Measuring Index Quality using Random Walks on the Web

Monika R. Henzinger   Allan Heydon   Michael Mitzenmacher   Marc Najork

Compaq Computer Corporation Systems Research Center
130 Lytton Ave.
Palo Alto, California 94301
E-mail: {monika,heydon,michaelm,najork}@pa.dec.com

Abstract

Recent research has studied how to measure the *size* of a search engine, in terms of the number of pages indexed. In this paper, we consider a different measure for search engines, namely the *quality* of the pages in a search engine index. We provide a simple, effective algorithm for approximating the quality of an index by performing a random walk on the Web, and we use this methodology to compare the index quality of several major search engines.

Keywords: search engines, index quality, random walks, PageRank.

1 Introduction

When search engines first appeared, they were often compared by quoting the number of pages they claimed to index. The more pages in an engine's index, the better was its claim for being the most comprehensive, and therefore the best, search engine. Because search engine size is a simple (and, especially in advertising, an effective) comparison metric, it is important to devise objective methods for determining, at least approximately, search engine sizes without depending on the figures provided by the search engines themselves. Only recently, however, has the question of how to develop independent tests of search engine size been tackled in a scientifically rigorous way [2, 13].

Although size is clearly an important measure for search engines, in this paper we claim that other measures may provide a more useful guide to search engine effectiveness. We suggest that quality, as well as quantity, is an important measure of a search engine. Intuitively, size is important when one seeks the answer for a specific query. For example, if one wishes to find the home page of a long-lost friend, then the important question is simply whether or not the search engine indexes that page. This is most likely dependent on the size of the search engine. If, however, one asks a broader query, where there are likely to be several possible matches, the important question is whether a search engine contains the pages most likely to be relevant to the user.[1] For example, if one wants to learn about a specific illness, then the important question is whether a search engine contains pages with high quality medical information. A recent study of AltaVista query logs show that over 60% of the non-empty queries use one or two words, and hence most queries are in fact likely to be broad [18].

[1] Another relevant question in this context is whether the search engine *returns* the important pages in a reasonable manner. This is the *ranking* question: in what order does a search engine provide results? Although this is clearly an important question, we do not consider ranking in this paper. Because ranking plays such an important role in user experience, we refer to the quality of the index (or the indexed pages) of a search engine, rather than to the quality of the search engine itself, in order to avoid confusion.

Another reason to focus on quality as opposed to size is that search engine sizes are currently growing at a slower rate than the Web itself [2]. A bottleneck for search engines is the memory and disk space required to store the growing number of pages and the processing power required to serve queries. Most Web pages, however, are unlikely to be of interest to an overwhelming majority of search engine users. Therefore, in the face of limited resources, it becomes imperative for search engines to index the most relevant and useful pages.

Furthermore, a search engine that indexes many pages will have to find an appropriate methodology for ranking such pages. Otherwise users may obtain too many results, most of which are not relevant to their search. Analysis of query logs has demonstrated that users are impatient, rarely examining more than the first page of results [18]. Hence search engines with high quality indexes may prove more useful to end users.

Indeed, a primary motivation for this work is our belief that a small but high quality index might satisfy user needs on broad queries better than a larger search engine with low quality pages. We therefore attempt to devise a reasonable definition for the quality of an index. In fact the definition we adopt is based on the definition of page quality defined by the PageRank ranking and applied in the Google search engine [6, 10]. This definition will be explained further in Section 2.3. Using our definition, we devise a simple methodology for approximating index quality that can be performed efficiently and easily by independent organizations not associated with any search engine. Our methodology is based on performing a random walk on the Web. We have attempted to design this quality test so that the resource requirements are low. In particular, one does *not* need to store large portions of the Web in order to perform the test.

Our quality measurements yield interesting results. We have found that search engines differ substantially in index quality, both in terms of the total quality of the indexed pages as well as the average quality per indexed page. Specifically, the total quality of the indexed pages is highest for AltaVista [1], while the average quality per page, i.e., the ratio of the total quality over the size, is highest for Lycos [14]. The Excite [9] search engine performs well in both regards.

The rest of the paper proceeds as follows. In Section 2, we introduce a general framework for measuring the quality of an index and provide the necessary background on the PageRank ranking and random walks. In Section 3, we discuss our methodology for approximating quality using random walks. Section 4 elaborates on a specific implementation of our approach, the results of which are presented and analyzed in Section 5.

2 Theoretical Foundations

2.1 Random Walks

We first provide some background on random walks. Let $X = \{s_1, s_2, \ldots, s_n\}$ be a set of states. A *random walk* on X corresponds to a sequence of states, one for each step of the walk. At each step, the walk switches from its current state to a new state or remains at the current state. Random walks are usually *Markovian*, which means that the transition at each step is independent of the previous steps and depends only on the current state.

For example, consider the following standard Markovian random walk on the integers over the range $\{0, \ldots, j\}$ that models a simple gambling game, such as blackjack, where a player bets the same amount on each hand (i.e., step). We assume that if the player ever reaches 0, they have lost all their money and stop, and if they reach j, they have won enough money and stop. Hence the process will stop whenever 0 or j is reached. Otherwise, at each step, one moves from state i (where $i \neq 0, j$) to $i+1$ with probability p (the probability of winning the game), to $i-1$ with probability

q (the probability of losing the game), and stays at the same state with probability $1 - p - q$ (the probability of a draw).

Our algorithm for approximating index quality will utilize a Markovian random walk on the Web, where each page in the Web represents a possible state. For the Web, a natural way to move between states is to follow a hyperlink from one page to another. This action will be a typical step in our random walk. To avoid getting stuck in cycles, the walk will also occassionally have to jump to a completely random page.

Unlike the example of the gambling game, our random walk will not have a natural stopping point. Instead, we can imagine our random walk continuing forever. In this case, an interesting feature of the walk is the fraction of steps the walk spends in each state. The *equilibrium distribution* of the walk specifies, for each state, the fraction of the steps the random walk would spend in each state if the random walk continued for an infinite amount of time. Another way to think of the equilibrium distribution is that it gives the probability that one would find a random walk in a given state at some step infinitely far in the future.

Of course we cannot allow our random walk to run for an infinite amount of time. In most well-behaved walks, however, the probabilities given by the equilibrium distribution are very close to the probabilities that result at some point far, but finitely far, into the future. We will use this fact in developing an algorithm for approximating index quality using random walks.

2.2 A Framework for Measuring Index Quality

We begin by providing a general definition of the *quality* of an index. Suppose each page p of the Web is given a *weight* $w(p)$. For convenience we assume the weights are scaled so that the sum of all weights is 1. Abusing notation slightly, we let S refer either to a search engine or to the set of pages indexed by a search engine. Then we may define the *quality* of the index S (with respect to w) $w(S)$ as

$$w(S) = \sum_{p \in S} w(p).$$

Note that since w is scaled, we always have $0 \leq w(S) \leq 1$.

This definition of quality can vary widely depending on the measure w chosen. For example, if the weights are equal for all pages, then the quality of an index would simply be proportional to its size. In this paper we will use a different weight function that better captures the notion of quality.

Note that regardless of the choice of w, according to our definition the quality of an index is to some extent related to its size. In particular, if the pages indexed by a search engine S_1 are a subset of the pages indexed by a search engine S_2, then S_2 will have at least as large a quality score as S_1 by our criterion. Thus, we introduce a second metric, the *average page quality* of an index. Let $|S|$ indicate the number of pages indexed by S. Then the average page quality of S is

$$a(S) = w(S)/|S|.$$

The average page quality provides insight into how well a search engine selects pages to index. Note, however, that this measure puts larger search engines at a disadvantage, since the more pages an engine contains, the harder it is to keep the average page quality high.

2.3 The PageRank Measure

A simple quality measure for a page is the number of pages that reference a page, or its *indegree*; this was suggested in the context of the Web by Carriere and Kazman [7] and was used by the Rankdex search engine [17]. The basic assumption underlying this quality measure is that a link from page p_1 to page p_2 means that the author of p_1 recommends p_2. The PageRank measure of a page suggested by Brin, Page, and others [6, 8, 16] is a recursive variation of this idea. The PageRank of a page depends not only on how many pages point to it, but on the PageRank (i.e., quality) of these pages as well. Thus, the PageRank of a page has the following intuitive properties: the PageRank of a page is high if it is linked to by many pages with a high PageRank, and a page with few links contributes more weight to the pages it links to than a page with many links.

Mathematically, the PageRank measure can be expressed by the following formula. Suppose there are T total pages on the Web. We choose a parameter d such that $0 < d < 1$; a typical value of d might lie in the range $0.1 \leq d \leq 0.15$. Let pages $p_1, \ldots, p_k$ link to page p, and let $C(p)$ be the number of links out of p. Then the PageRank $R(p)$ of a page is defined to satisfy:

$$R(p) = d \cdot \frac{1}{T} + (1-d) \cdot \sum_{i=1}^{k} \frac{R(p_i)}{C(p_i)}.$$

Note that the ranks $R(p)$ can be scaled so that $\sum R(p) = 1$, in which case $R(p)$ can be thought of as a probability distribution over pages and hence a weight function.

The PageRank distribution has a simple interpretation in terms of a random walk that will be useful to us. Imagine a Web surfer who wanders the Web. If the surfer visits page p, the random walk is in state p. At each step, the Web surfer either jumps to a page on the Web chosen uniformly at random, or the Web surfer follows a link chosen uniformly at random from those on the current page. The former occurs with probability d, the latter with probability $1-d$. The equilibrium probability that such a surfer is at page p is simply $R(p)$. This means that pages with high PageRank are more likely to be visited than pages with low PageRank.

The PageRank measure is used by the Google search engine as part of its ranking mechanism [6]. Also, estimates of the PageRank of a page can be used to guide a crawler, in order to have more important pages indexed first [8]. These works demonstrate that the PageRank measure captures the intuitive notion of page quality well, and hence it is a natural candidate for a weight function w for measuring index quality. For more details on PageRank and its previous uses, we refer the reader to the relevant papers [6, 8, 16].

2.4 A Random Walk Approach

We now describe an approach to approximating the quality of an index. Suppose one has a method for selecting pages from the Web so that the probability of selecting a particular page p is $w(p)$.

In that case, we can approximate the quality of a search engine index S according to the weight w by independently selecting pages $p_1, p_2, p_3, \ldots, p_n$, and testing whether each page is in S. We call this sequence of pages the *sample sequence*. Let $I[p_i \in S]$ be 1 if page p_i is in S, and 0 otherwise. Our estimate $\bar{w}(S)$ for $w(S)$ is

$$\bar{w}(S) = \frac{1}{n} \sum_{i=i}^{n} I[p_i \in S].$$

That is, we merely need to measure the fraction of pages in our sample sequence that are in the index S in order to approximate the quality of S. Note that the expectation of each $I[p_i \in S]$ is just $w(S)$, i.e.,

$$\mathbf{E}(I[p_i \in S]) = \sum_{p \in S} \mathbf{Pr}(p_i = p) = \sum_{p \in S} w(p) = w(S).$$

Thus, $\bar{w}(S)$ is the average of several independent binary random variables, each taking the value 1 with probability $w(S)$, which implies that

$$\mathbf{E}(\bar{w}(S)) = \frac{1}{n}\sum_{i=i}^{n}\mathbf{E}(I[p_i \in S]) = w(S).$$

Moreover, since each page in the sample sequence yields a binary random variable, we can provide standard statistical measures of the quality of our estimate, such as confidence intervals, etc.

Thus, in order to approximate the quality of a search engine index, we need two components. First, we need an appropriate method for selecting pages according to w. Second, we require a method for testing whether a page is indexed by a search engine.

To test whether a URL is indexed by a search engine, we adopt the approach used by Bharat and Broder [2]. Using a list of words that appear in Web documents and an approximate measure of their frequency, we find the k rarest words that appear in each document. We then query the search engine using a conjunction of these k rarest words and check for the appropriate URL. In our tests, we use $k = 9$. Following their terminology, we call such a query a *strong query*, as the query is designed to strongly identify the page.

Selecting pages according to a weight distribution w is significantly harder (assuming, of course, that it is difficult to crawl and store the *entire* Web!). Choosing a page uniformly at random from the Web, or even choosing a page just approximately uniformly at random, is a challenging open question [2]. As we discuss in more detail in the next section, it also appears difficult to select Web pages according to the PageRank distribution. However, we give in the next section an effective sampling algorithm that provides a quality measure close to that which would be obtained using the PageRank distribution. We then present evidence that our scheme does give a useful quality measure, and we give the quality values produced by our algorithm for several search engines.

3 Approximating Index Quality

3.1 Sampling Pages According to their PageRank

In this section we describe the problems encountered when trying to select pages according to the PageRank distribution, or some approximation thereof.

A first approach is to crawl a significant portion of the Web, and then determine the PageRank for the crawled pages using iterative methods; this is how the Google search engine computes PageRank values [6]. One could then sample pages according to this distribution.

One problem with this approach is that the weight measure obtained may not truly be the PageRank measure, but just an approximation based on the subgraph of the Web crawled by the crawler. It is not clear how good this approximation will be. Another major difficulty with this approach is that it requires crawling a large portion of the Web, storing the relevant link information, and calculating the appropriate PageRank values. We instead suggest an alternative approach based on the association between PageRank and a random walk over the Web.

3.2 Our Sampling Approach

Suppose one had a means of choosing a page uniformly at random. In that case, one could perform a random walk with an equilibrium distribution corresponding to the PageRank measure. At each step, the walk would either jump to a random page with probability d or follow a random link with probability $1 - d$. By executing this walk for a suitably long period of time, one could generate a sample sequence, by for example including each visited page in the sample sequence with a probability of $1/1000$. The pages in the sample sequence would have a distribution close to the

PageRank distribution. Notice that the random walk approach does not require storing a large portion of the Web graph; one need only keep track of the current Web page of the walk and the sample sequence. The sample sequence could be very small, as it corresponds to approximately 1/1000th of the pages visited on the walk; moreover, it can easily be logged to a disk file.

There are two problems with directly implementing such a random walk. First, as mentioned earlier, no method is known for choosing a Web page uniformly at random. Second, it is not clear how many steps one would have to perform in order to remove the bias of the initial state and thereby approximate the equilibrium distribution. (Recall that the equilibrium distribution corresponds to the distribution after an infinite number of steps.)

We approach these problems as follows: For the first problem, an obvious idea is to jump to a random page *encountered on the walk so far.* We implemented this solution, and found it less effective than we had hoped. The problem that arose was that initially, a random walk found its way into a site that had many links to itself and few to other sites. Hence, early on in the walk we found a large number of pages from this site, and this created a large bias to pages within this site for the rest of the walk.

Instead we adopted the following approach: instead of jumping to a random page, the walk occasionally chooses a host uniformly at random from the set of hosts encountered on the walk thus far, and jumps to a page chosen uniformly at random from the set of pages discovered on that host thus far. (There is one exception: if we find a page with no outgoing links, we do not record the host or the page.) Thus, like the approach we first tried, this approach randomly selects one of the pages visited thus far, but not uniformly so. This exception prevents restarting the walk at a dead end. In our simulations, this solution increased the spread, reducing the bias towards hosts with a large number of interconnected pages. Of course, the equilibrium distribution of this approach does not match the PageRank distribution, since pages that have not been visited yet cannot be chosen, and pages on hosts with a small number of pages are more likely to be chosen than pages on hosts with a large number of pages. However, our results suggest that this bias does not prevent our walk from approximating a good quality metric that behaves similarly to PageRank.

This solution requires keeping track of the URLs of all visited pages for the purpose of performing a random jump to a visited page. Although keeping track of the URLs visited for a long walk does require a lot of memory, one could mitigate this problem by limiting the number of URLs recorded. One way to do this would be to keep only the URLs visited most recently. Another possibility would be to store possible URLs to jump to probabilistically, recording URLs with some fixed probability. Yet another possibility would be to store the URLs on disk (at the cost of reduced performance).

Next we discuss the second problem, namely, that it is not clear how many steps one must perform in order to approximate the equilibrium distribution. The problem that we encounter is that we have to start our random walk from some initial page. This introduces a bias, and it is not clear how many steps one must perform for the bias to become sufficiently small. In fact, because we begin initially from the page www.yahoo.com[2], there may be a small bias towards pages close to the Yahoo home page; however, one way to view this effect is that our measure of quality gives a small advantage to sites close to Yahoo, which is not necessarily unreasonable.

When performing a random walk and trying to approximate PageRank to within a small error, probability theory suggests that randomly walking over a small subgraph of the Web suffices to handle the initial bias. This is because if the states are highly connected in a random walk (i.e, if from each state you can get to "many other" states in a few steps), then bounds on the number of steps required to achieve a distribution very close to the equilibrium distribution are known. For example, if every page on the Web linked to every other page, so that the Web graph was

[2]Throughout the text, we omit the prefix http:// where the meaning is clear.

fully connected in both directions, then only two steps would be necessary to remove almost all of the initial bias from the starting state. As another example, random walks are often considered on special graphs called expander graphs. For these graphs, the length of the walk need only be proportional to the logarithm of the size of the graph in order to achieve a distribution very close to the equilibrium distribution. Since it does appear that a substantial portion of the Web is highly connected, with reasonably short paths between pairs of pages, theory predicts that reasonably short runs suffice. Our results in Section 5 bear out this intuition.

Notice that this problem of initial bias is different from the problem encountered by the other possible approach, crawling a large portion of the Web and explicitly calculating PageRank for the resulting subgraph. If one does not gather the whole Web in this approach, then if an important part of the Web is missed, the approximate PageRank values obtained could be far from the true values.

In fact, we emphasize the point that becuase we gather a sample of pages based on a random walk, a relatively small crawl is sufficient for the random walk approach. That is, even if we actually crawl only one million pages, the *potential* number of pages that could have been visited over the course of our random walk is much larger than that. The sample we obtain should be thought of as being taken from the potentially visited pages, and not solely from the actually visited pages. Thus, assuming the biases we have considered are resonably small, a small crawl will be effective.

4 Our Random Walks

We performed our random walks using Mercator, an extensible, multi-threaded Web crawler written in Java [15]. We configured Mercator to use one hundred crawling threads, so it actually performs one hundred random walks in parallel, each walk running in a separate thread of control. When a walk randomly jumps to a random page instead of following a link, it chooses a host uniformly at random from all hosts seen by any thread so far, and then chooses a page on that host uniformly at random from all pages on that host seen by any thread so far (modulo the previously stated exception, see Section 3.2). The random walk algorithm is described in Figure 1.

This pseudo-code omits two details. The first is that our crawler follows HTTP redirects as necessary, up to at most 5 consecutive redirects. This limit is imposed to avoid redirect cycles. The second point is that we use only those pages p that are HTML pages. An HTML page is identified by a content type of text/html in the HTTP response header. Note that so long as we select a link u from U that cannot be downloaded or is not an HTML page, we continue looping until we find a valid outgoing link.

The results reported in the next section are based on two long random walks conducted independently. For both walks, we used a reset probability d of $\frac{1}{7}$. The reset probability corresponds to the damping parameter for PageRank. This value lies within the suggested range [6]. The sample sequence for our first walk consists of 1,015 URLs; for the second walk, the sample sequence consists of 1,100 URLs.

The first walk lasted 18 hours, during which time the crawler attempted to download 2,867,466 pages; 1,393,265 of the successfully downloaded pages were HTML pages, 509,279 of which were distinct. The second walk lasted 54 hours, during which time the crawler attempted to download 6,219,704 pages; 2,940,794 of the successfully downloaded pages were HTML pages, 1,002,745 of which were distinct. The high duplication rates are not surprising. Unlike a regular web crawl, in which duplicates are found only due to duplicated web content, our random walker may visit many pages multiple times. In fact, our whole approach relies on the idea that pages with high PageRank will be visited frequently.

```
The following variables are shared by all threads:
        HostSet, the set of host names discovered so far,
        UrlSet(h), the set of URLs discovered so far that belong to host h, and
        Samples, a list of URLs representing the sample sequence.

Their initial values are:
        HostSet = {www.yahoo.com}
        UrlSet(www.yahoo.com) = {www.yahoo.com}
        UrlSet(h) = {} for all other hosts h
        Samples = [ ]

All threads execute the following algorithm in parallel:

RandomReset:
        Choose a host h uniformly at random from HostSet.
        Choose a URL u uniformly at random from UrlSet(h).
        Download page p referred to by u.
Step:
        If p contains at least one link:
                Let h be the host component of u.
                If h is not in HostSet, add it.
                If u is not in UrlSet(h), add it.
        With probability c, add u to Samples.
        With probability d, go to RandomReset.
        Let U be the set of derelativized URLs contained in p.
        While U is non-empty:
                Choose and remove a URL u uniformly at random from U.
                Attempt to download page p referred to by u,
                 following HTTP redirects as necessary.
                If p could be downloaded and is an HTML document, go to Step.
        Go to RandomReset
```

Figure 1: Pseudo-code for the random walk algorithm.

5 Experimental Data and Analysis

In this section, we provide experimental results based on our random walks. Our results fall under two headings. First, we provide results to demonstrate that our random walk approach does capture the intuitive notion of quality. In particular, we show that the weight distribution appears heavily skewed toward pages users would consider useful. Second, we provide results comparing the measured quality of search engine indexes based on sampling from our random walks. We also discuss limitations of this approach, including possible biases and difficulties related to automating tests for whether a search engine contains a given document.

5.1 Random Walk Effectiveness

We provide evidence that our random walk approach does indeed capture an intuitive notion of quality. Since the intuitive concept of quality does not have a strict definition that can be objectively measured, most of the evidence will necessarily be circumstantial.

Page	W2 Freq.	W1 Freq. (Rank)
www.microsoft.com/	3172	1600 (1)
www.microsoft.com/windows/ie/default.htm	2064	1045 (3)
www.netscape.com/	1991	876 (6)
www.microsoft.com/ie/	1982	1017 (4)
www.microsoft.com/windows/ie/download/	1915	943 (5)
www.microsoft.com/windows/ie/download/all.htm	1696	830 (7)
www.adobe.com/prodindex/acrobat/readstep.html	1634	780 (8)
home.netscape.com/	1581	695 (10)
www.linkexchange.com/	1574	763 (9)
www.yahoo.com/	1527	1132 (2)
home.netscape.com/comprod/mirror/index.html	1015	479 (13)
www.lycos.com/	982	597 (11)
search.microsoft.com/default.asp	895	452 (15)
www.microsoft.com/search/default.asp	749	392 (17)
www.microsoft.com/Support/	721	388 (18)
www.adobe.com/homepage.shtml	690	361 (19)
www.excite.com/	678	436 (16)
www.infoseek.com/	676	320 (22)
www.microsoft.com/misc/cpyright.htm	673	355 (20)
www.microsoft.com/products/default.asp	663	343 (21)

Table 1: Most frequently hit pages on the random walks.

To begin, we show that our random walk reaches pages and hosts that are highly connected in the Web. In Tables 1 and 2, we list the twenty most visited pages and the twenty most visited hosts from the second, longer random walk (referred to as W2), along with their frequencies. We also present the frequencies for these hosts and pages from the first walk (W1), along with their rank order in frequency from the first walk.

From our theoretical framework in Section 2.3, one would expect our random walk to concentrate on pages and hosts that are linked to frequently, especially by other highly linked sites. Our frequency results demonstrate that this is indeed the case. In particular, our results demonstrate a high concentration on highly linked pages and hosts for major computer corporations. If one believes that PageRank accurately measures an intuitive measure of quality, then our random walk approach does as well.

The results are remarkably consistent over the two distinct walks, in that the most frequent hosts and pages appear in both lists, in close to the same order. This consistency further justifies our claim that our random walk approach captures the higher order phenomenon of quality. It is also worth noting that even though our search is biased initially by beginning at the page www.yahoo.com, this is neither the most frequently accessed page nor the most frequently accessed host. Moreover, in the longer walk W2, the rank for www.yahoo.com was lower than in that of the shorter walk W1, suggesting that the initial bias is reduced over time.

As a second test, we consider the indegree of the pages visited by our random walks. We determine the indegree using a prototype of the connectivity server described by Bharat *et. al.* in [4], which stores link information for the Web. We emphasize that the connectivity server provides only a lower bound on the number of links to a given page since it contains only about 180 million of the pages on the Web.

Site	W2 Freq.	W1 Freq. (Rank)
www.microsoft.com	32452	16917 (1)
home.netscape.com	23329	11084 (2)
www.adobe.com	10884	5539 (3)
www.amazon.com	10146	5182 (4)
www.netscape.com	4862	2307 (10)
excite.netscape.com	4714	2372 (9)
www.real.com	4494	2777 (5)
www.lycos.com	4448	2645 (6)
www.zdnet.com	4038	2562 (8)
www.linkexchange.com	3738	1940 (12)
www.yahoo.com	3461	2595 (7)
www.sun.com	2613	1309 (16)
www.hitbox.com	2570	1115 (19)
www.excite.com	2504	1644 (14)
members.aol.com	2450	1159 (18)
www.ibm.com	2418	1807 (13)
www.macromedia.com	2043	971 (23)
www.infoseek.com	2001	1005 (22)
www.compaq.com	1983	1079 (20)
www.digital.com	1927	1034 (21)

Table 2: Most frequently hit hosts on the random walks.

Several of the pages visited by our random walk, such as www.microsoft.com, have very high indegree. To avoid skewing caused by these pages, we consider only pages with indegree at most 1,000. For our second longer walk, the average indegree of the 719 pages from the 1,100 of the sample sequence found in the connectivity server with indegree at most 1,000 is just above 53; for the first walk, it is just over 60. Thus the average indegree of the pages visited by our walk is much larger than that of the average Web page [5], which provides further evidence that our random walk visits quality pages more frequently.

5.2 Comparing Search Engine Indexes

We used our technique to compare the quality of the AltaVista [1], Excite [9], Google [10], HotBot [11], Lycos [14], and Infoseek [12] search engine indexes. In this section we present the results of this comparison.

Recall that in comparing search engine indexes we use a strong query to identify whether an engine contains a given page. In practice, strong queries do not always uniquely identify a page; mirror sites, duplicates or near-duplicates of the page, or other spurious matches create difficulties. (See [2] for more discussion on this issue.)

To deal with these difficulties, we adopt an approach similar to [2]. In trying to match a URL with results from a search engine, all URLs are normalized by converting to lowercase, removing optional extensions such as index.htm[l] and home.htm[l], removing port numbers, and removing relative references of the form "#...". We also use three different matching criteria. A match is *exact* if the search engine returns a URL that, when normalized, exactly matches the normalized target URL. A *host* match occurs if a page with the same host as the target URL is returned.

Search Engine	Exact		Host		Non-zero		Est. Size (mill. pages)
	W1	W2	W1	W2	W1	W2	
AltaVista	0.2680	0.2709	0.3429	0.3409	0.5182	0.5164	125
HotBot	0.1517	0.1582	0.2128	0.2082	0.3764	0.3691	100
Excite	0.1675	0.1836	0.2227	0.2355	0.3892	0.3645	45
Infoseek	0.1025	0.1191	0.1399	0.1391	0.2374	0.2245	37
Google	0.0778	0.0764	0.1005	0.1036	0.2315	0.2191	25
Lycos	0.1281	0.1264	0.1606	0.1691	0.3005	0.2891	21

Table 3: Search engine index quality estimates $\bar{w}(S)$ from two walks.

Finally, a *non-zero* match occurs if a search engine returns any page as a result of the strong query. Host matches and non-zero matches will overestimate the number of actual matches; however, the number of exact matches may be an underestimate if a search engine removes duplicate pages or if the location of a Web page has changed. We report results using all three matching criteria, since there is some variance among the three. We note that the number of non-zero matches appears to be a rather large overestimate; the number of host matches and exact matches are much closer.

In Table 3, we report our results from testing several major search engines to obtain estimates $\bar{w}(S)$ of their quality $w(S)$, ordering the results according to the size of each search engine. (The size estimates, with the exception of the Google engine, are from Broder and Bharat, representing a size test done in July 1998 [3]. For Google's size, we used the estimate of 25 million pages given at the Google page http://google.stanford.edu.) Each score is the maximum of three different tests, since (due to machine load and other transient problems) search engines do not always return the same results over time. Recall that each score corresponds to the fraction of matches obtained from the URLs in our sample sequence. Figure 2 presents a visual interpretation of some of the data. Each bar represents the page quality score for the exact matches of a search engine. The values from each walk have been scaled so that the largest bar (AltaVista) for each walk has height 1. The estimated size of each search engine (in millions of pages) is listed below each for easy reference.

While there is some correlation between size and our quality metric, they are clearly not proportional. AltaVista scores the highest under the quality metric, and its superiority to HotBot is greater than one would expect from the difference in size. Both Excite and Lycos perform better than one might expect from their size. In particular, they both achieve higher scores than other significantly larger search engines. These results suggest that Excite and Lycos may be actively attempting to index pages more likely to be of high quality.

In Figure 3, we compare the average page quality. Each bar represents the average page quality for exact matches of a search engine, again scaled so that the largest bar (Lycos) for each walk has height 1. We again emphasize that such a measurement naturally favors small search engines, as larger search engines necessarily have to take some lower quality pages while small search engines can be more selective. (Consider the case of an engine that indexes a single page, www.microsoft.com!) Therefore, although smaller search engines generally perform better than larger ones with regard to this measure, this is to be expected. Even bearing this in mind, Lycos's average page quality appears quite high in comparison with the other search engines.

Another interesting point revealed in this figure is that in some cases, larger search engines do appear to have higher average page quality. For example, AltaVista ranks higher than HotBot, and Excite ranks higher than Infoseek and Google. In these cases, the difference in scaled average page quality scores is surprising, again suggesting fundamental differences in the algorithms used by each engine to determine which pages to index. The results are entirely similar for host matches.

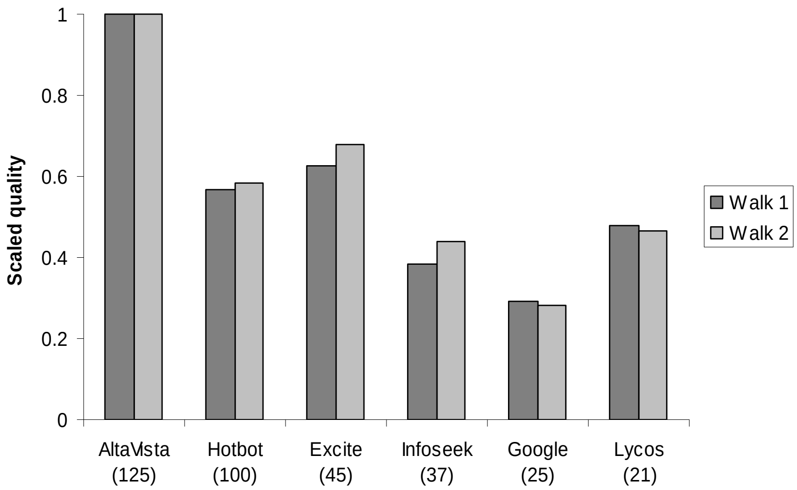

Figure 2: The quality scores for various search engine indexes, scaled, in the case of exact matches. Numbers in parentheses are the estimated number of pages indexed by the search engine as of July 1998.

(For non-zero matches, Google's performance appears slightly better.)

We highlight some of the main points of the data:

- Our two independent random walks provide nearly the same results. This gives us confidence that our results are repeatable and that our numbers appear reasonable.
- We are walking for a sufficient amount of time and sample sufficiently many pages to remove biases. In Figure 4, we show for example how our cumulative estimates of $\bar{w}(S)$ for the exact matches for AltaVista develop over the sample sequence of the first walk W1. After only a few hundred pages, the fluctuations in the estimate appear relatively small. Similar results hold for other measurements.
- The host matches and exact matches are relatively close; the non-zero matches, however, appear to overestimate matches considerably. Our experience suggests that strong queries often return spurious non-zero matches. We believe the score based on the exact matches is the most accurate.
- Comparing the quality scores of the search engine indexes in groups according to their size, AltaVista does substantially better than HotBot; Excite does extremely well for a search engine of intermediate size, achieving a higher quality score than HotBot; and Lycos appears to have more than its share of quality pages, given its small size.
- Comparing the average page quality of the search engine indexes, one finds that larger search engines sometimes have higher average page quality. This fact strongly suggests that the different methods used by a search engine for crawling or indexing can have a significant effect on search engine index quality.

Our measurements suggest that at least some of the major search engines are utilizing some quality metric, either to guide their crawlers or to ensure that only pages of sufficiently high quality

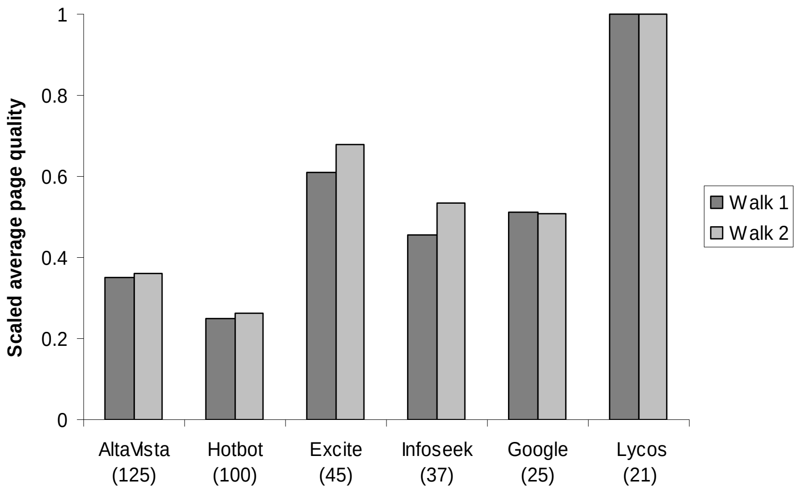

Figure 3: The average page quality for various search engine indexes, scaled, in the case of exact matches. Numbers in parentheses are the estimated number of pages indexed by the search engine as of July 1998.

are indexed. It would be interesting to learn what metrics search engine companies use, if any. It may well be the case that our quality metric is highly correlated with other simpler quality metrics, such as indegree or word content.

We reiterate that our quality measurements do not necessarily correspond to the total user experience in using a search engine. Other aspects, such as the ranking function used to order query results, could have a more dramatic effect. However, like a size measurement, our quality measurements provide insight into how search engines perform and how they are constructed.

5.3 Limitations of our Methodology

In this section, we remind the reader of some of the limitations of our approach. We also discuss how we handle some of the challenges in making accurate search engine measurements.

First, at a high level, our random walk yields a measure different than the actual PageRank measure, both because of bias from the initial starting point, and because our walk chooses a "random" URL to jump to in a non-uniform way (that is effective in practice). Moreover, our methodology cannot itself justify that our quality measure adequately captures the intuitive notion of quality from a user standpoint. To argue this we rely on ancillary evidence.

At an implementation level, the need to check whether a search engine contains a given page presents some difficulties, and these difficulties are exacerbated when one attempts to automate the process for repeated tests. For example, the strong query approach does not always uniquely identify a URL. We dealt with this problem by providing results for three different types of matches. Search engines may return different answers at different times, depending on load conditions or other variables. Therefore we performed our search engine tests at night and on holidays, and we took the maximum over three distinct tests for each search engine. We note that there was little variance among tests. Trying to cope with special characters in various languages can also skew results; in some cases these special characters might have caused us to miss matches. Because only a small percentage of our sample sequence was in a foreign language, and because extra misses caused by

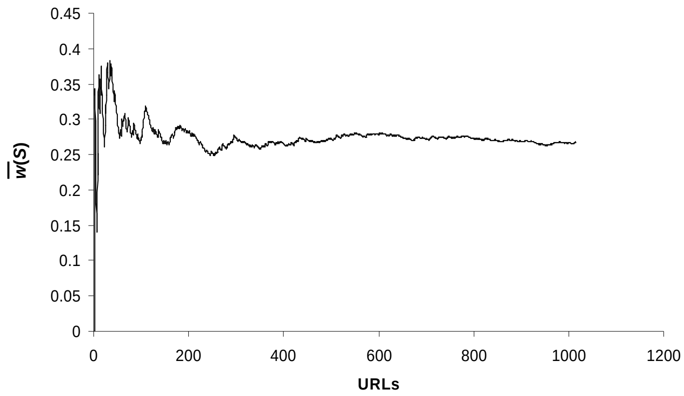

Figure 4: The development of $\bar{w}(S)$ for AltaVista over the 1,015 URL's in the sample sequence. Deviations in our estimate become small quickly.

special characters tend to affect all search engines, we believe the effect of this problem is very small. To simplify the checking of whether a page was indexed when several pages of URLs are returned, we limited all search engines to 100 returns. Again, this may affect the number of exact matches, but the number of cases where over 100 URLs are returned is so small that we do not believe the problem is significant.

For all of these reasons, our numbers should be taken as approximate. On the other hand, the consistency of our results over two separate long runs suggests that our quality measurements are repeatable. Moreover, our conclusions about search engine scores would remain valid even if the true scores varied within a few percent of those presented. We therefore feel that despite these limitations, our results reveal valuable information about the contents of various search engines.

6 Conclusions

We have introduced a general definition of the quality of a search engine index with respect to a given weight function, and we have suggested the use of a measure based on the PageRank measure for quantifying search engine index quality. We have also described a methodology for approximating search engine index quality using a random walk on the Web. Results obtained from implementing our approach provide evidence that it does capture an intuitive notion of quality.

Search engine measurements based on our implementation suggest that search engine indexes differ substantially in quality. These differences appear whether one considers the total quality or the average quality per page of the search engines. Based on our results, we believe that some search engines appear to be making a greater effort to index only higher quality pages, and that this effort does effectively lead to improved quality. In particular, our results suggest that the Excite search engine currently appears to emphasize high quality pages. Smaller search engines such as Lycos also appear to do a good job of emphasizing quality. For large search engines, AltaVista appears significantly more attuned to quality than HotBot.

The issue of quality raises several questions. First, what are good, useful quality weight functions? For this paper we have adopted a measure based on the PageRank quality measure, but others are possible. It would also be interesting to see how various quality criteria are correlated. Second, how can one improve the quality of a search engine index? This question is also the motivation of [8], who suggest using the PageRank ranking to order how a crawler accesses pages, but we believe a great deal more could be done in this area. Third, what is the overall impact of quality on the user experience? For this question, it would be interesting to consider the tradeoff between quality and size for real user queries.

References

[1] AltaVista, `http://www.altavista.com/`

[2] K. Bharat and A. Broder. *A technique for measuring the relative size and overlap of public web search engines.* In Proceedings of the 7th International World Wide Web Conference, Brisbane, Australia, pages 379-388. Elsevier Science, April 1998.

[3] K. Bharat and A. Broder. Private communication.

[4] K. Bharat, A. Broder, M. Henzinger, P. Kumar, and S. Venkatasubramanian. *The Connectivity Server: fast access to linkage information on the Web.* In Proceedings of the 7th International World Wide Web Conference, Brisbane, Australia, pages 469-477. Elsevier Science, April 1998.

[5] T. Bray. *Measuring the Web.* Available at http://www5conf.inria.fr/fich_html/papers/P9/Overview.html. The World Wide Web Journal 1(3), 1996.

[6] S. Brin and L. Page. *The Anatomy of a Large-Scale Hypertextual Web Search Engine.* In Proceedings of the 7th International World Wide Web Conference, Brisbane, Australia, pages 107-117. Elsevier Science, April 1998.

[7] J. Carriere and R. Kazman. Webquery: Searching and visualizing the web through connectivity. In *Proceedings of the Sixth International World Wide Web Conference*, Santa Clara, California, April 1997, pages 701–711.

[8] J. Cho, H. Garcia-Molina, and L. Page. *Efficient Crawling Through URL Ordering.* In Proceedings of the 7th International World Wide Web Conference, Brisbane, Australia, pages 161-172. Elsevier Science, April 1998.

[9] Excite, `http://www.excite.com/`

[10] Google, `http://google.stanford.edu/`, see also `http://www.google.com/`

[11] HotBot, `http://www.hotbot.com/`

[12] Infoseek, `http://www.infoseek.com/`

[13] S. Lawrence and C. L. Giles. *Searching the World Wide Web.* Science, 280(536):98, 1998.

[14] Lycos, `http://www.lycos.com/`

[15] Mercator Home Page, `http://www.research.digital.com/SRC/mercator/`

[16] L. Page, S. Brin, R. Motwani, and T. Winograd. *The PageRank citation ranking: Bringing order to the Web.* Work in progress. URL: http://google.stanford.edu/~backrub/pageranksub.ps.

[17] Rankdex, `http://rankdex.gari.com/`

[18] C. Silverstein, M. Henzinger, J. Marais, and M. Moricz. *Analysis of a very large AltaVista query log.* Technical Report 1998-014, Compaq Systems Research Center, Palo Alto, California, 1998.

About the Authors

Monika R. Henzinger received her Ph.D. from Princeton University in 1993 under the supervision of Robert E. Tarjan. Afterwards, she was an assistant professor in Computer Scinece at Cornell University. She joined the Digital Systems Research Center (now Compaq Computer Corporation's Systems Research Center) in 1996. Her current research interests are information retrieval on the World Wide Web and algorithmic problems arising in this context.

Allan Heydon received his Ph.D. in Computer Science from Carnegie Mellon University, where he designed and implemented a system for processing visual specifications of file system security. In addition to his recent work on web crawling, he has also worked on the Vesta software configuration management system, the Juno-2 constraint-based drawing editor, and algorithm animation. He is a member of the research staff at Compaq Computer Corporation's Systems Research Center.

Michael Mitzenmacher received his Ph.D. in Computer Science from the University of California at Berkeley in 1996. He then joined the research staff of the Compaq Computer Corporation's Systems Research Center. Currently he is an assistant professor at Harvard University. His research interests focus on algorithms and random processes. Current interests include error-correcting codes, the Web, and distributed systems.

Marc Najork is a member of the research staff at Compaq Computer Corporation's Systems Research Center. His current research focuses on 3D animation, information visualization, algorithm animation, and the World Wide Web. He received his Ph.D. in Computer Science from the University of Illinois at Urbana-Champaign in 1994, where he developed Cube, a three-dimensional visual programming language.