Mirror, mirror on the Web: a study of host pairs with replicated content

Krishna Bharat [*,1], Andrei Broder [1]

Compaq Systems Research Center, 130 Lytton Avenue, Palo Alto, CA 94301, USA

Abstract

Two previous studies, one done at Stanford in 1997 based on data collected by the Google search engine, and one done at Digital in 1996 based on AltaVista data, revealed that almost a third of the Web consists of duplicate pages. Both studies identified mirroring, that is, the systematic replication of content over a pair of hosts, as the principal cause of duplication, but did not further investigate this phenomenon. The main aim of this paper is to present a clearer picture of mirroring on the Web. As input we used a set of 179 million URLs found during a Web crawl done in the summer of 1998. We looked at all hosts with more than 100 URLs in our input (about 238,000), and discovered that about 10% were mirrored to varying degrees. The paper presents data about the prevalence of mirroring based on a mirroring classification scheme that we define. There are numerous reasons for mirroring: technical (e.g., to improve access time), commercial (e.g., different intermediaries offering the same products), cultural (e.g., same content in two languages), social (e.g., sharing of research data), and so forth. Although we have not done a exhaustive study of the causes of replication, we discuss and provide examples for several representative cases. Our technique for detecting mirrored hosts from large sets of collected URLs depends mostly on the syntactic analysis of URL strings, and requires retrieval and content analysis only for a small number of pages. We are able to detect both partial and total mirroring, and handle cases where the content is not byte-wise identical. Furthermore, our technique is computationally very efficient and does not assume that the initial set of URLs gathered from each host is comprehensive. Hence, this approach has practical uses beyond our study, and can be applied in other settings. For instance, for Web crawlers and caching proxies, detecting mirrors can be valuable to avoid redundant fetching, and knowledge of mirroring can be used to compensate for broken links. © 1999 Published by Elsevier Science B.V. All rights reserved.

Keywords: Mirroring; Content duplication; Smart proxies; Smart crawlers; Web statistics

1. Introduction

What exactly is mirroring? The term is often used in the Web research literature but a crisp definition is hard to come by. At one extreme one can say that two sites are mirrored if their content is byte-wise identical. In practice this definition is too restrictive: even on successive accesses to the same URL the fetched content may differ slightly because of dynamic components, timestamps, transaction-ids, etc. At the other extreme, one can say that two sites are mirrors if enough pages on one are very similar to pages on the other. This definition does not address the issue of structure. When two sites have different structures, the proof offered by content similarity is not compelling. For instance, consider the Web sites of *New York Times* and *Washington Post* (two major US newspapers). On any given day, they are likely to have some syntactically similar pages, e.g., because

[*] Corresponding author.

[1] E-mail: {bharat,broder}@pa.dec.com

they draw articles from common sources, or because they publish official documents. Nevertheless, the two sites can hardly be called mirrors.

Hence, we define two hosts to be mirrors if:

(1) A high percentage of *paths* (that is, the portions of the URL after the hostname) are valid on both Web sites, and

(2) These common paths link to documents that have similar content.

Therefore, hosts that replicate content but rename paths are not considered mirrors under our definition.

Two previous studies, one from Digital [3], based on 30 million pages collected by the **AltaVista**[2] search engine during 1996, and one from Stanford [13], based on 26 million pages collected by the **Google**[3] search engine [2] during 1997, point out the large amount of page duplication on the Web. The Digital study reported 30% duplication, and the Stanford study about 36% duplication. Both studies identified mirroring, that is, the systematic replication of content over a pair of hosts, as the principal cause of duplication, but did not shed further light on this phenomenon. The main aim of our investigation, based on a set of 179 million URLs, obtained from a Web crawl in the summer of 1998, is to present a clearer picture of mirroring on the World Wide Web.

Among the hosts represented in our input list of URLs roughly 238,000 hosts had contributed 100 URL samples or more. We looked for mirrors only in this set of *well represented* hosts. We discovered that about 10% of the hosts in the set were mirrored by other well represented hosts to varying degrees. In the next section we make this notion more precise, by defining several degrees of mirroring, and in Section 4 we present data about the prevalence and extent of mirroring according to our classification.

There are numerous reasons for mirroring: *technical* (e.g., replication to improve access time), *commercial* (e.g., different intermediaries offering the same products), *cultural* (e.g., same content in two languages), *social* (e.g., database of shared research), etc. Although we have not done a comprehensive study of the reasons behind mirroring, we present several cases that seem representative.

The technique we devised to detect mirrored hosts from large data sets depends mostly on the syntactic analysis of URL strings and requires fetching and content analysis only for a small number of pages. We use probabilistic tests for establishing the degree of mirroring. This makes our technique computationally very efficient. We are able to detect both partial and total mirroring, and handle cases where the content is not exactly identical. Furthermore, our strategy does not assume that the initial set of URLs from each host is comprehensive. Hence, our technique has practical uses beyond our study, and can be applied to other settings. For example, from the point of view of Web crawlers and caching proxies, detecting mirrors can be valuable to avoid redundant fetching and knowledge of the existence of mirrors can be used to compensate for broken links.

The paper proceeds as follows: in Section 2 we establish a classification of mirroring; Section 3 describes our approach to detecting and classifying mirrored hosts; Section 4 presents data from our experiment; Section 5 discusses motives for mirroring; Section 6 presents other applications of this technique; Section 7 mentions related work and in Section 8 we draw some conclusions.

2. A classification of mirroring

We observed various degrees of mirroring. To classify them, it is helpful to distinguish between *structure* and *content*. Structure is defined by the set of valid paths relative to the host under consideration. If two hosts have exactly the same set of paths, we say that they are *structurally identical*. With regard to content, we say that two pages are *content identical* if they are byte-wise equal. However during mirroring, pages often change at the byte level (e.g., by the addition of blank lines, by HTML reformatting, etc.) without any change of content. Hence we say that two pages are *content equivalent* if they have identical content after such normalizations. If pages change in content (e.g., due to a banner ad or other forms of dynamic content) but remain highly similar at the syntactic level, we call them *highly similar*. High similarity may be defined based on the edit distance or some other suitable measure. In this paper we use the similarity

[2] http://www.av.com/

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

measurement technique we developed in [3], with a suitable threshold. Finally, if pages change substantially at the syntactic level but are semantically similar (e.g., translated content), we call them *related*. This leads to the following classification of mirroring levels:

Level 1: Structural and content identity.
Every page on host A with relative path `P`, (i.e., a URL of the form `http://A/P`) is represented by a byte-wise identical page on host `B`, at location `http://B/P`, and vice versa.

Level 2: Structural identity. Content equivalence.
Every page on host `A` with relative path `P`, is represented by an equivalent content page on host `B`, at location `http://B/P`, and vice versa.

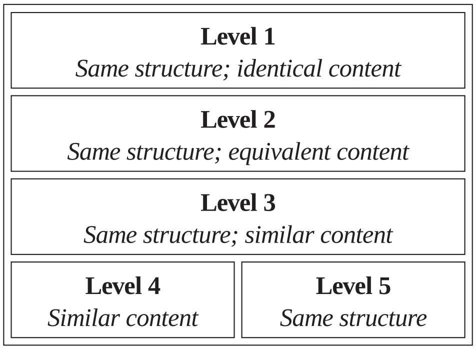
Level 3: Structural identity. Content similarity.
Every page on host `A` with relative path `P`, is represented by a highly similar page on host `B`, at location `http://B/P`, and vice versa.

Level 4: Partial structural match. Content similarity.
Some pages on host `A` with relative path `P`, are represented by a page on host `B`, at location `http://B/P`, and vice versa, and these pairs of pages are highly similar.

Level 5: Structural identity. Related content.
Every page on host `A` with relative path `P`, is represented by a page on host `B`, at location `http://B/P`, and vice versa. The pages are pairwise related (e.g., every page is a translation of its counterpart) but in general are not syntactically similar.

Mismatch: None of the above.

This induces the partial order shown below. Level 4 and Level 5 are not comparable.



Level 1
Same structure; identical content
Level 2
Same structure; equivalent content
Level 3
Same structure; similar content
Level 4
Similar content
Level 5
Same structure

Another factor that can be used to detect and classify mirrors is their IP address: mirrors often have related IP addresses. A DNS lookup provides one or more IP addresses for each host. If two hostnames map to a common IP address it indicates that they fetch pages from a common machine. However, this does not confirm a mirroring relationship. This is because a hostname may map to many IP addresses on different machines, and a given IP address may be used to implement virtual hosts that do not have a mirroring relationship. A DNS match helps build confidence about mirroring, but does not confirm it. We do not consider server IP address in our classification scheme.

3. Technique

Our technique allows us to detect pairs of mirrored hosts given a large set of URLs. The set need not be comprehensive, that is, not all paths on all hosts need to be present. This is typical of the results of even large crawls because no crawl is exhaustive due to the rapid rate of growth and change on the Web. Hence, the trivial solution of simply counting the number of common paths among hosts often does not work. Our procedure for detecting mirrored host pairs consists of two stages:

Stage I: Candidate Pair Detection.
In this stage we consider the URLs available from each of the hosts in our database, sample a subset of these URLs and compute syntactic similarity between these subsets. As explained below, we use coordinated sampling to increase the likelihood of selecting the same paths from hosts that happen to be mirrors. The output from this stage is a list of host pairs that are potential mirrors, ordered by likelihood.

Stage II: Host Pair Classification.
In this stage we process the list of candidate host pairs from Stage I, and test in each case if the hosts are indeed mirrors and estimate the extent of their overlap.

Note that our tests are probabilistic in nature. That is, it is conceivable that we may fail to recognize a pair of mirrored hosts in Stage I, or we may err in measuring the degree of overlap between them in Stage II.

The experimental evidence however suggests that the tests are fairly reliable.

We describe each of the stages next.

3.1. Candidate pair detection

In this stage we compute a set of *features* for each host, and search for pairs of hosts that share many features in common. As mentioned above, using full paths as features is not practical, since we assume that our information with respect to any given host is incomplete (i.e., not all paths are available). Instead we use fragments of the path and the hostname as features. The intuition here is that even if identical paths are not available from a host and its mirror, the URL strings will at least share some segments in common. This is a reasonable assumption because URL paths usually represent the directory structure within which the Web pages are located. Consequently, we can expect prefixes representing top level directories to be shared by many URLs. One possibility would be to use prefixes as features. However, it is often the case that two or more top level directories share the same naming scheme for the directories under them. For example, the paths:

```
dec97/sales/charts/...,
feb98/sales/charts/...,
aug98/sales/charts/...
```

sampled from different hosts, would fail to be correlated if one considered prefixes as features. Therefore, as we explain below, we use pairs of consecutive directories' names along the path (called *word bigrams*) as features. Matching *word bigrams* across hosts proves to be very effective.

For large data sets (179 million URLs in our case) reducing the number of features to be considered is a priority. One way to do this is to ignore hosts which 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. We decided to only consider hosts that had at least 100 URLs in our collection. A further data reduction is obtained as follows: for every host we first sort the list of URLs, and then only consider those paths whose strings upon hashing yielded a value which is 0 mod m. 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. Initially, we take m to be 5. 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 we sample more paths from larger sites than from smaller sites, but on the other hand large sites do not dominate the list of features.

3.1.1. Feature generation

The hostname and the set of paths derived from a given host as described above, contribute a set of features associated with host. Each string is converted to a list of terms by:

- *Converting to lowercase*. Paths may or may not be case sensitive, but hostnames are not. We ignore case for both.
- *Treating sequences of non-alphabetical characters as word-breaks*. This gives a list of words. Thus, `www7.infoseek.com` and `www6.infoseek.com` will yield the same list: `(www,infoseek,com)`.

We generate *word bigrams* by treating every contiguous pair of terms as a features. For example, the hostname `www.infoseek.com` generates:

`(www,infoseek)` and `(infoseek,com)`.

In fragmenting the path we also associate depth information. This gives us *positional word bigrams*. For example:

```
/cellblock16/inmates/dilbert/
  personal/foo.htm
```

gives

```
(cellblock,inmates,0)(inmates,dilbert,1)
  (dilbert,personal,2)(personal,foo,3)
  (foo,htm,4)
```

as features. Position is useful because we are trying to find mirror sites that share the same path structure.

Finding mirror sites that both mirror content and rename paths is a harder problem, which we do not address.

Finally, to reduce the set of features to be considered and reduce noise in the matching process we do three things:

(1) Eliminate stop terms such as: `htm`, `html`, `txt`, `main`, `index`, `home`, `bin`, `cgi`.

(2) Avoid URLs that have terms like `nph`, `dynaweb`, and `zyview`, which are characteristic of Web sites created automatically by tools. Such sites lead to spurious matches because they use a standard naming scheme.

(3) Eliminate path features that do not occur at least twice on the host. This eliminates leaf items in the path, unless they are part of a pattern.

3.1.2. Feature matching

For each valid feature we write the tuple `<feature,host>` to a file. At the end of the run, the file of tuples is sorted by the first element of the tuple, namely the feature. For any given feature, this causes all hosts that contain the feature 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 features. In our feature matching scheme, features are weighted in inverse proportion to the number of hosts they occur in. This corresponds to the usual `IDF` (inverse document frequency) weighting used in information retrieval. Features that occur in many hosts are considered too common to be significant, and correspondingly their `IDF` weight is low. As an optimization, we skip all features that occur within more than 20 hosts. For the rest, for every feature `f`, we consider each pair of hosts that share it and increment the $\mathtt{Similarity\_Score(Host_1,\ Host_2)}$ by the weight of the feature, `FW(f)`.

`FW(f)` is defined thus:

$$\mathtt{FW(f)} = \mathtt{S(f)/N(f)},$$

where `S(f)` is a measure of the significance of `f` independent of its distribution, and `N(f)` is the number of hosts in which `f` occurs. We set `S(f)` to 4 for host features and to 1 for path features, since a host feature match offers evidence that the two sites are part of the same organization (given that we consider only rare features).

The *Normalized Similarity Score* of each pair $\mathtt{(Host_1,\ Host_2)}$ is computed thus:

$$\mathtt{Normalized\_Score(Host_1,Host_2)} = \frac{\mathtt{Similarity\_Score(Host_1,Host_1)}}{\mathtt{1\ +\ 0.1\ *\ (Log(N_1)+Log(N_2)\ )}}$$

$\mathtt{N_1}$ and $\mathtt{N_2}$ represent the number of URLs in the input from $\mathtt{Host_1}$ and $\mathtt{Host_2}$ respectively, and are assumed to be proportional to their sizes. The denominator helps normalize the score by host size, to compensate for the fact that large hosts will have more feature matches than small hosts. Our parameters appear to perform satisfactorily but may afford some tuning.

At the end of this run, the normalized similarity score of every pair of hosts that share a significant feature will have been computed. A list of `<score,host-pair>` tuples is written to a file. To reduce mismatches we filter out host pairs that have only one feature in common, and also every pair that does not have a path feature in common. The remaining host pairs are sorted in the descending order of normalized similarity score. This optional step reduces false positives at the risk of skipping some mirrored pairs. This results in the output from Stage I, namely a list of prospective mirroring hosts, in the decreasing order of probability.

3.2. Host pair classification

In Stage II, the list of sorted host pairs from the previous stage is processed, and we classify each host pair into one of many categories based on the classification of mirroring hosts discussed in Section 2.

To classify a host pair we estimate the fraction of the paths from one host that are valid on the other host, and the extent to which the pages referenced by the common paths are similar. Thus, there are two steps: (i) selecting paths to test, and (ii) checking for validity of paths and computing similarity between pages corresponding to the valid paths.

The first step is done only once per host. Since a given host may occur in many host pairs, the same paths are reused. For each pair of host pages, the two root pages (the null path) are always considered. Also, in our experiment, we selected 9 additional paths at random per host from the paths chosen in Stage I, giving a total to 10 path samples per host.

In the second step the selected paths are used to fetch documents from both hosts, namely the *source host* where the path is known to be valid, and the *target host*, where the validity of the path is to be tested. The GET could fail at the source or at the target. If we succeed in fetching both pages we check if the contents are the same. This can be done at various levels of tolerance to change. In the strictest instance, this can be done by fingerprinting both documents (i.e., computing a checksum, such that with high probability any two distinct documents will have different checksums). However, in practice we find that content is often non fingerprint-identical. This can happen even on successive GETs to the same server, due to variable server-side includes 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 includes. To compensate for this we de-tag the content, ignore whitespace and check for syntactic similarity, that is, closeness of textual content, instead of fingerprint equality. Following [3] we use the mathematical concept of 'resemblance' to capture the informal notion of syntactic similarity.

The resemblance, `r(A, B)`, of two documents `A` and `B` is defined as follows: first each document is transformed into a set of word *k*-grams, `S(A)` and `S(B)`, also called 'shingles'. Then we compute:

$$\mathtt{r(A,B)} = \frac{|\mathtt{S(A)} \cap \mathtt{S(B)}|}{|\mathtt{S(A)} \cup \mathtt{S(B)}|}$$

where `|S|` is the size of the set `S`.

(As explained in [3] the resemblance can be estimated using a fixed size 'sketch' for each document. For a large collection of documents (say 100 million) the size of this sketch is of the order of a few hundred bytes per document.) Clearly, the resemblance value, `r(A,B)`, is a number between 0 and 1, which we can use to express the similarity between `A` and `B` as a percentage.

By considering structural resemblance and/or content resemblance (with different similarity thresholds), the host pairs were classified into one of many levels. For our experiment we defined the threshold for *high similarity* as 50%. Additionally, we used a 100% threshold to check for *full similarity* and a 0% threshold for *trace similarity*. When all similarity checks fail, the pair is classified as a *Mismatch*.

Between any pair of hosts there are 19 Web page comparisons in all: a pair of root pages + 9 source pages sampled from each host. In each case one of the following outcomes is possible:

- **SF**: Source Failure. GET failed at source host.
- **TF**: Target Failure. GET failed at target host.
- **FM**: Fingerprint Match. Content is byte-wise identical.
- **FS**: Full similarity. The documents are 100% equivalent after de-tagging.
- **HS**: High similarity. Common content is above the threshold for high similarity.
- **TS**: Trace Similarity. A small (non-zero) portion of the document is common.
- **NS**: Path is valid, but no syntactic similarity.

Based on what we observe in the 19 tests, each host pair is assigned a classification level. The assignment criteria for various classification levels is summarized in Table 1.

See Section 2 for a definition of these levels of similarity. Note that our test for Level 5 similarity is stricter than the definition in Section 2, which does not require trace similarity testing. This criterion was added to compensate for the case when all paths seem valid on the target host because it returns its own error page for mismatches instead of a 404 HTTP result. Also note that our tests are only statistical tests for the level of mirroring inferred. It is entirely possible that the actual level of mirroring is different, although the evidence suggests that our tests are fairly reliable.

4. Experiment

As our input we used a sorted list of about 179 million URLs. These were found on Web pages obtained in a broad crawl of the World Wide Web, and included some whose validity had yet to be verified. Thus, the set contained several bogus URLs, misspellings and broken links. Surprisingly, some hostname misspellings occurred commonly enough that they appeared in the Stage I candidate list, paired with the correct version of

Table 1
Classification of host pairs

Classification	Criterion	Implication
Level 1	All tests show SF or FM and not all are SF	Same structure; identical content
Level 2	All tests show SF or FM or FS, and not all are SF	Same structure; equivalent content
Level 3	All tests show SF or FM or FS or HS, and not all are SF	Same structure; similar content
Level 4	Some tests show HS or FM	Structure is partially replicated; for replicate paths, content is similar.
Level 5	No test yields TF and at least one with TS	Same structure (since all paths are valid); some of the content appears related.
Mismatch	All tests result in TF or NS	Not similar
DNS failure	DNS lookup failed for one of the hosts	No information; presently inaccessible
Server failure	One of the two servers was inaccessible	No information; presently inaccessible

the hostname. For example, we found that the non-existent site, ‘`www.geocites.com`’ was paired with ‘`www.geocities.com`’, indicating that it is a common misspelling.

In Stage I, we selected 238,679 hosts to process as described in the previous section. By sampling on average 20.8 paths per host, we obtained about 2 million features that matched our selection criteria. Using the similarity weighting and host pair filtering technique described in Section 3.1 we produced a list of 29,336 host pairs to be validated at the end of Stage I (see Table 2).

Table 2
Details of the experiment

Input set of URLs	179,369,374
Stage I processing	
Hosts considered (with >100 links)	238,679
Paths sampled (∼20.8 per host)	4,970,064
Features considered	2,001,423
Number of candidate host pairs after Stage I	29,336
Stage II processing	
Total number of successful tests	21,820
Level 1 matches	9,619
Level 2 matches	700
Level 3 matches	406
Level 4 matches	6,011
Level 5 matches	189
Mismatches	4,895
Total number of unsuccessful tests	7,516
Server failure	3,977
DNS failure (invalid server name)	3,539
Mirrored host pairs found (Level 5 match or better)	16,925
Distinct hosts that are mirrors (Level 5 or better)	22,363 (9.4% of hosts)

In Stage II, not counting failures, we had 21,820 successful tests. Of these 16,925 were found to be mirrors and 4,895 were mismatches. We found 22,363 hosts with a Level 5 mirroring relationship or better. This corresponds to about 9.4% of our total set of large hosts.

4.1. Ranking evaluation

As can be seen from Table 2, Level 1 matches and Level 4 matches account for most of the positive matches that we saw. Fig. 1 shows the effectiveness of the ranking strategy used in Stage I. After eliminating all unsuccessful tests from the ranked list, the graph plots the percentage of host pairs that were classified in a certain way for various prefixes of the reduced ranked list (21,820 host pairs in all).

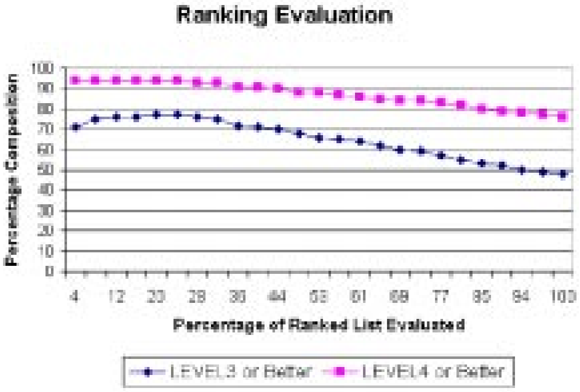


Fig. 1. The graph shows, for various cut-offs, the percentage of the successfully tested host pair list classified as: (a) Levels 1–3 (b) Levels 1–4.

E.g., on considering the top 49% of the candidate list (roughly 10,000 host pairs) we note that about 68% of the host pairs have a Level 3 similarity or better. This represents a set of hosts that can be used interchangeably from the point of view of most users. About 88% have a Level 4 similarity or better at this point, indicating that they have at least a partial mirroring relationship.

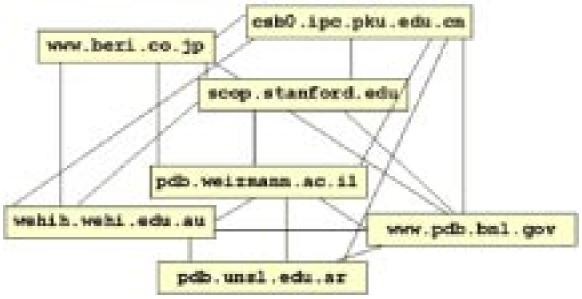


Fig. 2. Protein Data Bank Cluster. The edges indicate cross-domain Level 4 similarity between sites that have a mirrored copy of the *Protein Data Bank*.

4.2. Cross domain mirroring

Table 3 shows the distribution of cross-domain mirrors. These are pairs of mirrored hosts in which the domains of the two hosts differ. Cross-domain mirroring is interesting because it often reveals mirroring across organizational boundaries. In other cases it reveals mirroring for geographical reasons, within the same organization. The table shows the 10 most common domain combinations we encountered for various mirroring classifications.

When one considers hosts that are fully replicated (Levels 1–3), the most common case is when a ‘`com`’ site is mirrored as a ‘`org`’ site. Almost as frequent is the ‘`com-net`’ combination. We then see several instances of a ‘`com`’ site being mirrored under a country’s domain.

The ‘`com-edu`’ combination seems to occur more with partial replication (Level 4). This suggests the sharing of databases rather than a replication of the site’s main functionality.

Fig. 2 shows a cluster of research sites that we found to have Level 4 similarity because they all had copies of the *Protein Data Bank* (PDB). The PDB is an archive of experimentally determined three-dimensional structures of biological macromolecules, shared by researchers world-wide. Our experiment revealed almost all the pair-wise mirroring relationships between hosts containing the PDB database which appeared in the Stage I short-list. We discovered that there are some other PDB mirrors that did not make it to the short-list, either because they did not appear in our original list or because fewer than 100 URLs were retrieved from them. This suggests that a more comprehensive list of URLs and a lower inclusion threshold might have identified all of them.

With structural cross-domain replication (Level 5) we tend to find Web sites by the same organization mirrored in other languages or catering to local content. E.g., (`www.yahoo.ca`, `www.yahoo.co.jp`) or (`infoart.stanet.ru`, `infoart.baku.az`).

Table 3
Distribution of cross-domain mirrors

Rank based on frequency of occurrence	Full replication Levels 1–3	Partial replication Level 4	Structural replication Level 5
1	com-org (12.1%)	com-net (14.3%)	com-net (19.23%)
2	com-net (10.06%)	com-edu (3.08%)	ca-edu (19.23%)
3	com-de (5.17%)	com-org (3.08%)	edu-us (11.53%)
4	com-uk (4.89%)	de-edu (2.94%)	ca-com (11.53%)
5	com-fr (4.48%)	ca-com (2.32%)	az-ru (11.53%)
6	edu-org (3.8%)	com-de (2.19%)	com-sg (7.69%)
7	com-jp (3.53%)	au-com (2.12%)	cl-org (3.84%)
8	net-org (2.44%)	edu-uk (1.77%)	kz-ru (3.84%)
9	at-com (2.44%)	edu-jp (1.5%)	mx-org (3.84%)
10	com-kr (2.17%)	com-fr (1.5%)	ca-jp (3.84%)
Cross-domain mirrors	735	1461	26

4.3. Examples

The examples below were chosen to illustrate the various levels in our hierarchy. They behave as described at the time of writing. Since the Web is constantly changing, it is impossible to guarantee that they will continue to do so in the future.

Level 1 Examples (Cross Domain)

`www.boutiques-de-gestion.asso.fr` (20 samples)
`www.boutiques-de-gestion.com` (19 samples)
`www.ruskin-sch.milohedge.com` (20 samples)
`www.ruskin-sch.ox.ac.uk` (20 samples)
`www.upa.net` (20 samples)
`www.upaccess.com` (20 samples)

Level 2 Examples

These sites do virtual hosting from the same server, and have content that differs in minor ways:

`sys1.tpusa.com` (20 samples)
`www.i-trade.com` (20 samples)
`www.palladium.net` (20 samples)
`www.parroty.com` (20 samples)

Level 3 Examples

These are hosted by the same server but appear to be different because of dynamic content (a visitor counter):

`www.dancecorner.com` (20 samples)
`www.swingdance.com` (20 samples)

Level 4 Examples

These are mirror sites in French and English. They have the same path hierarchy but with translated content. This is characteristic of Canadian organizations with bilingual content. Some pages are common, so this does not appear to be Level 5:

`www.cbsc.org` (56 samples)
`www.rcsec.org` (63 samples)

These mirrors have identical content, but with French accented characters replaced with their HTML entity equivalents in one case. Consequently the pages that were compared appeared to have high or trace similarity rather than full similarity or fingerprint equivalence:

`bleue.ac-aix-marseille.fr` (20 samples)
`www.ac-aix-marseille.fr` (20 samples)

These are 2 different school sites, built with the same tool (Microsoft Communication Tool for Schools). Hence, they have a similar hierarchy, and layout, but differ in content:

`wchs02.washington.high.washington.k12.ga.us` (20 samples)
`www.sd70.bc.ca` (20 samples)

Although these have different IP addresses and their root pages are different, paths from one source seem to produce identical pages on the other. This led us to the discovery that *DesertNet* implements the newspaper, *Tucson Weekly*:

`www.desert.net` (111 samples)
`www.tucsonweekly.com` (66 samples)

Level 5 Examples

These two differ only in their encoding of Cyrillic. The content is the same:

`www-ibm866.consultant.ru` (20 samples)
`www-windows-1251.consultant.ru` (20 samples)

Two libraries that use the same query engine. Consequently queries on one are valid on the other:

`leagle.wcl.american.edu` (127 samples)
`library.cwu.edu` (102 samples)

Same content, mirrored in Spanish and English:

`tribute.lronhubbard.cl` (20 samples)
`tribute.lronhubbard.org` (18 samples)

5. Motives for mirroring

From extensive crawls of the World Wide Web we have an idea of how many Web sites there are out there. Earlier in 1998, **Alexa**[4] [11] recorded the existence of 500,000 Web sites and predicted a doubling of Web sites every 6 months. Understanding how much replication is going on and the reasons for it is necessary to understand the evolution of the World Wide Web.

We found that there are many incentives to replicate data at the present time:

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

(1) *Load balancing*: Replication decreases server load, e.g., www1.geocities.com and www14.geocities.com.
(2) *High availability*: The *Protein Data Bank* example presented in Section 4.2 is an instance of geographical mirroring for high availability.
(3) *Multi-lingual replication*: The same data is made available in many languages. For example several Canadian sites have mirrors that only differ in the language used, French or English.
(4) *Franchises/local versions*: This happens when content is licensed to other parties, e.g., quicken.excite.com and quicken.com have the same content, except for some site-specific branding.
(5) *Database sharing:* Two independent sites may share the same database or filesystem leading to mirroring, unintentionally, e.g., www.desert.net and www.tucsonweekly.com.
(6) *Virtual hosting:* These represent services that use the same IP address (and hence the same server) but implement two different Web sites based on the host name, e.g., sports.catalogue.com and www.accsports.com. In this case paths on one are valid on the other and yield identical pages. This need not always be so. In another case, www.autotune.com and www.borg-warner.com, the two virtual hosts share the same IP address but paths on one may not be valid on the other, and common paths lead to dissimilar content.
(7) *Maintaining pseudo identities*: Often, the incentive for such replication is to 'spam' search engines with seemingly different Web sites that in fact have the same content, hoping that one of them gets listed at the top of the ranking order, e.g., two 'adult' sites www.discountdvd.com and www.xxxpass.com.

6. Other applications

Our main reason for conducting this study was to further understand the geography of the Web. In addition, mirroring information is useful in the implementation of smart caching proxies and efficient crawlers. Levels 1 to 3, according to our classification, indicate pairs of hosts that can be used interchangeably. A proxy that maintains such a list of mirrors can serve a cached page from any of the mirrors of a given host, provided that the path is the same. A smart proxy can also try and compensate for broken links or server failure by transparently checking if the page is available on a mirror site. A crawler that tries to cover as much as the Web as possible in the shortest possible time can use mirroring information to avoid redundant crawling over mirrored parts of the Web. Also, in the interests of load balancing or in the interests of speed it can selectively download a path from an equivalent host.

Several recent ranking algorithms that rely on connectivity (e.g., [1,4,7]) treat hyperlinks within a host differently from hyperlinks across hosts. Hyperlinks in the latter case are considered indicative of quality, since they suggest a form of endorsement across organizations. Mirroring information has the potential to make these algorithms more robust and accurate, because it can identify organizational bounders that span many hosts.

Broken links on the World Wide Web are a common problem. The use of publishing tools and inter-server communication protocols has been advocated [5,6] to preserve referential integrity when links change. Until such a protocol is adopted globally, and legacy servers are retired, we will continue to see 'Page Not Found' errors. When a broken link refers to a known replicated host, mirroring information can be used to fetch an equivalent copy of the page.

To our surprise, we found that our scheme was even able to find 'pseudo mirroring' relationships involving misspelt hostnames (e.g., geocities.com misspelt as geocites.com). This suggests that given a large sample of user requests, as in a proxy log or a large crawl of the Web, this strategy might be able to correct frequent typographical errors.

7. Related work

The word bigram features that we select and the IDF based weighting scheme we use were influenced by standard feature extraction and classification practices in information retrieval (IR). See [14] for an overview of IR techniques.

Many researchers (e.g., [8,9]) have looked at the use of cooperating Web proxies to reap the benefits of caching, say within an intranet where a certain degree of access locality can be expected. Others (e.g., [15,16]) have studied hierarchical caching. On similar lines, one could imagine a set of cooperating proxies that periodically pool the sets of URLs they encounter to mine mirroring information that they can all use. This could be used to boot-strap a more pro-active scheme such as a mirror registry, where host administrators can register information about mirroring.

In some respects, our validation technique is similar to the one used in the **Ahoy!** [5] [12] home finder engine. Given a person's name and affiliation, Ahoy! generates various possible home page URLs and validates them by trying to fetch them. The templates for these URLs are computed by syntactic analysis of URL strings — specifically home page URLs encountered previously on the same host.

Statistics from the **Online Computer Library Center** [6] based on a sample of about 3000 hosts generated by random sampling of IP addresses (see [10]) lead to an estimate of the existence of 1.7 million public Web sites as of June 1998, of which about 12% were thought to be mirrored. Their result is not directly comparable to our 9.4% estimate since their study did not consider arbitrary mirrors, but only pairs of IP addresses that either map to the same hostname or share the first three octets. These pairs were tested to see if they have an HTTP server that returns identical root pages. On the other hand, we only considered mirroring relationship among the hosts that were well represented in our collection.

8. Conclusions

Previous studies of the Web have identified mirroring as a cause of duplication on the Web but have not analysed the mirroring issue in depth. In this paper we take the next step forward and describe a study to measure systematic replication among hosts. We started from a collection of 179 million unverified URLs found during a Web crawl from which we chose a set of 238,000 well represented hosts. We discovered that about 10% of these hosts were mirrored to various degrees within this set. We believe this to be a lower bound, since (a) not all Web hosts were represented and (b) taking more samples per host could in principle expose more mirroring. In the course of our study we identified several qualitatively different classes of mirroring and presented here characteristic examples from each class.

Our technique is efficient because it operates syntactically on URL strings and requires fetching only about 20 pages per prospective mirrored host for confirmatory content analysis. We advocate its use for detecting mirrors in Web proxies and crawlers in order to reduce redundant fetching and to improve caching behavior and reliability.

Acknowledgements

We would like to thank Jeff Dean, Steve Glassman, Monika Henzinger, Allan Heydon, Puneet Kumar, Mark Manasse, Hannes Marais, Mark Najork, and Jim Pitkow for help in conducting the study and writing the paper, and also our anonymous referees for valuable comments.

References

[1] K. Bharat and M.Henzinger, Improved algorithms for topic distillation in hyperlinked environments, in: Proc. ACM SIGIR '98, 1998, pp. 111–104, ftp://ftp.digital.com/pub/DEC/SRC/publications/monika/sigir98.pdf

[2] S. Brin and L. Page, The anatomy of a large-scale hypertextual Web search engine, in: Proc. 7th Int. World Wide Web Conference, Brisbane, Australia, April 1998, http://www7.scu.edu.au/programme/fullpapers/1921/com1921.htm

[3] A. Broder, S. Glassman, M. Manasse and G. Zweig, Syntactic clustering of the Web, in: Proc. 6th Int. World Wide Web Conference, 1997, http://decweb.ethz.ch/WWW6/Technical/Paper205/Paper205.html

[4] S. Chakrabarti, B. Dom, P. Raghavan, S. Rajagopalan, D. Gibson and J. Kleinberg, Automatic resource compilation by analyzing hyperlink structure and associated text, in: Proc. 7th Int. World Wide Web Conference, Brisbane, Australia, April 1998, http://www7.scu.edu.au/programme/fullpapers/1898/com1898.html

[5] M.L. Creech, Author-oriented link management, in: Proc. 5th Int. World Wide Web Conference, Paris, France, May

[5] http://ahoy.cs.washington.edu:6060/

[6] http://www.oclc.org/oclc/research/projects/webstats/statistics.htm

1996, http://www5conf.inria.fr/fich_html/papers/P11/Overview.html

[6] R.K. Jones and J. Pitkow, Supporting the Web: a distributed hyperklink database, in: Proc. 5th Int. World Wide Web Conference, Paris, France, May 1996, http://www5conf.inria.fr/fich_html/papers/P10/Overview.html

[7] J. Kleinberg, Authoritative sources in a hyperlinked environment, in: Proc. ACM SIAM Symp. on Discrete Algorithms, 1998, pp. 668 677, http://www.cs.cornell.edu/home/kleinber/auth.ps

[8] P. Krishnan and B. Sugla, Utility of co-operating Web proxy caches, in: Proc. 7th Int. World Wide Web Conference, Brisbane, Australia, April 1998, http://www7.scu.edu.au/programme/fullpapers/1892/com1892.htm

[9] R. Malpani, J. Lorch and D. Berger, Making World Wide Web caching servers cooperate, in: Proc. 4th Int. World Wide Web Conference, Boston, MA, December 1995.

[10] E.T. O'Neill, P.D. McClain and B.F. Lavoie, A methodology for sampling the World Wide Web, in: OCLC Annual Review of Research, Online Computer Library Center Inc., 1997, http://www.oclc.org/oclc/research/publications/review97/oneill/o'neillar980213.htm

[11] J. Selingo, Information technology: in attempting to archive the entire Internet, a scientist develops a new way to search it, in: The Chronicle of Higher Education, Washington D.C., March 6, 1998, http://chronicle.com/data/articles.dir/art-44.dir/issue-26.dir/26a02701.htm

[12] J. Shakes, M. Langheinrich and O. Etzioni, Dynamic reference sifting — case study in the homepage domain, in: Proc. 6th Int. World Wide Web Conference, 1997, http://ahoy.cs.washington.edu:6060/doc/paper.html

[13] N. Shivakumar and H. Garcia-Molina, Finding near-replicas of documents on the Web in: Proc. Workshop on Web Databases (WebDB'98), http://www-db.stanford.edu/~shiva/Pubs/web.ps

[14] K. Sparck-Jones and P. Willett, Readings in Information Retrieval, Morgan Kaufman, 1997.

[15] D. Wessels, Configuring hierarchical squid caches, Online Tutorial, http://squid.nlanr.net/Squid/Hierarchy-Tutorial/, 1997.

[16] P.S. Yu and E.A. MacNair, Performance study of a collaborative method for hierarchical caching in proxy servers, in: Proc. 7th Int. World Wide Web Conference, Brisbane, Australia, April 1998, http://www7.scu.edu.au/programme/fullpapers/1829/com1829.htm

Krishna Bharat is a member of the research staff at Compaq Computer Corporation's Systems Research Center in Palo Alto, California. His current research interests include Web content discovery and retrieval, user interface issues in Web task automation, and interaction with small devices. He received his Ph.D. in Computer Science from Georgia Institute of Technology in 1996, where he worked on tool and infrastructure support for building distributed user interface applications.

Andrei Broder has a B.Sc. from Technion, Israel Institute of Technology and an M.Sc. and Ph.D. in Computer Science from Stanford University. He is a member of the research staff at Compaq Computer Corporation's Systems Research Center in Palo Alto, California. His main interests are the design, analysis, and implementation of probabilistic algorithms and supporting data structures, in particular in the context of Web-scale applications.