SOFTWARE—PRACTICE AND EXPERIENCE
Softw. Pract. Exper. 2004; **34**:213–237 (DOI: 10.1002/spe.577)

A large-scale study of the evolution of Web pages

Dennis Fetterly[1,*,†], Mark Manasse[1], Marc Najork[1] and Janet L. Wiener[2]

[1]*Microsoft Research, 1065 La Avenida, Mountain View, CA 94043, U.S.A.*
[2]*Hewlett-Packard Labs, 1501 Page Mill Road, Palo Alto, CA 94304, U.S.A.*

SUMMARY

How fast does the Web change? Does most of the content remain unchanged once it has been authored, or are the documents continuously updated? Do pages change a little or a lot? Is the extent of change correlated to any other property of the page? All of these questions are of interest to those who mine the Web, including all the popular search engines, but few studies have been performed to date to answer them.

One notable exception is a study by Cho and Garcia-Molina, who crawled a set of 720 000 pages on a daily basis over 4 months, and counted pages as having changed if their MD5 checksum changed. They found that 40% of all Web pages in their set changed within a week, and 23% of those pages that fell into the `.com` domain changed daily.

This paper expands on Cho and Garcia-Molina's study, both in terms of coverage and in terms of sensitivity to change. We crawled a set of 150 836 209 HTML pages once every week, over a span of 11 weeks. For each page, we recorded a checksum of the page, and a feature vector of the words on the page, plus various other data such as the page length, the HTTP status code, etc. Moreover, we pseudo-randomly selected 0.1% of all of our URLs, and saved the full text of each download of the corresponding pages.

After completion of the crawl, we analyzed the degree of change of each page, and investigated which factors are correlated with change intensity. We found that the average degree of change varies widely across top-level domains, and that larger pages change more often and more severely than smaller ones.

This paper describes the crawl and the data transformations we performed on the logs, and presents some statistical observations on the degree of change of different classes of pages. Copyright © 2004 John Wiley & Sons, Ltd.

KEY WORDS: Web characterization; Web evolution; Web pages; rate of change; degree of change

1. INTRODUCTION

The searchable Web and the search engines which survey it have become indispensable tools for information discovery. From academic researchers to elementary-school students, from cancer patients to pensioners, from local high-school football fans to international travellers, the indexed content of the Web is becoming the primary research tool for many. With hundreds of millions of people relying

[*]Correspondence to: Dennis Fetterly, Microsoft Research, 1065 La Avenida, Mountain View, CA 94043, U.S.A.
[†]E-mail: fetterly@microsoft.com

Received 31 March 2003
Revised 31 July 2003
Accepted 5 August 2003

on these tools, one is led to ask if the tools provide useful, up-to-date results. Ideally, one would like the entire index of a search engine to be *fresh*, that is, to contain the most up-to-date version of a Web page. The Google search engine attempts to maintain a fresh index by crawling over 3 billion pages once a month [1], with more frequent crawls of hand-selected sites that are known to change more often. In addition, it offers access to cached copies of pages, to obviate problems arising from some of the crawled, uniform resource locators (URLs) being out-of-date or having disappeared entirely.

To improve the freshness of results returned by search engines and allow them to spend more of their efforts crawling and indexing pages which have changed, it is interesting and important to answer some questions about the dynamic nature of the Web. How fast does the Web change? Does most of the content remain unchanged once it has been authored, or are the documents being continuously updated? Do pages change a little or a lot? Is the extent of change correlated to any other property of the page? Do pages change and then change back? How consistent are mirrors and near-mirrors of pages? Questions like these are of great relevance to search engines, and more generally to any party trying to maintain an up-to-date view of the Web, but they are also interesting in their own right, as they shed light on the evolution of a major sociological phenomenon: the largest collectively constructed information repository known to man.

In this paper, we attempt to answer some of these questions. We recount how we collected 151 million Web pages 11 times over, retaining salient information including a feature vector of each page. We describe how we distilled the collected information about each URL into a summary record, tabulating the feature vectors. We sketch the framework we used to mine the distilled data for statistical information. We present the most interesting results of this data mining. Finally, we draw some conclusions and offer avenues of future work.

2. RELATED WORK

This paper is an extended version of our earlier paper [2]. It expands on a study by Cho and Garcia-Molina [3]. The authors of that study downloaded 720 000 pages drawn from 270 'popular' Web servers (not exceeding 3000 pages per server) on a daily basis over the course of 4 months, and retained the MD5 checksum of the contents (including the HTML markup) of each page. This checksum allowed them to determine if a document had changed, although it did not allow them to assess the degree of change. Among other things, they found that pages drawn from servers in the `.com` domain changed substantially faster than those in other domains, while pages in the `.gov` domain changed substantially slower. Overall, they found that about 40% of all Web pages changed within a week, and that it took about 50 days for half of all pages to have changed. They also found that almost 25% of the pages in `.com` changed within a day, and that it took 11 days for half of all `.com` pages to have changed. In contrast, it took 4 months (the duration of their study) for half of the `.gov` pages to have changed.

Sun *et al.* [4] conducted a study on the efficacy of Web anonymizers. As part of that study, they drew a set of 100 000 Web pages from the Open Directory listing, and crawled each page twice, with the second retrieval immediately following the first one. Since they were interested in the information leakage of encrypted channels, they did not compare checksums of the returned pages; rather, they compared the lengths of the pages and the number and lengths of their embedded images and frames, which will appear to an eavesdropper as temporally closely spaced Transmission Control Protocol (TCP) packets. They found that 40% of pages changed signatures, and that 14% of pages changed by

30% or more, using a Jaccard-coefficient-based similarity metric akin to the one we use, but based on the size of a document including sizes of embedded images, not contents.

Douglis *et al.* [5] studied a Web trace consisting of 950 000 records (each record representing a Web access) collected over 17 days at the gateway between AT&T Labs–Research and the Internet. They recorded the 'last-modified' timestamp transmitted by the Web server (implicitly assuming that Web servers transmit accurate information). In addition, they mined each page for items such as phone numbers using a domain-specific semantic analysis technique called 'grinking', and measured the rate of change of these items. They found that according to the last-modified metric, 16.5% of the resources (including HTML pages as well as other content, such as images) that were accessed multiple times changed every time they were accessed. They also found that among the HTML pages that were accessed more than once, almost 60% experienced a change in HREF links, and over 50% of them experienced a change in IMG links.

Brewington and Cybenko, who built a Web clipping service, leveraged this service to study the rate of change of Web pages [6]. They did so by recording the last-modified time stamp, the time of download, and various stylistic attributes (number of images, links, tables, etc) of each downloaded HTML page. Their service downloaded about 100 000 pages per day, selected based on their topical interest, recrawling no page more often than once every 3 days. They evaluated data collected between March and November 1999. For pages that were downloaded six times or more, 56% did not change at all over the duration of the study (according to the features they retained), while 4% changed every single time.

Our study differs from previous studies in several respects. First, it covers a roughly 200 times larger portion of the Web (although the interval between revisits is seven times larger than, say, the one used by Cho and Garcia-Molina). Second, we used a different and more fine-grained similarity metric than any of the other studies, based on *syntactic document sketches* [7]. Third, we selected our pages based on a breadth-first crawl, which removed some of the bias inherent in the other studies (although breadth-first crawling is known to be biased towards pages with high PageRank [8])[‡]. Fourth and finally, we retained the full text of 0.1% of all downloaded pages, a sample set that is comparable in size to the set of pages summarized by other studies.

3. EXPERIMENTAL SETUP

Our experiment can be divided into three phases: collecting the data through repeated Web crawls, distilling the data to make it amenable to analysis, and mining the distillate. This section describes each of these phases in more detail.

3.1. Collecting the data

Between 26 November 2002 and 5 December 2002, we performed a large crawl ('crawl 1') that downloaded 151 million HTML pages as well as 62 million non-HTML pages, which we

[‡] Alternatively, we could have attempted to select a set of URLs uniformly at random, or we could have selected a set of Web pages according to their popularity with real Web users. However, generating a uniform random sample of URLs is notoriously difficult, and we did not have access to a large collection of Web usage data.

subsequently ignored. We then attempted to download each of these 151 million HTML pages 10 more times over a span of 10 weeks. Naturally, some of these pages became either temporarily or permanently unavailable. Moreover, we experienced a catastrophic disk failure during the third crawl, causing us to lose a quarter of the logs of that crawl.

Our hardware infrastructure consisted of a cluster of four Compaq DS20 servers, each one equipped with a 667 MHz Alpha processor, 4 GB of RAM, 648 GB of disk, and a fast Ethernet network connection. The machines were located at the Palo Alto Internet Exchange, a peering point for 12 Telcos and 20 major and about 130 minor Internet service providers (ISPs).

We conducted these crawls using the Mercator Web crawler [9]. Mercator is both fast and highly configurable, making it a suitable tool for our purposes.

We seeded crawl 1 with the Yahoo! home page. We restricted ourselves to content retrievable using HTTP, ignoring HTTPS, FTP, Gopher, and the like. Mercator crawled using its standard breadth-first search Web traversal strategy, which is biased towards pages with high PageRank [8]. This crawl ran for 9 days, and downloaded a total of 6.4 TB of data. As said earlier, it logged the URLs of 151 million HTML pages that were successfully retrieved, i.e. that were returned with an HTTP status code of 200 and a content type of `text/html`.

We used these 151 million URLs to seed the following ten crawls. These crawls ran consecutively, starting on 5 December 2002 and ending on 12 February 2003. We disabled link extraction and HTTP redirections; in other words, we configured Mercator to download only the pages in its seed set. Every crawl slowed down by two orders of magnitude once it had processed all but a million or so seeds, because the remaining URLs all referred to Web servers that were extremely slow to respond to us during that crawl, and because Mercator's politeness policies cause it to space out requests to the same host proportional to the delay of the previous response from that host. In order to keep the overall duration of each crawl within reasonable bounds, we terminated the crawls after this happened, typically on the sixth day of the crawl, and started the next crawl a week after the start of the preceding one.

Figure 1 shows the distribution of successful and failed download attempts. The graph shows 11 sets of three bars. Considering set n, the first bar indicates the percentage of pages that were successfully downloaded exactly n times, or, phrasing it differently, the percentage of URLs that, when requested, produced an HTTP response code of 200 in n of the 11 weeks; the second bar indicates the percentage of URls for which an HTTP request received any response from the Web server in exactly n weeks; and the third bar indicates the percentage of URLs for which an HTTP request was issued in exactly n weeks.

There are two reasons why the third bar of set 11 does not extend all the way to 100%: first, we treat the pages that we lost during the catastrophic disk failure mentioned above as not having been requested during that week. Second, again as mentioned above, we terminated crawls 2–11 once the download rate fell below a certain threshold.

As can be seen, we succeeded in downloading 49.2% of the pages all 11 times, and 33.6% 10 times, leaving 17.2% of the pages that could only be downloaded nine times or fewer.

For all 11 crawls, we provided Mercator with two new processing modules: a module that recorded a checksum and a fixed-size feature vector plus some ancillary data for each page, and a second module that selected 0.1% of all pages and saved them to disk.

We computed the feature vectors using a modified version of the document shingling technique due to Broder *et al.* [7], which uses a metric of document similarity based on syntactic properties of

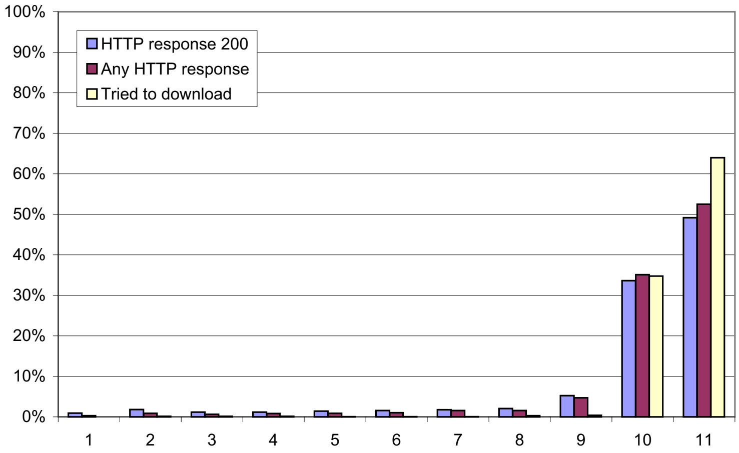

Figure 1. Number of successful, completed, and attempted downloads per URL.

the document. This similarity metric is applicable to any kind of document that consists of an ordered sequence of discrete features. The following description assumes that the features are words, i.e. that a document is an ordered sequence of words. In order to compare two documents, we map each document into a set of k-word subsequences (groups of adjacent words or 'shingles'), wrapping at the end of the document, so that every word in the document starts a shingle.

Two documents are considered to be identical if they map to the same set of shingles[§]; they are considered to be similar if they map to similar sets of shingles. Quantitatively, the similarity of two documents is defined to be the number of distinct shingles appearing in both documents divided by the total number of distinct shingles. This means that two identical documents have similarity 1, while two documents that have no shingle in common have similarity 0.

Note that the value of k parameterizes how sensitive this metric is. Changing the word 'kumquat' to 'persimmon' in an n-word document (assuming that 'kumquat' and 'persimmon' occur nowhere else in the document) results in a similarity of $(n - k)/(n + k)$ between the original and the modified document. This means that one should not choose k to be too large, lest even small changes result in low similarity. On the other hand, neither should one choose k to be too small. As an extreme example, setting $k = 1$ results in a comparison of the lexicon of two documents, making the metric completely insensitive to word ordering.

Our feature vector extraction module substitutes HTML markup by whitespace, and then segments the document into five-word shingles, where each word is an uninterrupted sequence of alphanumeric characters. We chose to elide markup because one of the goals of this study was to examine the

[§] This definition of identity differs from the standard definition. For example, a document A is considered identical to AA, the concatenation of A to itself. Similarly, given two documents A and B, document AB is identical to document BA.

evolution of textual content with an eye towards incremental crawling and indexing by search engines, and because of our hypothesis that changes to URLs and image references constitute the majority of page changes. Moreover, we expect that meaningful changes to links will be accompanied by changes to the corresponding anchor text, which is part of the indexable text.

Next, it computes a 64-bit checksum of each shingle, using Rabin's fingerprinting algorithm [10,11]. We call these fingerprints the 'pre-images'. Next, the module applies 84 different (randomly selected but fixed thereafter) one-to-one functions to each pre-image. For each function, we retain the pre-image which results in the numerically smallest image. This results in a vector of 84 pre-images, which is the desired feature vector.

If the one-to-one functions are chosen randomly[¶], then given the feature vectors of two documents, two corresponding elements of the vectors are identical with probability equal to the similarity of the documents.

The feature vector extraction module logs the feature vector of each document, together with a checksum of its raw content, the start time and the duration of its retrieval, the HTTP status code (or an error code indicating a TCP error or a robot exclusion), the document's length, the number of non-markup words, and the URL. If the document cannot be downloaded successfully or does not result in an HTML document[‖], the module logs the URL and special values for everything else. URLs that were not downloaded because the crawl was terminated before it was complete are treated in a similar fashion.

The document sampling module saves all those successfully downloaded documents whose URLs hash to 0 modulo 1000; in other words, it saves 0.1% of all downloaded documents (assuming we used a uniform hash function). The log contains the URL of each document as well as the entire HTTP response to each request, which includes the HTTP header as well as the document.

3.2. Distilling the data

Crawling left us with 44 very large logs (produced by the 11 crawls on the four crawlers), each spanning multiple files, one per day. The logs totaled about 1200 GB, whereas the sampled documents took up a mere 59 GB.

As they were, these logs were not suitable for analysis yet, because the URLs occurred in non-deterministic order in each log. One way to rectify this would be to perform a merge-sort of the logs based on their URL, in order to bring together the various occurrences of each URL. However, performing such a merge-sort on 1200 GB of data is prohibitively expensive. To overcome this problem, we *bucketized* each of the logs, dividing its contents over 1000 buckets, using the same URL-based hash function we employed in the document sampling module. This step produced 44 000 buckets (1000 buckets per crawler per crawl). While this might appear to double the storage requirement, we could process an individual daily log file, and then move it to near-line storage. As a result of the bucketization, all the occurrences of a given URL appear in corresponding buckets across generations,

[¶] We drew the 84 functions from a smaller space, which does not seem to be a problem in practice. In addition, because we fixed them at the time when we wrote the feature vector extraction module, an adversary who had the code of that module could produce documents that would fool our metric.

[‖] We actually did encounter several documents whose content type changed during the time we observed them.

and each bucket is less than 30 MB in size. This allowed us to then perform a fast in-memory sort of each bucket, using the URL as the key, and replace the original bucket with a sorted one.

At the end of this process, the corresponding buckets of each generation all contained exactly the same set of URLs, in sorted order.

Finally, we merged the buckets across crawls, and distilled them in the process. We did so by iterating on all four machines over the 1000 bucket classes, reading a record at a time from each of the 11 corresponding buckets, and writing a combined record to a distilled bucket. Each combined record contains the following information.

- The URL.
- The start times of the 11 downloads.
- The duration of each download.
- The length of the document at each download.
- The number of non-markup words in the document at each download.
- The HTTP status code (or error code) for each download.
- Six 'supershingles' for each download.
- For each pair of downloads, an 84-bit vector indicating whether the 84 corresponding pre-images matched.
- For each pair of downloads, a match count (see below).

Note that the distilled record does not include the checksums or the pre-images of each document. These values are subsumed by the 'supershingles', the bit-vectors, and the match counts.

Each of the six 'supershingles' represents the concatenation of 14 adjacent pre-images. Due to the independence of the one-to-one functions used to select the pre-images, if two documents have similarity p, each of their supershingles matches with probability p^{14}. For two documents that are 95% similar, each supershingle matches its counterpart with probability 49%. Given that we retain six supershingles, there is a probability of almost 90% that at least two of the six supershingles will agree for documents that are 95% or more similar. Retaining the supershingles will be useful in the future, should we try to discover approximate mirrors [12] and investigate their update frequency.

The match count, when non-negative, indicates how many of the 84 pre-image pairs matched between the two documents. If the document fingerprint matches as well, the match count is set to 85. The match count is negative if either document was not downloaded successfully or contained no words at all (which caused the pre-images to have a default value); its value indicates which condition applied to which document, and whether the documents were identical (i.e. their checksums matched).

3.3. Mining the data

While the original logs were 1200 GB in size, the distilled buckets consume only about 222 GB. Even so, it still takes about 10 h to read the distilled logs. In order to conduct the statistical experiments we report on below (as well as a number of others which proved less informative), we needed to find a way to conduct each experiment considerably faster. We achieve this by running several experiments simultaneously, thereby amortizing the cost of reading the logs over multiple experiments.

Using this approach, we were no longer limited by our computer's ability to read the logs, but rather by our ability to decide *a priori* which experiments would be interesting and relevant. Given the trial-and-error character of any data mining activity, we still had to do several passes over the data to construct all of the experiments described below.

We built an analyzer harness that reads through the distilled buckets one record at a time, expands each record into an easy-to-use format, and presents it to each of a set of analyzer modules. Once all buckets have been read, the harness invokes a 'finish' method on each of the analyzers, causing them to write out any aggregate statistics they may have gathered. The analyzers as well as the harness are written in Java; the harness uses Java's dynamic class loading capabilities to load the analyzers at runtime.

Examples of the analyzers we wrote include the following.

- *StatusCodeAnalyzer*, which produces a histogram of HTTP status codes and TCP error conditions.
- *FetchDurationAnalyzer*, which produces a logarithmic histogram of document download durations.
- *DocLengthAnalyzer*, which produces a logarithmic histogram of document lengths.
- *NumVersionsAnalyzer*, which produces a histogram of how many versions of each URL we managed to download successfully.
- *TopLevelDomainAnalyzer*, which produces a histogram of top-level domains, and optionally counts the number of hosts in that domain.
- *ChangeAnalyzer*, which produces a histogram of the number of unchanged pre-images between two successive successful downloads of a URL.

Analyzers can be nested into higher-level analyzers. For example, it is possible to put a *ChangeAnalyzer* into a *TopLevelDomainAnalyzer*, producing a list of document change histograms, one for each top-level domain. This is again implemented using Java's dynamic class loading machinery. In order to keep the size of the output manageable, we also provide ways to limit the number of top-level domains considered (putting all unspecified domains into a catch-all category), to group numbers of unchanged pre-images into clusters, and the like.

For some of these higher-level analyzers, we need to aggregate multiple values (e.g. the sizes of the 11 versions of a Web page) into a single value, in order to decide what lower-level analyzer to invoke. We currently provide three different ways to aggregate values, namely minimum, maximum, and average. We plan to add support for mode, median, and geometric average.

Recall that during the data collection phase, we saved the full text of 0.1% of all successfully downloaded pages. We selected these pages based on a hash of the URL, using the same hash function as the bucketizer. In particular, we saved the full text of all pages that went into bucket 0. This enables us to use the analyzer framework to detect interesting patterns, to use a special analyzer (using higher-level analyzers around a '*DumpURLAnalyzer*') to get a listing of all URLs in bucket 0 that fit this pattern, and then to examine the full text of some of these documents.

We built some infrastructure to make this process easier. In particular, we prepend each file of sampled documents with an index of URLs and document offsets, which allows us to retrieve any page in constant time. Second, we implemented a Web service that accepts a URL, a version number, and whether to return the HTTP header or the document, and returns the requested item for display in the browser. Third, we implemented another Web service that accepts a list of analyses to run, executes them on a subset of the distilled buckets, and returns the results as a Web page. We hope to make these services, as well as some of the distilled buckets, available to the research community.

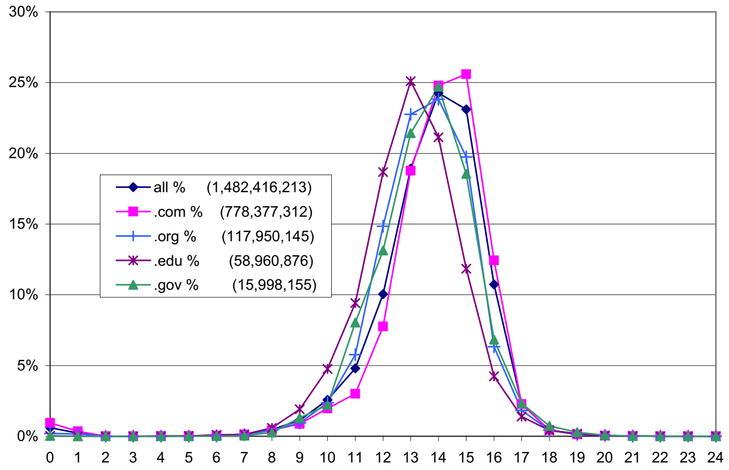

Figure 2. Distribution of document lengths overall and for selected top-level domains.

4. RESULTS

The results presented in this section are derived from analyzing the 151 million distilled records in our collection, using the analyzer harness and many of the analyzers described above.

Figure 2 shows a histogram of document length, for all of the 1 482 416 213 documents that were successfully downloaded (i.e. that had an HTTP status code of 200), as well as broken out by a few selected top-level domains. The x-axis denotes the document size; a value of n means that the size of the document was below 2^n bytes, but not below 2^{n-1} bytes (a value of 0 indicates that the document had length 0).

The distribution we observed centers at 14 with standard deviation 1; 66.3% of all observed HTML pages are between 4 and 32 KB in length. Looking at selected top-level domains, pages in `.com`, which represent 52.5% of all observed pages, largely reflect the overall distribution, but are biased slightly higher. Pages in `.org` and `.gov`, which account for 8.0% and 1.1% of all observed pages, respectively, are similar to the overall distribution, but are biased slightly lower. Pages in `.edu` tend to be smaller, with 64.9% of the pages being between 2 and 16 KB.

Figure 3 is similar to Figure 2, but shows a histogram of the number of words per document, instead of the number of bytes. Note that the distribution of the `.edu` domain is closer to the overall distribution when it comes to words, suggesting that pages in `.edu` either have shorter words or, more likely, less HTML markup.

Figures 4–6 attempt to capture different aspects of the permanence of Web pages. Figure 4 shows for each crawl generation the percentage of page retrievals resulting in different categories of status codes. The *200* category (corresponding to the HTTP status code '200 OK') shows pages that were successfully downloaded. Note that the y-axis starts at 85%; in all generations, over 85% of the retrievals were successful. The *3xx* category contains all those pages that return an HTTP status code

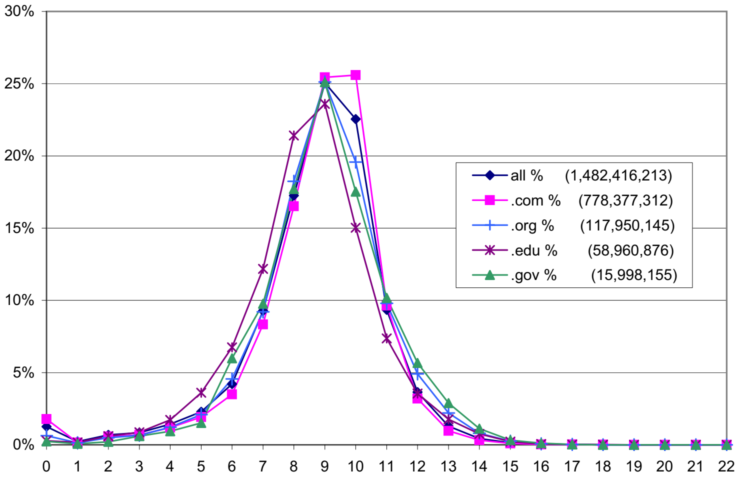

Figure 3. Distribution of words per documents overall and for selected top-level domains.

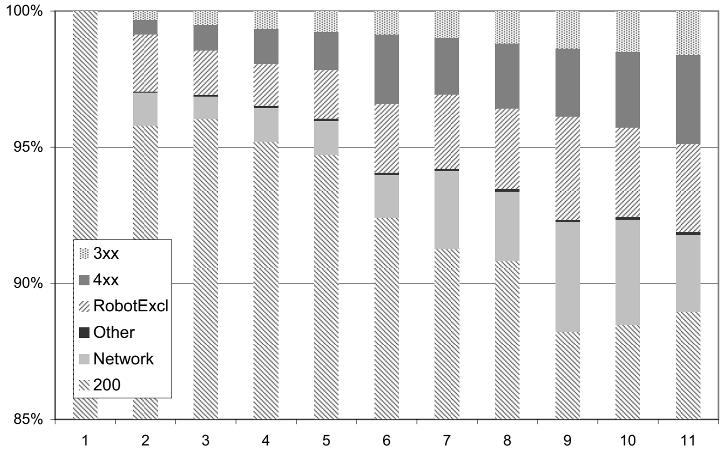

Figure 4. Distribution of HTTP status codes over crawl generations.

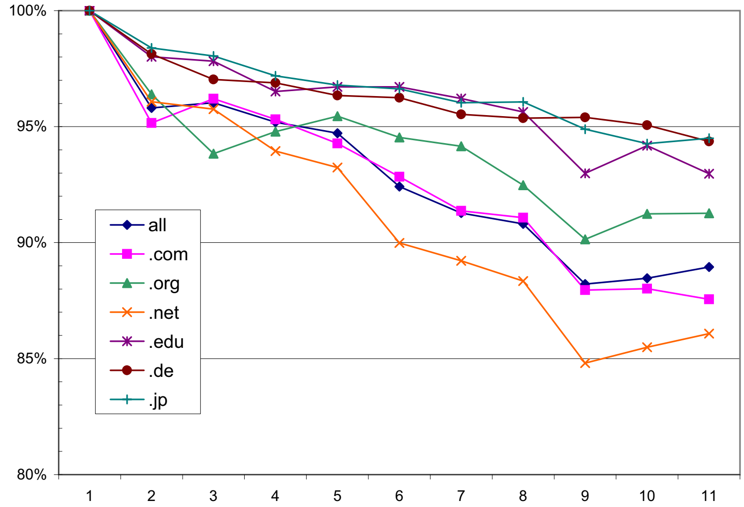

Figure 5. Distribution of successful downloads over crawl generations, broken down by selected top-level domains.

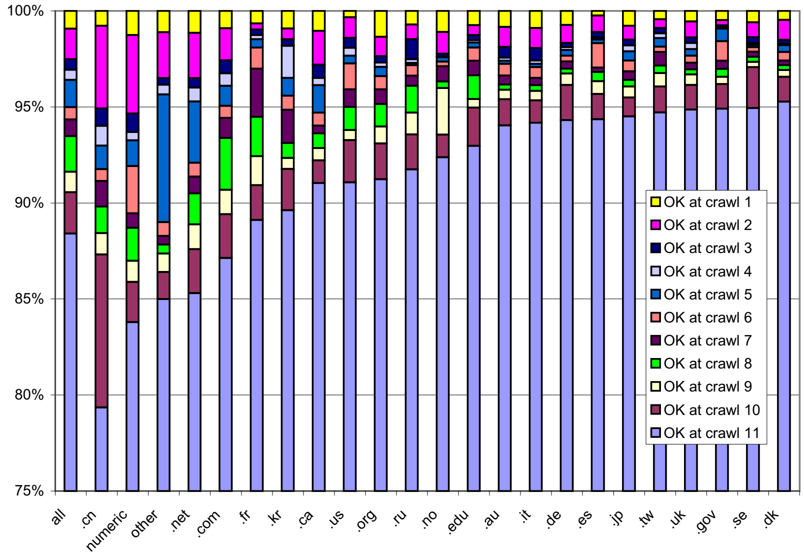

Figure 6. Breakdown showing in which crawl a Web page was last successfully downloaded, broken down by selected top-level domains.

indicating that the page has moved. Since all URLs in our set produced a status code of 200 during crawl 1, the page has moved since. The *4xx* category contains all client errors. The most common one is 404 ('Not Found'), indicating that the page has disappeared without leaving a forwarding address, distantly followed by 403 ('Forbidden'). The *other* category contains all pages for which the Web server returned a status code not listed above. The various *5xx* return codes dominated; we also found many Web servers returning status codes not listed in the HTTP RFC. The *network* category contains all retrieval attempts that failed due to a network-related error, such as DNS lookup failures, refused connections, TCP timeouts, and the like (note that Mercator makes five attempts to resolve a domain name, and three attempts at retrievals that fail due to TCP errors). The *RobotExcl* category contains all those pages that were blocked by the Web server's `robots.txt` file, and that we therefore refrained from even attempting to download. Again, since these pages were not excluded during crawl 1, the exclusion was imposed later. This appears to be a form of the Heisenberg effect, where the existence of an observer (a pesky Web crawler that trots by every week) changes the behavior of the observed.

As one might expect, Figure 4 demonstrates that the lifetime of a URL is not unlimited, although over 85% of document download attempts were successful every week. As the crawl generations increase, more and more URLs move, become unreachable, or are blocked from us. While one would expect a geometric progression of these trends, we did not observe the Web long enough to distinguish the trend from a linear progression. The growth of these three categories comes at the expense of the *200* category; fluctuations in the share of the *network* category appear better correlated with network conditions at the crawler side.

Figure 5 shows only the successful downloads as a percentage of all download attempts (including non-attempts due to robot exclusion rules), broken down by a few selected top-level domains. Note that the y-axis starts at 80%, reflecting the fact that all domains in all generations had at least that level of success. In general, pages in `.jp`, `.de`, and `.edu` were consistently more available than pages in `.net` and `.com`. The decline in the curves bears out the limited lifetime of Web pages discussed above.

In Figure 6 we tried another approach for viewing the lifetime of URLs from different domains. Each bar represents a top-level domain (the leftmost bar represents the entire data set). We grouped URLs by the crawl generation of their last successful retrieval, the intuition being that a URL which could not be downloaded after some point is likely to have expired. This approach partitions URLs into 11 sets. Each shaded region of each bar represents the relative size of one such set. The region at the top of a bar corresponds to URLs that were consistently unreachable after crawl 1 (the crawl that defined the set of URLs), while the region at the bottom of a bar corresponds to URLs that were successfully downloaded during the final crawl. Note that the y-axis, which shows the percentage breakdown of the total, starts at 75%, due to the fact that for all domains considered here, 75% of all URLs were still reachable during the final crawl. Looking at the 'all domains' bar, it can be seen that 88% of all URLs were still available during the final crawl.

For most domains, the 'OK at crawl 10' regions are larger than regions for preceding crawls. This makes intuitive sense: it represents documents that could not be retrieved during the final crawl, but might well come back to life in the future. Other than that, we see no discernible patterns between the lengths of regions within a bar.

Looking across domains, we observe that Web pages in China expire sooner than average, as do pages in `.com` and `.net`.

The remaining figures display information about the amount of change in a document between two successive successful downloads. Figure 7 shows a fine-grained illustration of amount of change,

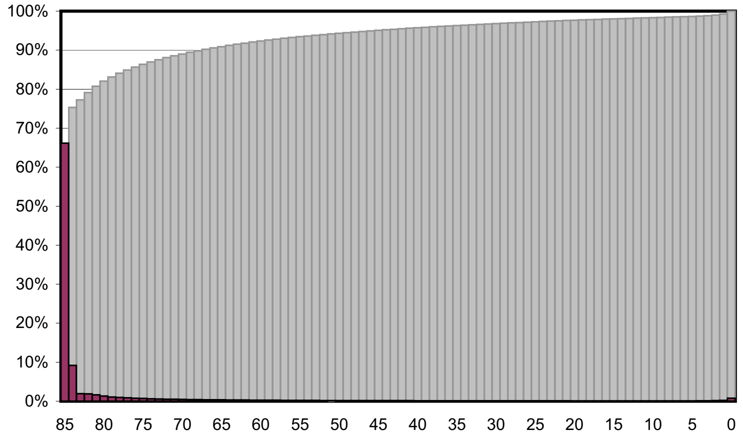

Figure 7. Distribution of change.

independent of other factors. We partition the set of all pairs of successive successfully retrieved pages into 85 subsets, based on how many pre-images the two documents in each pair share. Subset 0 contains the pairs with no common pre-images, subset 84 the ones with all common pre-images, and subset 85 the ones that also agree in their document checksums (strongly suggesting that the documents are identical as byte sequences).

The x-axis shows the 85 subsets, and the y-axis shows percentages. The lower curve (visible only at the extreme left and right) depicts what percentage of all documents fall into a given subset. The curve above shows the cumulative percentage distribution. Each point (x, y) on the curve indicates that y percent of all pairs had at least x pre-images in common.

As can be seen (if your eyes are sharp enough), 65.2% of all page pairs do not differ at all. Another 9.2% differ only in their checksum, but have all common pre-images (suggesting that only the HTML markup, which was removed before computing pre-images, has changed).

Figure 8 magnifies the lower curve from Figure 7, making it easy to see that all other change buckets contain less than 2% of all page pairs, and that all buckets with fewer than 79 common pre-images (representing document pairs that are less than 94% similar) contain less than 1%. For the most part, the curve is monotonically decreasing, but this pattern is broken as we get to the buckets containing documents that underwent extreme change. Bucket 0, which represents complete dissimilarity, accounts for 0.8%, over ten times the level of bucket 7, the smallest bucket. We will revisit this phenomenon later on.

Although we cannot tell from the pre-images what changed in the markup, we were curious what types of markup changes would accompany 84 identical pre-images. To investigate, we looked at the 9785 URLs for which we had saved the full text of each download and which consistently had markup-only changes. For each pair of consecutive downloads of these URLs, we extracted the markup of each download, one tag per line in a new file, and ran the Unix command ‘diff’ over the two files. We then

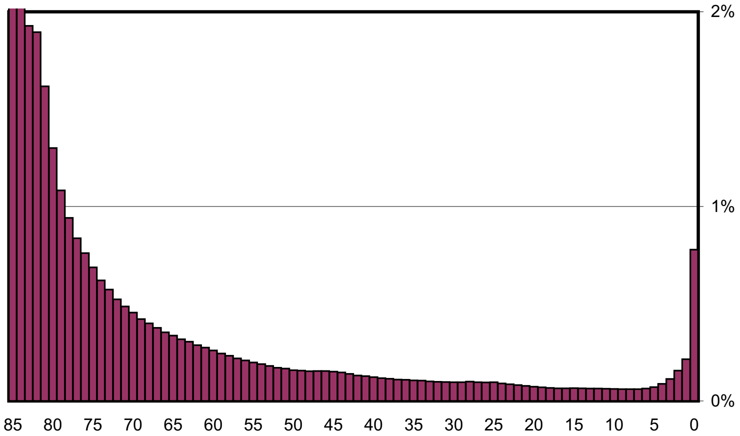

Figure 8. Distribution of change, scaled to show the low-percentage categories.

examined the output of 'diff' and categorized each change as an addition or deletion of a tag or an attribute, as a change to an attribute's value, or as a whitespace change. We further parsed the changed values to determine whether the change occurred before the first instance of a '?' or ';' in the value: changes that occur afterwards (almost exclusively in URL values) are in the query portion of a URL.

We observed a total of 1 468 671 changes in the markup of these pages. Figure 9 shows the types of markup changes, aggregated over all changes. Over half of the changes (62%) are changes to an attribute; 48% are changes to a URL attribute that occur after a '?' or ';'. We ascertained by manual inspection that most of these changes are session identifiers embedded in the query portion of URLs. Observing link evolution of this type may help a crawler in spotting session identifiers. If a crawler were able to recognize embedded session identifiers and remove them, then it could avoid recrawling the same content multiple times. In our experience, such unwarranted recrawling accounts for a non-negligible fraction of total crawl activity.

A smaller fraction of markup changes are advertisements, chosen again by embedding some identifier in the query portion of a URL. In fact, such URL changes are so common that at least one attribute change in the query portion of a URL appears in 66% of all document pairs. Adding and deleting attributes accounts for another 33% of markup changes. Tag changes are much less common, comprising only 6% overall. Whitespace changes are negligible.

We then decided to see whether the statistics in Figure 9 were highly influenced by either URLs with high numbers of changes or by hosts with many URLs. In Figure 10 we show the same statistics, now normalized so that only one change of each type is counted per host. The percentage of changes attributed to URL queries has now shrunk from 48% to 4%, indicating that these changes tend to appear many times on the same page when they occur at all. This discovery makes sense: when session identifiers are embedded in links to URLs, they tend to appear in all relative links on the page. The relative percentages of the other types of changes are the same in Figures 9 and 10.

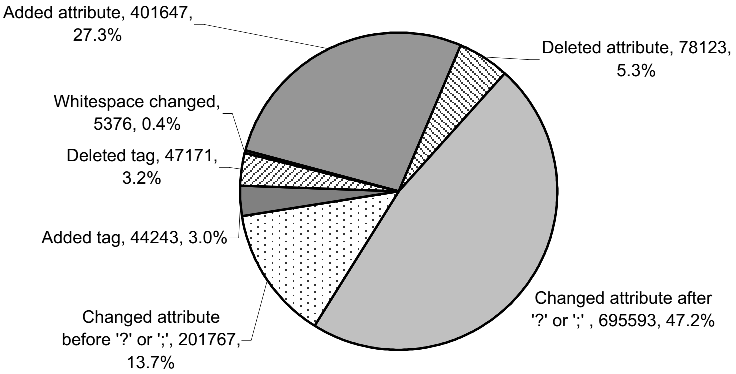

Figure 9. Type of markup change.

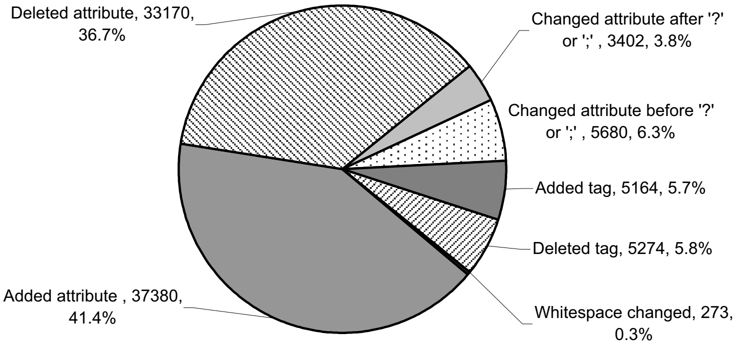

Figure 10. Type of markup change normalized by hosts.

In Figure 11 we show the changes that were additions or deletions of tags, broken down by tag. One surprising result is that 23% of the changes were within tags that started with the comment character '!', indicating that the change was exclusively to a comment. Closer inspection of the comments revealed that the majority were actually changes to Javascript, which is embedded in HTML as comments. Another result is that nearly half of the tag changes are to 'A' tags, and 90% of those are 'HREF' attributes, i.e. URLs. For the 'IMG' tags, only 52% of the changes were to the 'SRC' attribute, i.e. URLs.

We next tried to determine if there are any attributes of a document that help to predict its rate and degree of change. In Figure 12, we examine the relationship between top-level domain and change. The figure shows a bar for all page pairs and several more for selected domains. Each bar is divided into six regions, corresponding to the following six change clusters: *complete change* (0 common

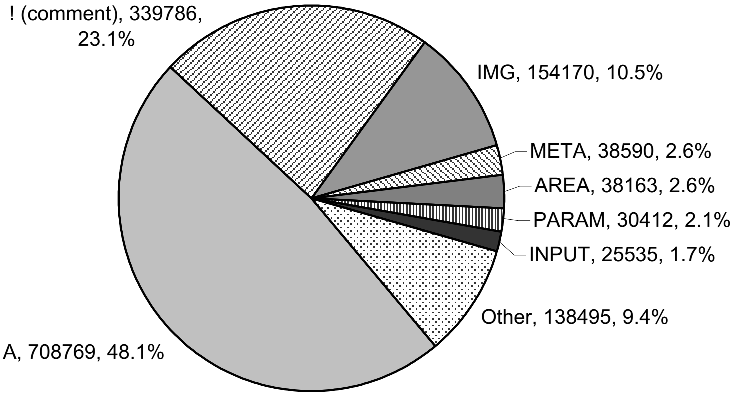

Figure 11. Breakdown of tags that changed.

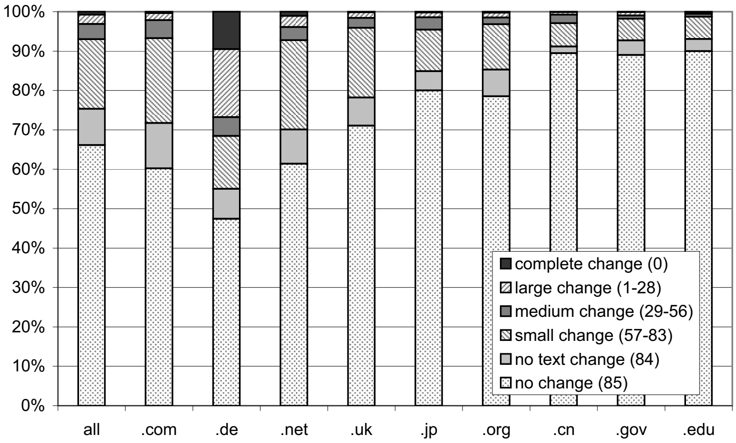

Figure 12. Clustered rates of change, broken down by selected top-level domains.

pre-images), *large change* (1–28 common pre-images), *medium change* (29–56 common pre-images), *small change* (57–83 common pre-images), *no text change* (84 common pre-images), and *no change* (subset 85: 84 common pre-images and a common checksum).

The top region depicts the *complete change* cluster, the one at the bottom the *no change* cluster. The y-axis shows the percentage.

We observe significant differences between top-level domains, confirming earlier observations by Cho and Garcia-Molina [3]. Pages in the `.com` domain change more frequently than those in the `.gov` and `.edu` domains.

We were surprised to see that pages in `.de`, the German domain, exhibit a significantly higher rate and degree of change than those in any other domain. A total of 27% of the pages we sampled from `.de` underwent a large or complete change every week, compared with 3% for the Web as a whole. Even taking the fabled German industriousness into account, these numbers were hard to explain.

In order to shed light on the issue, we turned to our sampled documents, selecting documents from Germany with a high change rate. Careful examination of the first few pages revealed more than we cared to see: of the first half dozen pages we examined, all but one contained disjoint, but perfectly grammatical phrases of an adult nature together with a redirection to an adult Web site. It soon became clear that the phrases were automatically generated on the fly, for the purpose of 'stuffing' search engines such as Google with topical keywords surrounded by sensible-looking context, in order to draw visitors to the adult Web site. Upon further investigation, we discovered that our data set contained 1.03 million URLs drawn from 116 654 hosts (4745 of them from outside the `.de` domain), which all resolved to a single IP (Internet provider) address. This machine provided over 15% of the `.de` URLs in our data set!

We speculate that the purpose of using that many distinct host names as a front to a single server is to circumvent the politeness policies that limit the number of pages a Web crawler will attempt to download from any given host in a given time interval, and also to trick link-based ranking algorithms such as PageRank into believing that links to other pages on apparently different hosts are non-nepotistic, thereby inflating the ranking of the pages in the clique.

After this discovery, we set out to explore if there were other such servers in our data set. We resolved the symbolic host names of all the URLs in our data set, and singled out each IP address with more than a 1000 symbolic host names mapping to it. There were 213 such IP addresses, 78 of which proved to be of a similar nature as the site that triggered our investigation. We excluded all URLs on the 443 038 hosts that resolved to one of the 78 identified IP addresses, and reran the analysis that produced Figure 12. This eliminated about 60% of the excessive large and complete changes in `.de`. The adjusted distribution is shown in Figure 13. Continued investigation of the excessive change found that automatically generated adult content accounts for much of the remainder.

In Figure 14, we look at the same data, but omit all pairs of documents with no change. Other than Germany (and to a lesser extent China and Japan), there is remarkably little difference between the various top-level domains. Our conclusions are three-fold: first, adult content continues to skew our results. Second, our shingling technique might not be well adapted to writing systems like Chinese or Kanji that do not employ inter-word spacing, which in turn might cause documents to have a small number of shingles, which means that any change is considered significant. Finally, even in top-level domains where most documents rarely change, when they *do* change, the extent of change is quite consistent with other top-level domains.

Figure 15 is similar to Figure 8, but excludes the same URLs that were excluded in Figures 13 and 14. Note that most of the non-monotonicity at the right end of the distribution has disappeared, except for bucket 0, which nonetheless has been cut in half.

We next consider whether the length of pages impacts their rate of change. In Figure 16, we use the same x-axis semantics as in Figure 2, and the same y-axis semantics and bar graph encodings as in Figure 12. The most striking feature of this figure is that document size is strongly related to amount

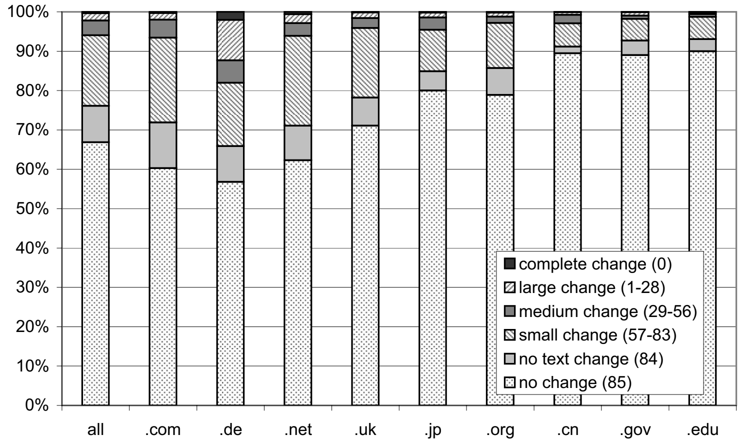

Figure 13. Clustered rates of change, broken down by selected top-level domains, after excluding automatically generated keyword-spam documents.

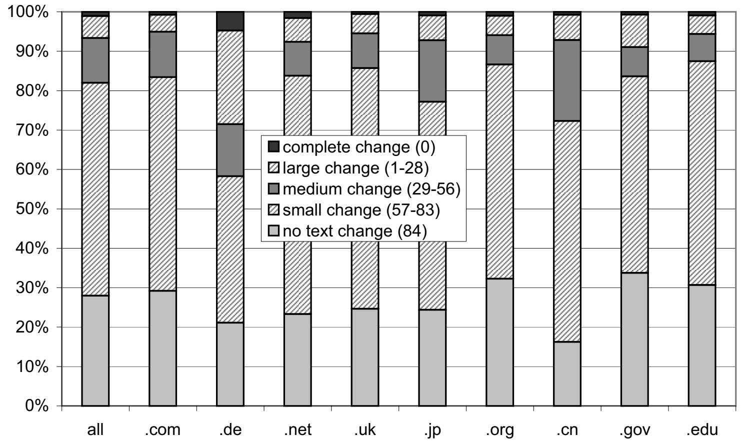

Figure 14. Clustered rates of change, broken down by selected top-level domains, and omitting the *no change* cluster, after excluding automatically generated keyword-spam documents.

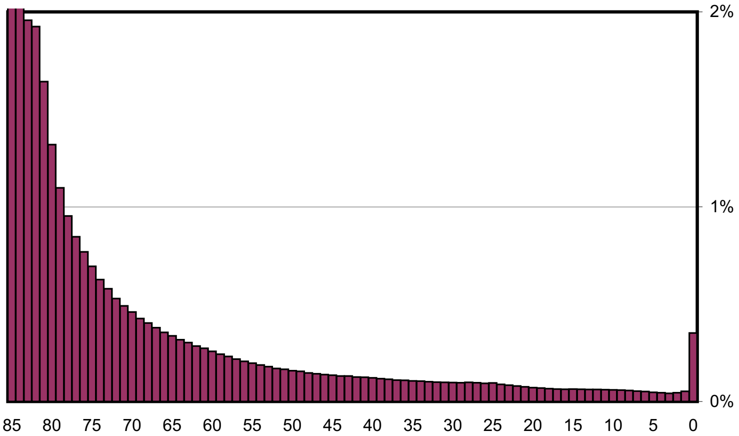

Figure 15. Distribution of change, scaled to show the low-percentage categories, after excluding automatically generated keyword-spam documents.

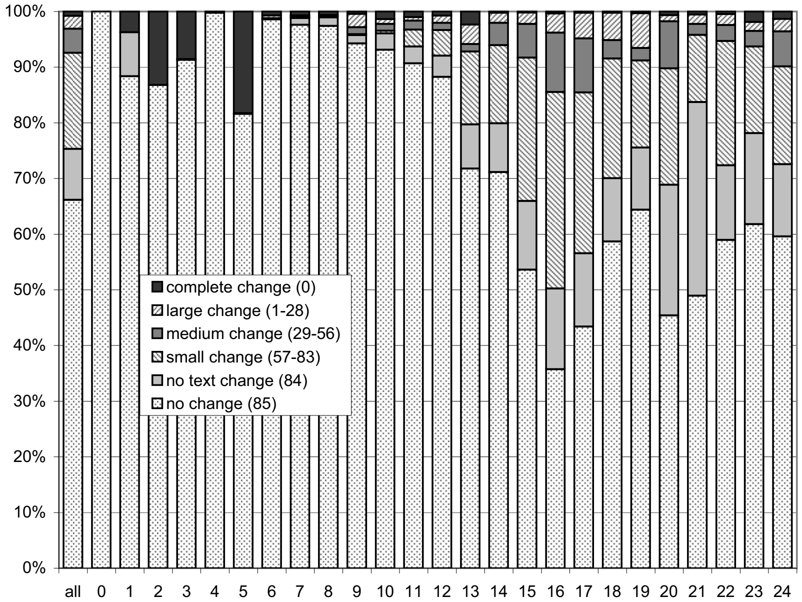

Figure 16. Clustered rates of change, broken down by document size.

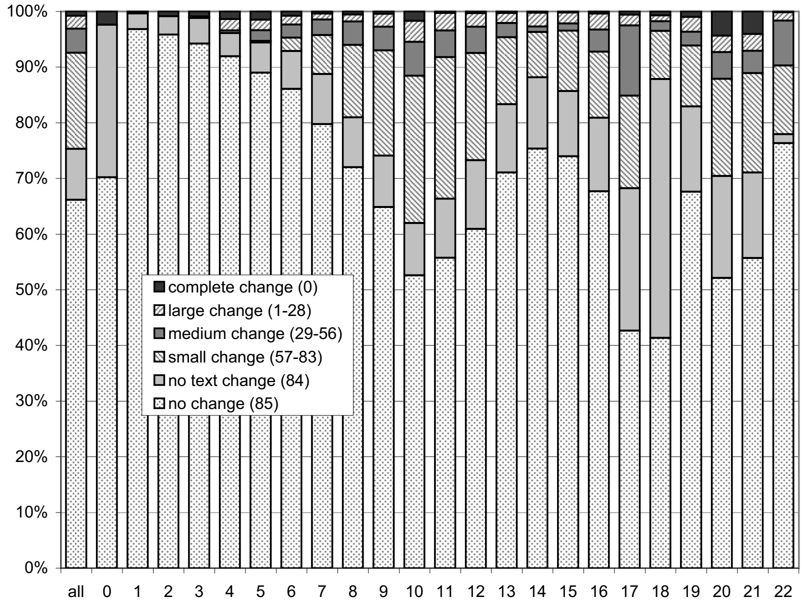

Figure 17. Clustered rates of change, broken down by the number of words per document.

and rate of change, and counterintuitively so! One might think that small documents are more likely to change, and if they do, change more severely (since any change is a large change). However, we found that large documents (32 KB and above) change much more frequently than smaller ones (4 KB and below). Cursory examination of the URLs and some sampled pages reveals no single explanation of this phenomenon; however, many of the pages in this category are log files and discussion lists.

Figure 17 is similar in spirit, but examines the relationship between the number of words and the rate of change. In a way, this metric is more straightforward, since the sensitivity of our shingling techniques depends on the number of words in a document. For documents with only a few words, our metric gives a relatively coarse, 'all-or-nothing' similarity metric. Nonetheless, this figure echoes the observation of Figure 16, i.e. that large documents are more likely to change than smaller ones.

In Figure 18, we examine the same information, excluding the documents with no change. We observe that pages with different numbers of words exhibit similar change behavior, except that pages with just a few words cannot show an intermediate amount of change, due to our sampling technique. We also spotted some apparent anomalies in the changes to documents with word counts greater than 2^{16}. However, the combined total of these documents is fewer than 0.01% of the total, and the anomalies turn out to be statistically insignificant.

We further investigated whether there are any confounding relationships between document size and top-level domain. Figure 19 uses the same representation as Figure 16, but each chart considers only

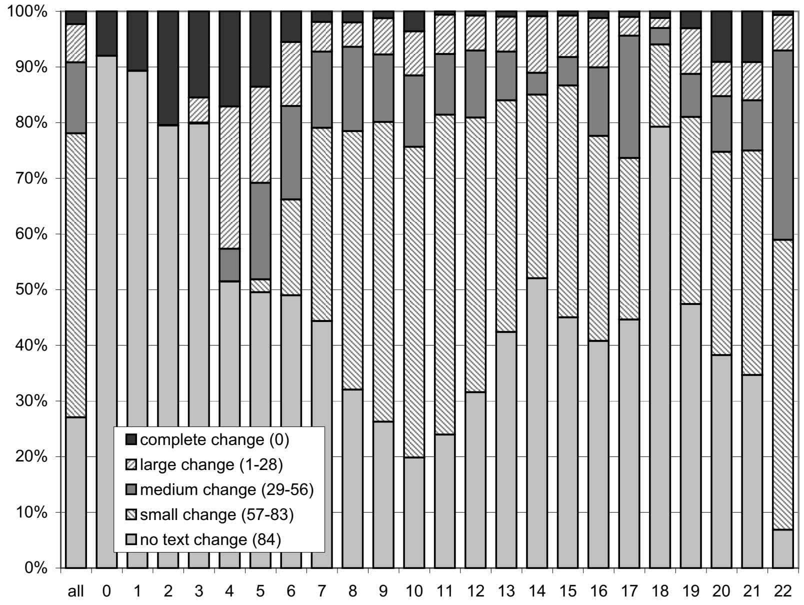

Figure 18. Clustered rates of change, broken down by the number of words per document, and omitting the *no change* cluster.

those URLs from a specific top-level domain. The distributions for the `.com` and `.net` domains exhibit a much stronger threshold effect for large documents than do the `.gov` and `.edu` domains.

We speculate that this might be due to the fact that commercial Web sites have an interest in maintaining the appearance of freshness, while a large fraction of the content on governmental and educational Web sites serves more of an archival purpose. For example, many university Web servers will host scholarly papers and many governmental Web sites will host legal content. Academic papers and legal texts generally remain unchanged once created. By contrast, commercial Web sites often employ a dedicated editorial staff that continually revises Web content; perhaps even more importantly, many commercial Web sites present dynamically created content, which tends to have a higher change rate, since it is created automatically, and thus can be changed much more cheaply.

It is interesting to note that the correlation between change rate and document size for pages in the `.org` domain is halfway between that for `.com` and `.edu`. We attribute this to the fact that the `.org` domain was originally created to host non-commercial organizations. While this policy is not enforced, it appears that even today a large fraction of the sites in `.org` are non-commercial in nature, and therefore do not have dedicated staff to continually refresh the content.

In non-US domains, such as `.de` and `.jp`, commercial, governmental, and educational Web sites all exist under the same national top-level domain. We therefore expect the correlation between change rate and document size for these top-level domains to be intermediate to that of `.com` on one hand and `.edu` or `.gov` on the other. This is largely borne out by the figures, except for the fact that German

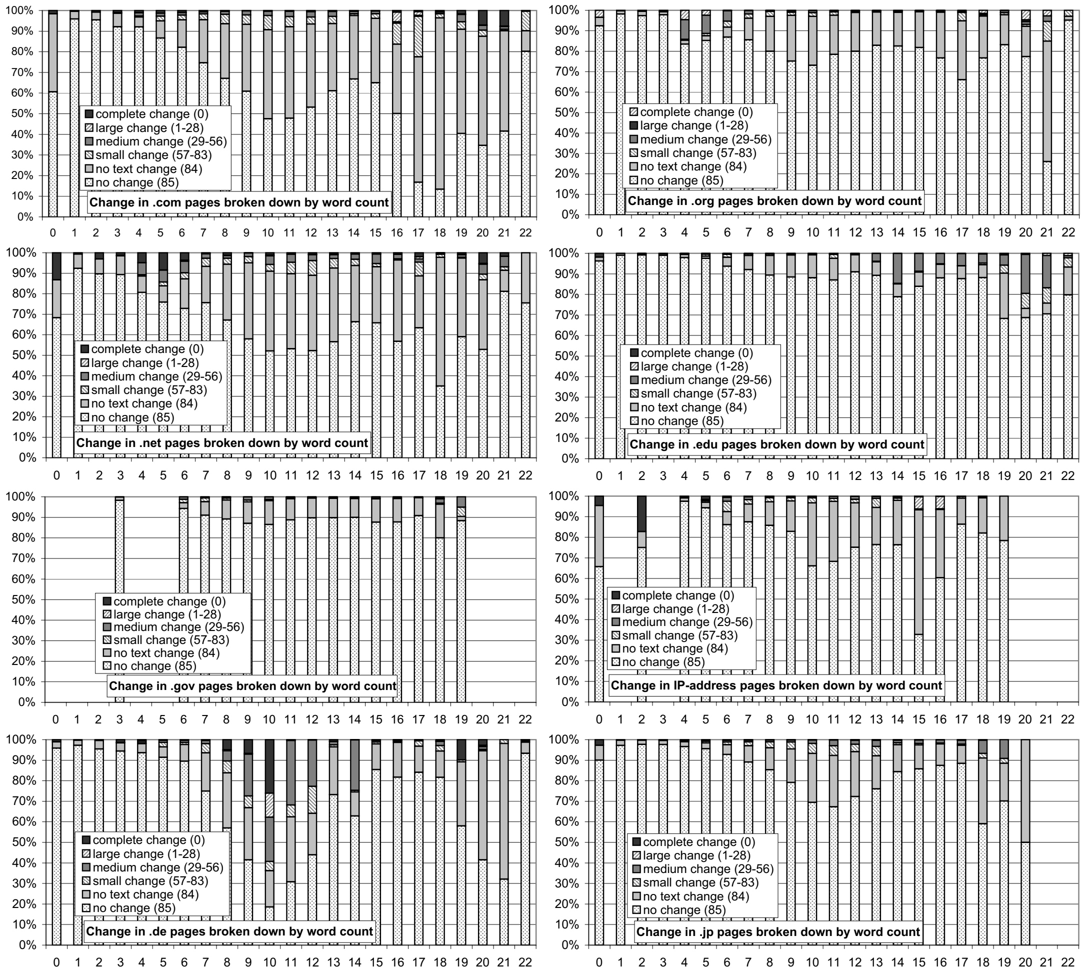

Figure 19. Clustered rates of change, broken down by top-level domain and number of words per document.

documents containing between 129 and 4096 words show a much higher than expected change rate. We attribute this to the prevalence of machine-generated adult content described above in the discussion of Figure 12.

Our overall conclusion is that page size and top-level domain have independent influence on the rate of change of documents; the prevailing rate of change for a top-level domain acts predominantly as a scaling factor for the change rate curve dependent on document size.

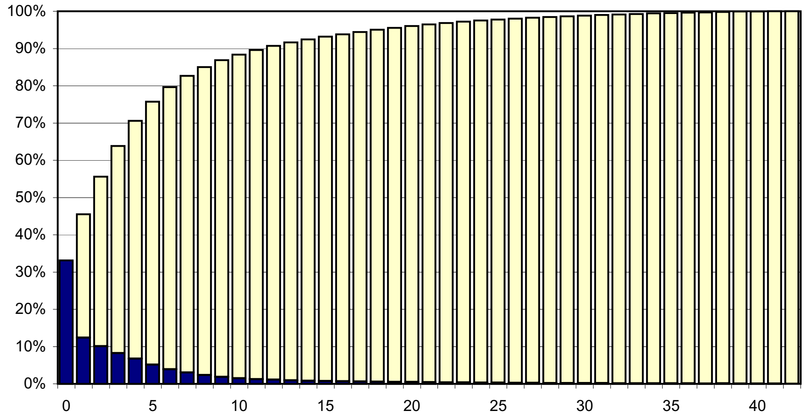

Figure 20. Distribution of the standard deviations of the rate of change in a given document over its lifetime.

The dark set of bars in Figure 20 shows a histogram of the standard deviations of the number of changed features in a given document over the entire sampling period for that document, ignoring any samples that did not change from the previous successful samples.

The horizontal axis shows the standard deviations, which could reach 85, since two versions of a document can differ in up to 85 aspects (84 shingles and 1 checksum). However, the largest standard deviation we actually observed was below 43. Documents are grouped into buckets; bucket n contains all those documents whose standard deviation of changed features is in the range $[n, n+1)$. The vertical axis shows the percentage of documents in a particular bucket. As can be seen, for about a third of the documents, the standard deviation is less than 1, meaning that for these documents, the rate of change was very stable.

The light set of bars shows the cumulative distribution of standard deviations. As can be seen, 80% of all documents have a standard deviation of less than 7, and 90% of all documents have a standard deviation of less than 12. More generally speaking, for most documents, the standard deviation of changed features is small, meaning that the amount of week-to-week change is predictable and fairly constant.

We excluded samples that did not change from the previous week because they overwhelm the remaining data. Were we to include samples that did not change from the previous week, bucket 0 would include 56% of all documents, reflecting that most documents are quite stable.

Plate 1 examines the correlation of successive changes to a document. The figure shows a three-dimensional histogram. The x-axis denotes the number of pre-images in a document unchanged from week $n-1$ to n, the y-axis the number of pre-images unchanged from week n to $n+1$, and the z-axis shows the logarithm (base 2) of the number of such documents. A data point (x, y, z) indicates that there are 2^z document/week pairs (d, n) for which the versions of document d had x pre-images in common between weeks $n-1$ and n, and y pre-images in common between weeks n and $n+1$.

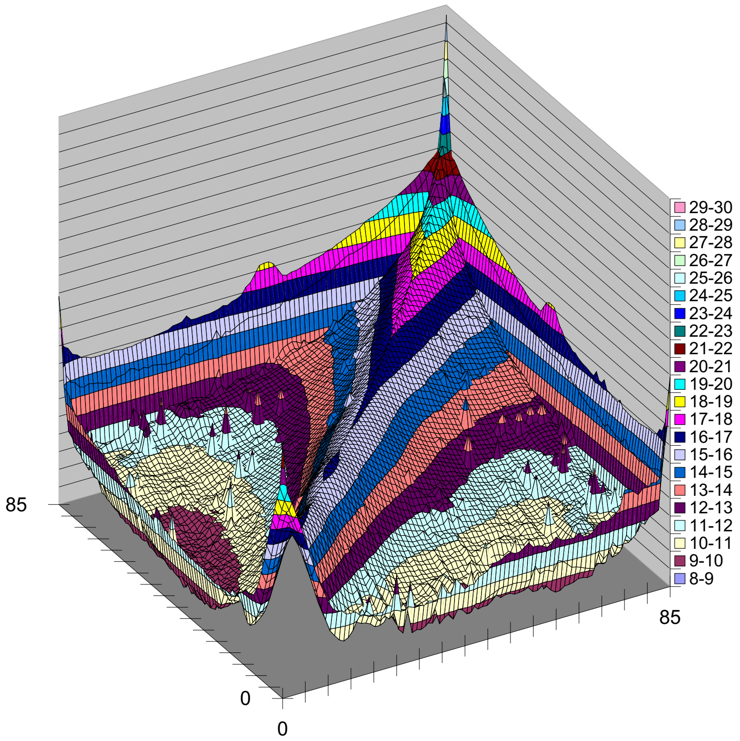

Plate 1. Logarithmic histogram of intra-document changes over three successive weeks, showing the absolute number of changes.

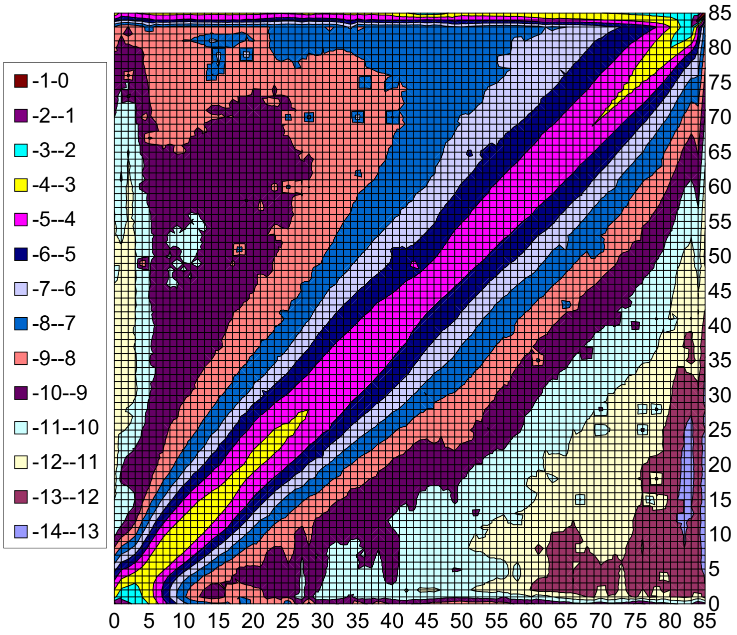

Plate 2. Logarithmic histogram of intra-document changes over three successive weeks, normalized to show the conditional probabilities of changes.

The spire surrounding the (x, y) coordinate $(85, 85)$ represents the vast majority of Web pages that do not change much over a three-week interval. The tip of the spire is 10 000 times higher than any other feature in the plot, except for the smaller spire at the other end of the diagonal, which represents documents which differ completely in every sample. Much of this second peak can be attributed to machine-generated adult content, as described earlier.

The second-most prominent feature is the pronounced ridge along the main diagonal of the xy-plane. The crest of the ridge represents a 1000-fold higher number of instances per grid point than the floor of the valley. This ridge suggests that changes are highly correlated; past changes to a document are an excellent predictor of future changes.

The plumes at the far walls of the plot demonstrate that a sizable fraction of documents do not change in a given week, even if they changed in the previous or following week. In other words, there is a sizable portion of documents for which the update interval exceeds one week.

Plate 2 modifies the previous figure in two ways: The view is down the z-axis, transforming the three-dimensional plot into a two-dimensional contour map, where color/shading indicate the elevation of the terrain. In addition, rather than displaying absolute numbers of samples, we consider each column as a probability distribution (meaning that every data point is divided by the sum of the data points in its column). Since these values range from 0 to 1, their logarithms are negative. This normalization eliminates the spires that were so prominent in the previous figure. The diagonal ridge, however, remains, indicating once again that past change is a strong predictor of future change. Likewise, the plume along the top remains clearly visible.

5. CONCLUSIONS

This paper describes a large-scale experiment aimed at measuring the rate and degree of Web page changes over a significant period of time. We crawled 151 million pages once a week for 11 weeks, saving salient information about each downloaded document, including a feature vector of the text without markup, plus the full text of 0.1% of all downloaded pages. Subsequently, we distilled the retained data to make it more amenable to statistical analysis, and we performed a number of data mining operations on the distilled data.

We found that when Web pages change, usually they either changed only in their markup or in trivial ways. Moreover, we found that there is a strong relationship between the top-level domain and the frequency of change of a document, whereas the relationship between top-level domain and degree of change (when there is a change) is much weaker.

To our great surprise, we found that document size is another strong predictor of both frequency and degree of change. Moreover, one might expect that any change to a small document would be a significant one, by virtue of small documents having fewer words, so that any word change affects a significant fraction of the shingles. Contrary to that intuition, we found that large documents change more often and more extensively than smaller ones.

We investigated whether the two factors—top-level domain and document size—were confounding, and discovered that for the most part, the relationship of document size and rate and degree of change are more pronounced for the `.com` and `.net` domains than for, say, the `.edu` and `.gov` domains, suggesting that they are not confounded.

We also found that past changes to a page are a good predictor of future changes. This result has practical implications for incremental Web crawlers that seek to maximize the freshness of a Web page collection or index. However, the 'German anomaly' that manifested itself in Figure 12 demonstrates that not every fast-changing page is worthy of frequent crawling. Conversely, there are pages that change only to a small degree, but where the changing portion is the most salient component of the page. Therefore, we argue that observed change rate should be just one of several factors that determine the frequency at which a URL is crawled. Other factors to consider include the 'quality' of a page (e.g. as measured by the PageRank metric [13]).

We have done some limited experiments with the sampled full text documents to investigate some of our more perplexing results. These experiments helped us in uncovering a source of pollution in our data set, namely machine-generated pages constructed for the purpose of spamming search engines. We hope that future work using the sampled full text documents will provide us with additional insights.

REFERENCES

1. Google Information for Webmasters. http://www.google.com/webmasters/2.html [16 October 2003].
2. Fetterly D, Manasse M, Najork M, Wiener J. A large-scale study of the evolution of Web pages. *Proceedings of the 12th International World Wide Web Conference*, May 2003. ACM Press, 2003; 669–678.
3. Cho J, Garcia-Molina H. The evolution of the Web and implications for an incremental crawler. *Proceedings of the 26th International Conference on Very Large Databases*, September 2000. Morgan Kaufmann, 2000; 200–209.
4. Sun Q, Simon D, Wang Y, Russell W, Padmanabhan V, Qiu L. Statistical identification of encrypted Web browsing traffic. *Proceedings IEEE Symposium on Security and Privacy*, May 2002. IEEE Computer Society, 2002; 19–30.
5. Douglis F, Feldmann A, Krishnamurthy B, Mogul J. Rate of change and other metrics: A live study of the world wide Web. *USENIX Symposium on Internetworking Technologies and Systems*, December 1997. USENIX Association, 1997; 147–158.
6. Brewington B, Cybenko G. How dynamic is the Web? *Proceedings of the 9th International World Wide Web Conference*, May 2000. Elsevier, 2000; 257–276.
7. Broder A, Glassman S, Manasse M, Zweig G. Syntactic clustering of the Web. *Proceedings of the 6th International World Wide Web Conference*, April 1997; 391–404.
8. Najork M, Wiener J. Breadth-first search crawling yields high-quality pages. *Proceedings of the 10th International World Wide Web Conference*, May 2001. ACM Press, 2001; 114–118.
9. Najork M, Heydon A. High-performance Web crawling. *SRC Research Report 173*, Compaq Systems Research Center, Palo Alto, CA, September 2001.
10. Broder A. Some applications of Rabin's fingerprinting method. *Sequences II: Methods in Communications, Security, and Computer Science*, Capocelli R, De Santis A, Vaccaro U (eds.). Springer-Verlag: Berlin, 1993; 143–152.
11. Rabin M. Fingerprinting by random polynomials. *Report TR-15-81*, Center for Research in Computing Technology, Harvard University, 1981.
12. Bharat K, Broder A. Mirror, mirror on the Web: A study of host pairs with replicated content. *Proceedings of the 8th International World Wide Web Conference*, May 1999. Elsevier, 1999; 1579–1590.
13. Page L, Brin S, Motwani R, Winograd T. The PageRank citation ranking: Bringing order to the Web. *Technical Report 1999-66*, Database Group, Stanford University, 1998.