A Comparison of Techniques to Find Mirrored Hosts on the WWW

Krishna Bharat
Google Inc.
2400 Bayshore Ave
Mountain View, CA 94043
bharat@google.com

Andrei Broder
AltaVista Company
1825 S. Grant St.
San Mateo, CA 94402
andrei.broder@av.com

Jeffrey Dean
Google Inc.
2400 Bayshore Ave
Mountain View, CA 94043
jeff@google.com

Monika R. Henzinger
Google Inc.
2400 Bayshore Ave
Mountain View, CA 94043
monika@google.com

Abstract

We compare several algorithms for identifying mirrored hosts on the World Wide Web. The algorithms operate on the basis of URL strings and linkage data: the type of information about web pages easily available from web proxies and crawlers.

Identification of mirrored hosts can improve web-based information retrieval in several ways: First, by identifying mirrored hosts, search engines can avoid storing and returning duplicate documents. Second, several new information retrieval techniques for the Web make inferences based on the explicit links among hypertext documents – mirroring perturbs their graph model and degrades performance. Third, mirroring information can be used to redirect users to alternate mirror sites to compensate for various failures, and can thus improve the performance of web browsers and proxies.

We evaluated 4 classes of "top-down" algorithms for detecting mirrored host pairs (that is, algorithms that are based on page attributes such as URL, IP address, and hyperlinks between pages, and not on the page content) on a collection of 140 million URLs (on 230,000 hosts) and their associated connectivity information. Our best approach is one which combines 5 algorithms and achieved a precision of 0.57 for a recall of 0.86 considering 100,000 ranked host pairs.

1 Introduction

The WWW differs from information retrieval in classical collections in many respects. Two of the most obvious are the very large size of the Web (the largest search engines have over a billion URLs in their index [2] as of mid 2000) and redundancy in the form of sources with the same or roughly the same information. The fraction of the total WWW collection consisting of duplicates and near-duplicates has been estimated at 30 to 45 percent. (See [6] and [15].)

Copyright 2000 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Duplication has both positive and negative aspects. On one hand the redundancy makes retrieval easier: if a search engine has missed one copy, maybe it has the other; or if one page has become unavailable, maybe a replica can be retrieved. On the other hand, from the point of view of search engines storing duplicate content is a waste of resources and from the user's point of view, getting duplicate answers in response to a query is a nuisance.

The principal reason for duplication on the Web is the systematic replication of content across distinct hosts, a phenomenon known as "mirroring" (These notions are defined more precisely below.) It is estimated that at least 10% of the hosts on the WWW are mirrored [3, 11]. The aim of this paper is to present and evaluate algorithms for detecting mirroring on the WWW in an information retrieval framework.

Our definition of mirroring is as follows:

> Two hosts A and B are mirrors iff for every document on A there is a *highly similar* document on B with the same path, and vice versa.

Highly similar is a subjective measure. We made this notion precise by adopting the resemblance distance described in [6, 5] that experimentally computes a number between 0 and 1 as a measure of similarity. Other edit distance measures can be substituted.

Reliable algorithms for finding host mirrors can improve web-based information access in several ways:

- Knowledge of replication can save search engines crawling and index space overhead.
- Recent web IR techniques analyze the linkage between pages in the web graph to rank document (e.g., PageRank [12], CLEVER [8, 7, 10], and Topic Distillation [4]) The independence assumption between subsections of the web graph is violated by mirroring and can perturb the outcome of such link based computations.
- Knowledge of alternate mirrors can help compensate for "broken links" on the WWW.
- Caches on the web can exploit knowledge of redundancy to increase their effective coverage.

The algorithms described in this paper use only the URLs of pages on the web and in some cases the hyperinkage between them (also called "connectivity"). In particular we do not require web page content, unlike the technique of Cho, Shivakumar and Garcia-Molina [9].

Our input corresponds to a graph in which the nodes are web pages and edges are hyperlinks. We require neither the set of nodes, nor edges to be complete. In particular a given host can be represented partially, with paths that differ from those sampled from its mirrors. Such incompleteness is typical of the graphs computable from web crawls or proxy usage logs.

In this paper we discuss and evaluate four classes of algorithms for finding mirrored hosts on the Web. Each algorithm produced a ranked list of distinct host pairs, each pair representing a possible mirroring relationship. The validity of each mirroring relationship was then judged in an automated way. This was done by fetching from the web a subset of pages from each host and testing to see if the other host had a page with an identical path with highly similar content. The host pair was considered valid iff all of 40 such tests succeeded. We then used the classic IR measures of precision and recall to graph the results. The best approach we found was a combination of 5 algorithms, which on our test data achieved a precision of 0.57 for a recall of 0.86 considering 100,000 result host pairs.

Section 2 presents a brief description of our algorithms. The evaluation methodology and performance of algorithms is summarized in section 3. Section 4 discusses related work. For more details see our full paper [1].

2 Ranking Algorithms

In our experiments the input was a subset of an approximately 179 million URL crawl by AltaVista. We chose a subset of 140.6 million URLs that represented hosts with at least 100 pages in our input, corresponding to 233,035 hosts. The URL strings and hyperlinkage required 11.2 Gigabytes uncompressed.

Only mirroring relations between host pairs drawn from this set were determined.

We used four classes of algorithms to compute a ranked list of host pairs representing potential mirrors. These are summarized below (see full paper for details and rationale [1]:

2.1 IP Address Based (Algorithms *IP3* and *IP4*)

These algorithms list hosts that have identical or highly similar IP addresses. Identical or highly similar IP addresses are indicative of 2 hosts on the same server or subnet - which can imply mirroring. However, when many hosts resolve to the same IP address it is indicative of virtual hosting by an internet service provider, which hosts many distinct web sites.

These considerations led us to two algorithms:

- Algorithm *IP4:* We cluster hosts with the same IP address. With a bias against virtual hosting, we process clusters in the increasing order of siz ande enumerate up to 200 host pairs from each cluster
- Algorithm *IP3:* Same as the above case but we instead cluster based solely on the first three octets of the IP address and list at most 5 pairs per cluster.

2.2 URL String Based (Algorithms *hosts, paths, prefix* and *shingles*)

In this class of algorithms we detect similarity between either the hostnames or the path structure (which usually refects the directory structure) of the hosts being compared. In each algorithm a set of features (or “terms”) is selected from each host. Term vector matching [14] is used to compute the likelihood that a pair of hosts are mirrors based on weights of terms in common. Based on the type of term used we get four algorithms, each with a weighting scheme. Only terms occurring in fewer than 100 documents were considered.

These are summarized below:

- Hostname Matching (Algorithm *hosts*): Substrings of the hostname by breaking only at periods ('.') are terms. The weight of term t is $\frac{\log(len(t))}{1+\log(df(t))}$ where $df(t)$ is the number of documents with the term, and $len(t)$ is the number of '.' separated segments.
- Full Path Matching (Algorithm *paths*): Entire paths of URLs are used as terms. The weight of term t is $1 + \log(\frac{maxdf}{df(t)})$, where $maxdf$ is the maximum value of $df(t)$ over all t.
- Prefix Matching (Algorithm *prefix*): Prefixes of the path that either end in a '/' or terminate the path are terms. The weighing is as in *path*, except, to remedy the bias against small hosts we multiply by $\frac{1}{0.1+0.15(\log(n_1)+log(n_2))}$, where n_1 and n_2 represent the number of URL strings in the input from each of the two hosts.
- Positional Word Bigram Matching (Algorithm *shingles*): The path is broken into a list of tokens by treating '/' and '.' as breaks. Pairs of tokens in sequence with ordinal position are used as terms. Non-alphabetical character sequences are stemmed to '*'. E.g., the path `conferences/sigweb/web2001/xyz` will have `<2:sigweb 3:web*>` as a term. The same weighting as in *prefix* is used.

2.3 URL String and Connectivity Based (Algorithm *conn*)

The identification of mirrors can be aided by the use of connectivity information. We extend the URL string based *paths* algorithm (which selects host pairs with many common paths), with a connectivity based filtering stage.

Given a host pair $< host_1, host_2 >$ from the output of *paths* we test if 0.9 of the union of their outlinks are common to both. An approximate measure is used because some outlinks may be missing in the input. Paths that qualify are said to be “equivalent”. After testing 20 common paths, if 0.75 of the paths are found to be equivalent then the host pair is allowed to remain in the ranking. Otherwise, it is eliminated.

2.4 Host Connectivity-Based (Algorithms *hconn1* and *hconn2*)

In this class of algorithms similarity between hosts is computed based on the similarity between the sets of other hosts they connect to.

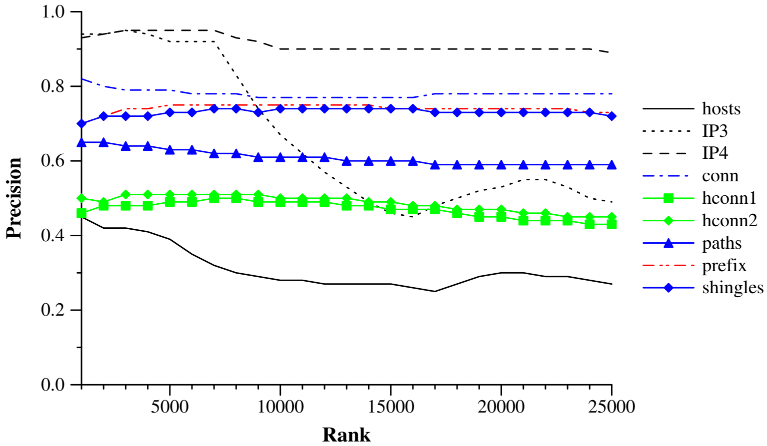

Figure 1: Precision at $rank$ for the algorithms

Again, term vector matching is used to compute host similarity. The terms in this case are hosts. Specifically, host t is a term in host x's vector if there exists at least one hyperlink from a page on x to a page on t. We denote the total number of such links by $f(t,x)$. For efficiency we keep only $10 \log P(x)$ terms with the highest value of $f(t,x)$, where $P(x)$ is the number of URLs sampled from the host x. The weight of each term however is a function of $in(t)$, which is computed by summing $f(t,y)$ for all hosts y that contain t in their term vector.

Algorithm *hconn1* weights term t proportional to $1/in(t)$, varying in the range $[\frac{1}{5} \cdots 1]$. Algorithm *hconn2* further multiplies by a factor $1 + \log(\frac{maxin}{in(t)})$ where $maxin$ is the maximum value of the in function over all terms.

3 Evaluation

3.1 Methodology

As mentioned before each algorithm outputs a ranked set of host pairs. We evaluate whether a given host pair is indeed a mirror using an automated technique, and we call this a *correct* host pair. This allows us to use the standard information retrieval notion of precision and recall to compare algorithms.

- We define *precision at rank* k to be the fraction of correct host pairs given within the first k host pairs.
- We define *recall at rank* k to be the number of correct host pairs found within the first k host pairs divided by R, where R is the total number of distinct correct host pairs.

We computed these numbers as follows: Since some of our algorithms returned several million host pairs, human evaluation for mirroring was impossible. Hence to decide if a host pair was "correct" we sampled 20 random paths from each host and tested for the presence of a highly similar document on the other host. If all 40 tests succeeded the host pair was considered correct.

We evaluated the up to 30,000 host pairs from each ranking in this manner. Figure 1 graphs precision vs rank up to rank 25,000. The best algorithms were *IP4*, *prefix*, *shingles* and *conn*. We found that *IP3* rapidly deteriorated in precision, and *hosts* was too naive. Connectivity analysis in *conn* improved *paths* significantly, but did not help as much with host connectivity (*hconn1* and *hconn2*). *Hconn1* and *hconn2* tended to pairs hosts on the same topic rather than mirrors.

The total pool of mirrors consisted of 41,388 host pairs at rank 25,000. The best single algorithm *IP4* was able to find only 53.9% of the known mirrors. Further, looking at the overlap between the pairs found indicated that algorithms in different classes were finding complementary sets of mirrors. Hence, we introduced a new algorithm *combined* which merged the top 100,000 results from the output of five different algorithms *IP4*, *prefix*, *hconn2*, *paths* and *hostnames*. The host pairs in the resulting union were sorted in the decreasing order of

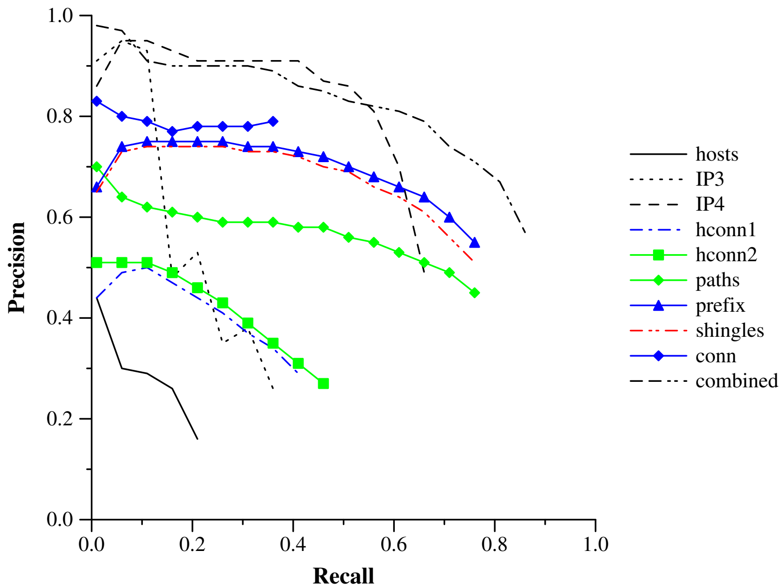

Figure 2: Precision vs Recall

the number of algorithms that found them. Ties between host pairs were resolved in favor of the host pair with the lowest rank in any of the five algorithms.

Figure 2 shows a precision vs recall plot. For this graph we considered the top 100,000 results from each ranking. Host pairs not seen previously were considered correct if and only if they appeared in the transitive closure of the set of known correct host pairs - which we took to be the universe of correct host pairs.

We found that *IP4*'s precision drops after a recall of 0.5. With increasing cluster size IP address matching becomes progressively less reliable. *Prefix* is the best of the individual algorithms, achieving a precision of 0.49 for a recall of 0.8 (*shingles* performs similarly). *Conn* seems potentially better but since we were unable to run the connectivity filtering on the output of *paths* beyond rank 30,000, it was incompletely evaluated. The combined algorithm *combined* had the best performance overall. In all there were 66,337 distinct correct host pairs in our set, and *combined* found 57,000 of them at rank 100,000 (precision=0.57, recall=0.86).

4 Related Work

The problem of computing mirrored host pairs on the WWW was introduced in [3]. They use a more generalized definition of mirroring, which includes *partial mirroring* as well. They tested a single algorithm corresponding to a weighted combination of our *hosts* and *shingles* algorithms, but did not investigate the use of IP addresses or connectivity in finding mirrors. Since we were interested in a comparative study of algorithms we extended their validation scheme to include the pooling of validated results from various algorithms, and also the use of transitive inferences to expedite validation.

At first sight the problem of finding mirrors seems like it could be cast as a clustering problem in which each host is represented by the contents of the documents it contains. There has been considerable prior work in document clustering (see e.g. [16, 13]). As mentioned earlier, owing to the scale of the Web comparing the full contents of hosts is neither feasible, nor (as we believe) necessarily more useful than comparing URL strings and connectivity. See also [17] for a discussion on the applicability of clustering algorithms to the WWW.

Connectivity information has been applied to the problem of improving precision of WWW search results [12, 8, 7, 10, 4]. The graph model used by these methods is perturbed by duplicate pages. We believe our *hconn** and *conn* algorithms represent the first attempt to use connectivity information to finding mirrors. Furthermore, these two algorithms can be easily integrated within the methods cited above to alleviate the above perturbation problem.

The "bottom up" approach taken by Cho et al in [9] requires content analysis and is more compute intensive. Also, it requires all URLs to have been crawled within a small window of time. Our techniques can operate solely with URL structure, which tends to be more stable. The two approaches have yet to be compared.

5 Conclusion

Finding mirrors on the WWW is an information retrieval problem of interest to the search engine and web caching communities. In this paper we evaluated 4 classes of algorithms for ranking potential mirrored host pairs, namely: (i) IP address based approaches, (ii) URL string comparison techniques, (iii) host connectivity based approaches, and (iv) an approach that compares connectivity of documents with shared paths. These were evaluated on a collection of 140 million URLs (on 230,000 hosts) and associated connectivity information. *IP4* and *prefix* were our best single algorithms. We concluded that single approaches are limited in terms of recall. Our best approach is one which combines 5 algorithms and achieves a precision of 0.57 for a recall of 0.86 considering the top 100,000 results.

References

[1] K. Bharat, A. Broder, J. Dean, M. Henzinger. A Comparison of Tehcniques to Find Mirrored Hosts on the WWW. *Journal of the American Society for Information Science (JASIS)* (To Appear, Nov 2000.)

[2] Google Inc. Google Launches World's Largest Search Engine Press Release, June 26, 2000. URL: http://www.google.com/pressrel/pressrelease26.html

[3] K. Bharat and A. Z. Broder. Mirror, mirror on the web: A study of host pairs with replicated content. In *Proceedings of the Eighth International World Wide Web Conference*, May 1999.

[4] K. Bharat and M. Henzinger. Improved algorithms for topic distillation in hyperlinked environments. In *Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'98)*, pages 111–104, 1998.

[5] A. Z. Broder. Filtering near-duplicate documents. In *Proceedings of FUN 98*, 1998.

[6] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the Web. In *Sixth International World Wide Web Conference*, pages 391–404, Santa Clara, California, April 1997.

[7] S. Chakrabarti, B. Dom, D. Gibson, S. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Experiments in topic distillation. In *SIGIR'98 Workshop on Hypertext Information Retrieval for the Web*, 1998.

[8] S. Chakrabarti, B. Dom, R. P., S. Rajagopalan, D. Gibson, and J. Kleinberg. Automatic resource compilation by analyzing hyperlink structure and associated text. In *Proceedings of the Seventh International World Wide Web Conference*, pages 65–74.

[9] J. Cho, N. Shivakumar, and H. Garcia-Molina. Finding Replicated Web Collections. In *ACM SIGMOD 2000*, pages 355–366.

[10] J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 668–677, January 1998.

[11] E. T. O'Neill, P. D. McClain, and B. F. Lavoie. A methodology for sampling the world wide web. Technical report, OCLC Annual Review of Research, 1997.

[12] S. Brin and L. Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine In *Proceedings of the Seventh International World Wide Web Conference*, pages 107-117, 1998.

[13] E. Rasmussen. Clustering algorithms. In W. Frakes and R. Baeza-Yates, editors, *Information Retrieval*, pages 419–42, 1992.

[14] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[15] N. Shivakumar and H. García-Molina. Finding near-replicas of documents on the web. In *Proceedings of Workshop on Web Databases (WebDB'98)*, March 1998.

[16] P. Willet. Recent trends in hierarchical document clustering: a critical review. *Information Processing and Management*, 24(5):577–597, 1988.

[17] O. Zamir and O. Etzioni. Web document clustering: A feasibility demonstration. In *Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'98)*, pages 46–53, 1998.