosts that share a term. In the example of FIG. **8**(*c*), the hosts www.a.b.com and www.x.y.com share all their terms, even though their paths differ slightly. If f is a term that is common to H1 and H2 then, similarity score is incremented by weight(f). Weight(f) is determined as follows:

$$\text{weight}(f) = S(f)N(f),$$

where S(f) is a measure of the significance of f independent of its distribution. In the described embodiment, S(f) is set to 4 for host terms and to 1 for path terms, since a host term match offers evidence that the two hosts are part of the same organization (given that only rare terms are considered). N(f) is the number of hosts in which term f occurs. Dividing by N(f) reduces the score of terms that occur frequently and raises the score of terms that occur rarely. This is similar to the notion of IDF (inverse document frequency) weighting using in the information retrieval field. See Sparck-Jones and Willett, "Readings in Information Retrieval," Morgan Kaufman Publishers, July 1997, which is herein incorporated by reference.

Because common terms have low weights, as an optimization, the described embodiment skips all terms that occur within more than 20 hosts. When all terms have been processed, every pair of hosts that share a rare term will have a similarity score. Step **808** normalizes and sorts the similarity scores of each host pair (H1, H2). The similarity score is normalized as follows:

$$\text{Normalized similarity score}(H1, H2) = k1 * \text{Similarity\_score}(H1, H2) \ k2 + \text{Log(size of } H1) + \text{Log(size of } H2)$$

The parameters k1 and k2 help normalize the score by host size, to compensate for the fact that large hosts will have more term matches than small hosts. In the described embodiment, k1=30 and k2=20. The size of a host is the number of URLs from the host present in the input set.

At the end of this run, the normalized similarity score of every pair of hosts that share a significant term will have been computed. A list of <score,host-pair> tuples is written to a file. To reduce mismatches, host pairs having only one term in common are filtered out. Similarly, hosts pairs that do not have a path term in common are filtered out. The remaining host pairs are sorted in the descending order of normalized similarity score. Step **810** yields the output from Stage I, namely a list of potential mirrored hosts, in decreasing order of probability.

As discussed above, URL based mirror detection can also use hostnames, paths, prefixes, etc. of URLs as terms forming a term vector. Each of these is discussed below. When the terms used are substrings of hostnames, only terms occurring in less than H hosts are used in the hosts term vectors (in the described embodiment, H=100). In the described embodiment, a hostname substring is defined as a substring from a URL either beginning with the hostname or with a "." and ending either with the hostname or with ".". When the hostname is specified as an IP address, the first three octets of the IP address are used. As a further optimization, if the document frequency is greater than, for example, 25, the term is restricted to appear in the vectors of only, for example, 25 hosts chosen at random. The weight of term t is computed in terms of its document frequency df(t) and len(t), which is the number of segments obtained by breaking the term at "." characters. The weight of term t is computed as $((\log(\text{len}(t))/(1+\log(\text{df}(t))))$. This weighting favors substrings composed of many segments that are likely to be very specific.

When the terms used are the entire path, the same document frequency based restrictions used for hostnames are used. The weight of term t is computed in terms of its document frequency df(t) and maxdf, which is the maximum document frequency of any term in the collection. Because the described embodiment discards all terms with document frequencies greater than 100, maxdf is effectively 100. The weight is computed as $(1+\log(\text{maxdf}/\text{df}(t)))$.

The terms used can also be the prefix of a path. A prefix is defined as the part of the path that either ends in a "/" or terminates the path. This results in a large number of terms per host, of which only 10 log p terms with the largest term frequency are kept, where p is the number of documents in the input set. The same weighting scheme used for path matching is preferably used. However, to compensate for the fact that smaller hosts tend to contribute fewer terms, the host similarity weight is normalized by a multiplication factor $(1/(1+0.15(\log(nt1)+\log(nt2))))$, where nt1 and nt2 represent the number of terms from each of the two hosts.

c. Connectivity based methods for determining a list of potential mirrors

1. Page level connectivity based method

FIG. **9**(*a*) shows an example of several connected hosts (web sites). FIG. **9**(*b*) is a table showing page level connections between the hosts. Various pages within the hosts www.xyz.com, www.uvx.edu, and www.rst.org have links to other pages in other hosts. Pages are shown as rectangles and links are shown as oblongs within pages. The described page level method looks at the links (i.e., connectivity information) within certain pages when determining potential mirrors. In FIG. **9**(*b*), four of the pages have one link to other pages; one of the pages has two links to other pages, and four of the pages have no links to other pages.

FIG. **10**(*a*) is a flow chart showing a method of using connectivity at a page level to detect mirrors. This method is also known as the "conn method" (for "connectivity"). In step **1002**, the described method preferably uses a URL based method similar to that described above in connection with FIGS. **8**(*a*)–**8**(*c*) in which the terms used are "paths" (instead of bi-grams, although any appropriate URL based method could be used). This step identifies host pairs with many common paths as mirrors. The connectivity-based method involves testing further for certain paths present on both hosts of a host pair, if the corresponding pages have the same (or similar) set of out-links on both hosts under consideration. Host pairs that fail the test are eliminated.

Step **1004** loops through all host pairs. The loop considers only paths (of pages) that are known to be present on both

hosts of a host pair. Step **1006** looks at the page-level connectivity of each of the hosts in the host pair. For example, hosts www.abc.com and www.def.com may have respective common paths www.abc.com/xyz and www.def.com/xyz. The described method looks at the page-level connectivity (i.e., all links going out of) of paths on the two hosts, including pages www.abc.com/xyz and www.def.com/xyz.

Not all paths/pages of a host pair are looked at. Step **1006** selects n paths from each host in the host pair. The n paths correspond to n pages that are both present on both hosts in a host pair and have a highest "outdegree" for the host (n is, for example, 10). Thus, 2*n pages are selected for each host pair. An outdegree of a path is defined as the number of links from the page corresponding to the path. Thus, if path www.abc.com/xyz contains links pointing to three other pages, page www.abc.com/xyz will have an outdegree of 3. The intuition is that pages with many links are likely to produce more evidence of a mirroring relationship than those with a small number of links. Some of the selected 2*n paths may (and hopefully will) correspond to mirrored pages.

Each of the 2*n paths for a host pair is examined in turn in loop **1007**. Step **1008** counts what percentage of the links in the page corresponding to the path match (i.e. point to the same place). Links to other documents on the same host will involve the name of the local host and will hence appear to differ. We compensate by removing the hostname from such URLs, effectively turning them into "relative URLs." FIG. **10(**b**)** shows a first example of two pages (www.acb.com/xyz and www.defcom/xyz) that have links to the same third and fourth pages and, thus, "match." Both pages contain a link to document/page www.lmn.com and document/page www.example.com. FIG. **10(**c**)** shows a second example of two pages (www.acb.com/xyz and www.defcom/xyz) that "match." Both pages contain a link to document/page www.example.com and a link to a document/page on the respective mirrored systems (www.abc.com/xyz and www.def.com/xyz). Thus, even though the links to the mirrored systems do not point to the same physical pages, the relative URLs (xyz and xyz) pointed to by the links are the same.

Because content changes over time (e.g., at news hosts) and mirrors engage in local customizations, the set of out-links in replicated documents often varies slightly. Hence, when testing a common path for equivalence we require only that a fraction of the union of the set of out-links from the two versions be common to both (in the described embodiment, this fraction is 0.9). Thus, the links of the common 2*n paths are examined and each of the 2*n paths for the current host pair is determined to match or not match.

Step **1010** determines what percentage of the 2*n common pages for the current host pair "match." If a fraction m of the selected common pages of a host pair are found to be equivalent (m=0.75, for example) then the host pair is allowed to remain in the ranking. Otherwise the host pair is eliminated (filtered).

This method further refines the URL based ranking by filtering host pairs that do not point to a high percentage of the same pages.

2. Hosts level connectivity based method

FIG. **9(**c**)** is a table of the host level connectivity of the three hosts of FIG. **9(**a**)**. In FIG. **9(**c**)**, one of the hosts has no links to other hosts and two of the hosts have two links to other hosts. Note that pages on host www.xyz.com have three page-level links to various pages on host www.uvw.edu (see FIG. **9(**b**)**), but host www.xyz.com has only two

host level connections to host www.uwv.edu. This is because pages www.uwv.edu/pqr and www.uvw.edu/abc are both on the host www.uwv.edu and links to both of these pages are counted as one page level connection.

Two hosts are likely to be mirrors if they point to the same document/pages. However, because the input set is incomplete and due to local and temporal variations in content, mirrored hosts tend to have a "similar" rather than identical set of out-links.

FIG. **11** is a flow chart showing a first method of using connectivity at a host level to detect mirrors. FIG. **12** is a flow chart showing a second method of using connectivity at a host level to detect mirrors. Both methods use term vector matching to compute the likelihood of a host pair being mirrors. The terms used are a hosts that a host points to. Thus, at host level connectivity, no distinction is made which page within the host contains the link pointing out of the host. Note that the document frequency of a term is precisely the indegree of the host it corresponds to. The term weight is based on inverse document frequency but not on term frequency. This is because many hosts tend to be poorly sampled and hence the method only has a lower bound on term frequencies.

Step **1102** computes a set of terms for each host pair. Each term is the name of a host. This is done as follows: If host1 has n pages which link to some page on host2, then host2 is a term for host1 with a term frequency of n.

However, as shown in step **1104**, term frequency is also used in term selection. Initially, the method prunes the hosts by keeping only 10 log p terms with highest term frequency. The value p is the number of pages on that host in the input set.

As shown in step **1106**, the method of FIG. **11** ("hconn1") assigns the following term weight to each term t. The indegree in(t) of a host is the number of hosts that have links to it. If the indegree in(t) of host t is at most 25, the term weight is 1. Otherwise, the term weight of host t is min(1/5, 200/in(t)). The term weight is linear in 1/in(t) since computing an approximation of this term weight can be implemented efficiently by allowing the term to appear only in vectors for min(200, in(t)/5) of the hosts chosen at random. Step **1108** uses term vector matching to determine the likelihood of a pair of hosts being mirrors.

FIG. **12** is a flow chart describing a second method of mirror detection using host level connectivity matching. The method of FIG. **12** is also called "hconn2." Steps **1201** through **1204** are similar to step **1101** through **1104** of FIG. **11**. In step **1206**, a weight for each term t is chosen as in the method of FIG. **11** (see step **1106** of FIG. **11**). In addition, each term weight is multiplied by (1+(log(maxin/in(t)))), where in(t) is the number of hosts that have t as a term and where maxin is the highest value of in(t) amongst all terms of all hosts. As in FIG. **11**, term vector matching is used to determine a likelihood score of a pair hosts being mirrors.

b. Combined method for determining a list of potential mirrors

FIG. **13(**a**)** is a flow chart showing an example of a method that combines more than one method of detecting mirrors in a parallel order. The combination method obtains a top number of host pairs (e.g., 100,000) from each of various Stage I methods. In the example, the URL based method using terms derived from hostnames is a first method used. An IP address based method is the second method used. The third and fourth methods used are URL based methods using paths and prefixes as terms. The fifth method used is one of the above-described hconn methods (see FIGS. **10, 11** and **12**). The output of each of these Stage

I methods is a list of potential mirrored host pairs. Other appropriate methods for determining pairs of potentially mirrored hosts can be used instead of or in addition to those shown here. In step **1304**, all the host pairs are sorted according to the number of methods that found each host pair. Ties are broken using a best rank obtained by each method over each host pair for all methods. Step **1306** uses a predetermined number of the newly sorted host pairs as the output of the combined method. These host pairs can become input to a Stage II method described below.

FIG. **13**(*b*) is a flow chart showing an example of a method that combines more than one method of detecting mirrors in a serial order. Step **1352** obtains a list of ranked host pairs that are potential mirrors using a URL based method, such as a method explained above. Step **1354** uses a connectivity based method, such as that described above to filter out host pairs less likely to be mirrors. Step **1354** results in output to Stage II. The method of FIG. **13**(*b*) is provided by way of example. In general, a fastest method is used on the initial large set of input data to output a list of sorted potential mirrored hosts pairs. At least one other of the mirroring method or methods described herein are used to filter the sorted hosts pairs.

4. Classifying potential host pairs

All of the above methods results in a list of prioritized potential mirrored host pairs. The methods described in this section determine whether each potential mirrored host pair are in actuality mirrored hosts. FIGS. **14–15** described a first method that determines whether a host pair is mirrored or not mirrored. FIGS. **16–17** described a second method that outputs a range of classification values ranging from mirrored, through possibly mirrored, through not mirrored.

FIGS. **14**(*a*) and **14**(*b*) are flow charts showing a method that takes a list of potential pairs of mirrors and examines mirror content and linkage to determine whether each pair represents mirrors. Step **1402** is the beginning of a loop performed for each potential host pair <host1, host2>. Step **1404** fetches a root page for each host and checks for similarity of content in the root pages. In the described embodiment, a 50% content match is required for the root pages to be similar. Note that, in this method, the actual root pages of each host are fetched and the content of the fetched root pages are compared according to the methods described below.

Various methods can be used for determining content match of two pages without departing from the spirit of the invention. In one embodiment, testing is done strictly by finger printing (computing a checksum for each fetched page). In practice, however, the content of even mirrored pages is not fingerprint identical. This can happen even on successive fetches of the same page from the same server, due to variable server-side operations or dynamically-generated web pages. With mirrored content, there is an even greater likelihood of a discrepancy, due to version inconsistencies and local server side operations.

To compensate for this, the content is preferably de-tagged, whitespace is ignored, and a check is done for "syntactic similarity" instead of fingerprint equality. One such syntactic similarity scheme is disclosed in Broder et al, "Syntactic Clustering of the Web," SRC Technical note, 1997-015, Jul. 25, 1997, Digital Equipment Corp, Systems Research Center, Palo Alto, Calif., which is herein incorporated by reference. Another similar scheme is disclosed in the above-referenced U.S. patent application Ser. No. 08/665,709 of Broder et al.

This allows the syntactic similarity between two pages to be computed as a percentage. In addition, at least one

embodiment does a DNS lookup on each host to check if the hosts have a common IP address. (When a host has more than one IP address, all are checked to see if at least one is common between the two hosts of the host pair).

If both root pages loaded (step **1406**) and the root pages have high syntactic similarity (i.e.,>=50%) (step **1408**), step **1410** determines if a set of paths sampled from host1 and host2 yields a predetermined number of highly similar documents. Note that, in this method, the exemplary pages from each host are fetched and the content of the fetched pages are compared. (This step is described in more detail in FIG. **14**(*b*).) If the set of paths yield highly similar documents, the hosts are identified as mirrored hosts in step **1412**. Otherwise, step **1416** determines that the hosts are not mirrored. Step **1418** is the end of the loop.

FIG. **14**(*b*) shows an exemplary method for determining whether a set of sampled paths from a host pair yields highly similar documents. The set of sampled paths described here is the set of sampled paths from FIG. **14**(*a*). Because a given host may occur in many host pairs, the described embodiment re-uses the same list of sampled paths for a given host. Step **1452** sorts all paths of host1 and host2 lexicographically. These paths are taken from the initial input data about host1 and host2. Thus, step **1452** sorts all known paths on host1 and host2 into respective sorted lists. In the described embodiment, step **1454** samples fifteen paths uniformly from each sorted list of paths. Thus, a total of thirty paths are sampled. These thirty paths represent paths on the two hosts in the host pair being tested. There is no guarantee that the paths correspond to pages present on both hosts. Step **1456** randomly permutes both lists of paths. Step **1458** tests paths alternately from the two permuted lists by attempting to access each tested path on both host1 and on host2. Note that step **1458** actually attempts to access the pages corresponding to the sorted paths.

The remaining steps of FIG. **14**(*b*) described the results of this access test. If, in step **1460**, the path fails on both hosts, the path is ignored and testing continues with the next path from step **1458**. If, in step **1464**, the path succeeds on both hosts, but the pages are not highly similar (see above), the method declares a mismatch between these hosts and moves on to the next host pair.

If, in step **1468**, the path fails on one host, but not on the other, the method marks this path for later retesting. If, upon later retesting, the path fails on one host but not on the other, a mismatch is declared between the hosts. In this case, it is desirable to test a second time because of the possibility that a failing host was temporarily overloaded or non-operational. Thus, paths that are accessible from one host but not the other are given a second chance. Processing continues with the next path from step **1458**.

If, in step **1472**, the path succeeds and the fetched pages are highly similar (see above), step **1474** determines whether this is the tenth successful matching path for this host pair. If not, a note is made of the successful match (for example, a counter is incremented) and testing continues with the next path from step **1458**. Otherwise, ten highly successful matches have occurred for the current host pair and it is determined that the hosts are mirrors.

FIG. **15** is a flow chart representing additional details of FIG. **14**. In the described embodiment, certain additional processing steps occur after the steps of FIGS. **14**(*a*) and **14**(*b*). For example, if HostA and HostB are mirrors and if HostB and HostC are mirrors, then the method determines that HostA and HostC are also mirrors. Similarly, if HostA and HostB are a mismatch and if HostB and HostC are a mismatch, then HostA and HostC are also a mismatch. These additional steps are omitted from certain other embodiments.

FIGS. **16**(*a*) and **16**(*b*) are flow charts showing a method that takes a list of potential pairs of mirrors and examines mirror content and linkage to classify the mirror pairs as one of several types of mirrors or non-mirrors. Step **1602** loops for each potential host pair <host1, host2>. As described above, an optional IP address check can also be performed. Step **1604** fetches the root page for each host and checks for equivalence/similarity between the two root pages. In step **1606**, the described embodiment separates similarity into similarity categories. In the following a source host is the host on which it is known that the path should exist. (Thus, in the special case of the root pages, there are two source hosts and no target hosts.) The target host is the second host in the host pair.

SF—source failure (access of the path on the source host failed)

TF—target failure (access of the path on the target host failed) (cannot happen for root pages since both root pages are sources)

FM—fingerprint match (content of accessed pages are byte-wise identical)

FS—Full similarity (pages are 100% similar after removal of HTML tags, whitespace, etc.) p1 HS—High similarity (common content on accessed pages above a threshold for high similarity (e.g., 50%))

NS—path is valid on both hosts, but content of accessed pages is not sufficiently similar

In the described embodiment, step **1608** samples 9 more paths from each host for a total of 18 additional paths. Thus, each host is a source host for 9 paths and a target hosts for 9 paths. The method attempts to access each of the 18 paths on the source and target hosts and assigns a syntactic similarity category to each path (for 18 paths, 36 page accesses will be attempted). Step **1610** classifies each host pair as shown in more detail in FIG. **17**. Step **1612** ends the loop.

FIG. **17** is a flow chart representing additional details of FIG. **16**. Each of the 20 paths for a host pair (two root paths and 18 sampled paths) has returned a similarity score using, for example, the similarity score method disclosed in above-referenced. U.S. patent application Ser. No. 08/665,709 of Broder et al. Any other appropriate similarity scoring method could be used.

If, in step **1652**, all tests show SF or FM and not all are SF, the host pair is assigned level **1** similarity (i.e., all paths are replicated and content is identical). If, in step **1656**, all tests show SF or FM or FS and not all are SF, the host pair is assigned level **2** similarity (i.e., all paths are replicated and content is equivalent). If, in step **1660**, all tests show SF or FM or FS or HS and not all are SF, the host pair is assigned level **3** similarity (i.e., all paths are replicated and content is highly similar). If, in step **1664**, not all tests are SF or TF or NS, the host pair is assigned level **4** similarity (i.e., for replicated paths, content is similar). If, in step **1668**, no test shows TF and not all show SF, the host pair is assigned level **5** similarity (i.e., all paths are replicated but content may not be similar). If, in step **1672**, all tests show TF, SF, or NS the hosts are not mirrors.

The mirror categories of FIG. **17** are useful in sophisticated applications where a determination of mirror/no mirror may not be sufficient.

While the invention has been described in conjunction with a specific embodiment, it is evident that many alternatives, modifications and variations will be apparent to those skilled in the art in light of the foregoing description. For example, term vectors can be built according to several

methods at once, or methods can be combined in some combination of serial and/or parallel combinations. Accordingly, it is intended to embrace all such alternatives, modifications and variations as fall within the spirit and scope of the appended claims and equivalents.

What is claimed is:

1. A method of determining mirrored web hosts, comprising:

receiving information about the addresses of a plurality of web sites stored on a plurality of hosts;

determining a plurality of terms of the URLs associated with every host; weighting the terms in inverse proportion to frequency;

determining a similarity score for host pair in accordance with the weighted terms; and

outputting a list of potential pairs of mirrored hosts in accordance with their similarity scores.

2. The method of claim **1**, wherein the terms include "n-grams" having two or more contiguous tokens from the URLs.

3. The method of claim **1**, wherein the terms include "n-grams" having two or more continuous tokens from the URLs and URL depth information for the tokens.

4. The method of claim **1**, wherein the terms include paths of the URLs.

5. The method of claim **1**, wherein the terms include substrings of the paths of the URLs.

6. The method of claim **1**, wherein the terms are placed in a <term,host> tuple and the tuples are sorted by the terms.

7. The method of claim **1**, wherein the terms include prefixes of the URLs.

8. The method of claim **1**, wherein the similarity scores are normalized.

9. The method of claim **1**, further comprising accessing pages on each host corresponding to paths selected from both hosts to determine whether the hosts are mirrored hosts.

10. The method of claim **9**, wherein the hosts are mirrored if at least a predetermined number of path matches occurs for the selected paths.

11. The method of claim **1**, further comprising:

attempting to access selected pages on each host of a host pair, the pages corresponding to paths selected from each host, to determine whether the hosts are mirrored hosts;

categorizing the pages as having various matching categories; and

categorizing the host pairs in one of a plurality of similarity categories in accordance with the matching categories of the selected pages.

12. The method of claim **11**, wherein the matching categories indicate one of: access of the path on a source host failed; access of the path on a target host failed; content is byte-wise identical; documents are 100% similar after removal of; common content above a threshold for high similarity; and path is valid but no similarity.

13. The method of claim **11**, wherein the host pairs are divided into five similarity categories in accordance with the matching categories of the pages.

14. An apparatus that determines mirrored web hosts, comprising:

software configured to receive information about the addresses of a plurality of web sites stored on a plurality of hosts;

software configured to determine a plurality of terms of the URLs associated with every host;

software configured to weight the terms in inverse proportion to frequency;

software configured to determine a similarity score for host pair in accordance with the weighted terms; and

software configured to output a list of potential pairs of mirrored hosts in accordance with their similarity scores.

15. The apparatus of claim **14**, wherein the terms include "n-grams" having pairs of contiguous tokens from the URLs.

16. The apparatus of claim **14**, wherein the terms include "n-grams" having pairs of continuous tokens from the URLs and URL depth information for the tokens.

17. The apparatus of claim **14**, wherein the terms include paths of the URLs.

18. The apparatus of claim **14**, wherein the terms include substrings of the paths of the URLS.

19. The apparatus of claim **14**, wherein the terms include prefixes of the URLs.

20. The apparatus of claim **14**, wherein the similarity scores are normalized.

21. A computer program product, comprising:

computer program code devices configured to receive information about the addresses of a plurality of web sites stored on a plurality of hosts;

computer program code devices configured to determine a plurality of terms of the URLs associated with every host;

computer program code devices configured to weight the terms in inverse proportion to frequency;

computer program code devices configured to determine a similarity score for host pair in accordance with the weighted terms; and

computer program code devices configured to output a list of potential pairs of mirrored hosts in accordance with their similarity scores.

* * * * *