User-Centric Web Crawling

Sandeep Pandey
Carnegie Mellon University
Pittsburgh, PA 15213
spandey@cs.cmu.edu

Christopher Olston
Carnegie Mellon University
Pittsburgh, PA 15213
olston@cs.cmu.edu

ABSTRACT

Search engines are the primary gateways of information access on the Web today. Behind the scenes, search engines crawl the Web to populate a local indexed repository of Web pages, used to answer user search queries. In an aggregate sense, the Web is very dynamic, causing any repository of Web pages to become out of date over time, which in turn causes query answer quality to degrade. Given the considerable size, dynamicity, and degree of autonomy of the Web as a whole, it is not feasible for a search engine to maintain its repository exactly synchronized with the Web.

In this paper we study how to schedule Web pages for selective (re)downloading into a search engine repository. The scheduling objective is to maximize the quality of the user experience for those who query the search engine. We begin with a quantitative characterization of the way in which the discrepancy between the content of the repository and the current content of the live Web impacts the quality of the user experience. This characterization leads to a *user-centric* metric of the quality of a search engine's local repository. We use this metric to derive a policy for scheduling Web page (re)downloading that is driven by search engine usage and free of exterior tuning parameters. We then focus on the important subproblem of scheduling refreshing of Web pages already present in the repository, and show how to compute the priorities efficiently. We provide extensive empirical comparisons of our user-centric method against prior Web page refresh strategies, using real Web data. Our results demonstrate that our method requires far fewer resources to maintain same search engine quality level for users, leaving substantially more resources available for incorporating new Web pages into the search repository.

Categories and Subject Descriptors

H.3 [**Information Systems**]: Information Storage and Retrieval; D.2.8 [**Software Engineering**]: Metrics—*performance measures*

General Terms

Performance, Design, Experimentation

Keywords

Web crawling, Web page refreshing, user-centric

WWW 2005, May 10-14, 2005, Chiba, Japan.
ACM 1-59593-046-9/05/0005.

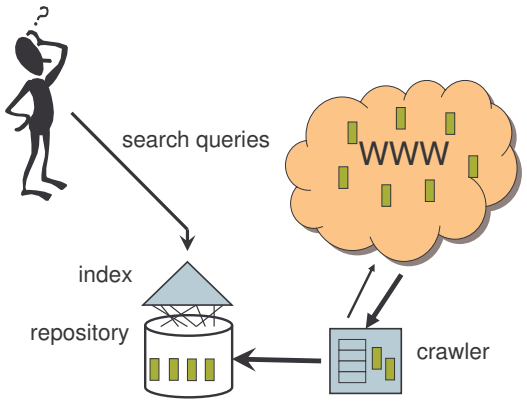

Figure 1: Basic Web search engine architecture.

1. INTRODUCTION

Web search engines, taken together, handle hundreds of millions of queries each day [19]. They compile responses to user queries by accessing a local *repository* that mirrors the Web, as shown in Figure 1. When a user submits a query, usually in the form of a list of textual terms, an internal *scoring function* is applied to each Web page in the repository (in practice an inverted index is used to speed up this process). Applying this function to a page produces a numerical score, representing the best available estimate of the usefulness of the page to the user who submitted the query. Query results are usually presented in the form of a sequential list of links to pages arranged in descending order of score. When the user clicks on a link in the query result list, her Web browser fetches the current copy of the linked page from the live Web.[1]

Two factors govern the quality of a search engine, from the user's perspective:

1. **Scoring function suitability:** A plethora of heuristics are used to estimate the usefulness of viewing a Web page as a result of issuing a particular query, including TF-IDF [18], anchortext inclusion [6], and link analysis [14, 17]. Clearly, use of scoring functions that are better at estimating actual usefulness leads to a higher-quality user experience, on the whole.

[1]In this paper we assume that the search engine does not serve copies of Web pages directly from its repository.

2. **Repository freshness:** A search engine uses its local repository to assign scores to the Web pages in response to a query, with the implicit assumption that the repository closely mirrors the current Web. However, it is infeasible to maintain an exact mirror of a large portion of the Web due to its considerable aggregate size and dynamicity [16], combined with the autonomous nature of Web servers. If the repository is not closely synchronized with the Web, then the search engine may not include the most useful pages for a query at the top of the result list. Since users' attention is strongly biased toward the top of query result lists [13, 15] and they have limited time to inspect results, users are likely to visit Web pages that are on the whole less useful than the ones they would visit if presented with a hypothetical result list generated from a fully synchronized repository.

In this paper we focus on the second factor, and assume that a reasonable scoring function is in use. Our work is in the context of an *incremental* way of crawling the Web in which the repository is updated on the fly as Web pages are downloaded [10]. We assume an incremental crawler that permits fine-grained control over (re)downloading of individual Web pages. Our goal is to devise a policy for scheduling downloading and redownloading of Web pages into the repository so as to maximize the overall quality of the user experience.

1.1 Impact of Repository Freshness on User Experience

We present a simple example to illustrate the ways in which an out-of-date repository can adversely impact the user experience. Before proceeding we introduce some notation. Let $\mathcal{W}$ refer to the current collective content of the Web, and let $\mathcal{W}^L$ refer to the collective content of the search engine's local repository. For a Web page p, $\mathcal{W}[p]$ denotes the current live Web copy of p, and $\mathcal{W}^L[p]$ denotes whatever copy of p (if any) is currently stored in the repository.

Now, suppose the Web is tiny, such that $\mathcal{W} = \{\mathcal{W}[p_1], \mathcal{W}[p_2], \mathcal{W}[p_3]\}$. Suppose the search repository $\mathcal{W}^L$ contains copies of two of the three pages currently available on the Web (namely, pages p_1 and p_2), plus a copy of one more page p_4, which has been removed from the Web since it was last downloaded into the repository. Hence, $\mathcal{W}^L = \{\mathcal{W}^L[p_1], \mathcal{W}^L[p_2], \mathcal{W}^L[p_4]\}$. Suppose furthermore that the repository copies of p_1 and p_2 are both out of date, such that $\mathcal{W}^L[p_1] \neq \mathcal{W}[p_1]$ and $\mathcal{W}^L[p_2] \neq \mathcal{W}[p_2]$. The content of each copy of each page is shown in Table 1.

Consider the query "cancer." For the sake of our example assume a simple Boolean scoring function that returns `true` if there is a keyword match, and `false` otherwise. Observe four types of discrepancies between the repository and the live Web, each of which leads to distorted results for this query: (1) Web pages with increased score not yet reflected in the repository, e.g., p_1, (2) pages with decreased score, e.g., p_2, (3) pages not yet incorporated into the repository, e.g., p_3, and (4) pages that have been removed from the Web but remain present in the repository, e.g., p_4.

Of course, with real search engines the number of matches for a given query frequently number in the thousands or millions. Users typically focus their attention on the top few results [13, 15], so the crucial factor governing the quality of search results is the order in which links to result pages are presented to the user. Next we introduce a model of search result quality that takes into account the bias in viewing likelihood, and use it to derive a metric for the quality of a search engine's repository with respect to the user experience. Then we describe how this metric forms the basis for our new, user-centric Web crawling paradigm.

Page p	**Web copy** $\mathcal{W}[p]$	**Search engine copy** $\mathcal{W}^L[p]$
p_1	New Technology: A new thyroid cancer therapy	New Technology: A new chipset designed for cell phones
p_2	Seminar: Important traffic laws and rules	Seminar: Cancer symptoms
p_3	Cancer Management: Early tests to detect breast cancer	(*Not present in the repository*)
p_4	(*Removed from the Web*)	Cancer association seeking volunteers to help raise awareness

Table 1: Example scenario.

1.2 User-Centric Search Repository Quality Metric

We begin by introducing some additional notation. Let $A(q, \mathcal{W}^L)$ denote the answer provided by a search engine in response to query q, which we assume is in the form of a ranked list, compiled according to scores computed over copies of Web pages stored in the local repository $\mathcal{W}^L$. Let $S(\mathcal{W}^L[p], q)$ denote the result of applying the search engine's scoring function S to the locally-available repository copy of p for query q. Similarly, let $S(\mathcal{W}[p], q)$ denote the result of applying the same scoring function to the live Web copy $\mathcal{W}[p]$ of page p for query q. Recall that we assume the scoring function provides an estimate of the usefulness of a page to a user who submits a particular query.

If $V(p, a)$ denotes the likelihood with which a typical user would view page p if presented with result list a (most likely influenced strongly by the rank position of p within a), then we can express the expected cumulative usefulness of the search engine's answer $a = A(q, \mathcal{W}^L)$ to query q as:

$$k \cdot \sum_{p \in a} V(p, a) \cdot S(\mathcal{W}[p], q)$$

where k is an arbitrary constant of proportionality. If we expect a certain workload $\mathcal{Q}$ of queries, with each query $q \in \mathcal{Q}$ issued with frequency f_q, we can write the expected average usefulness of querying the search engine as:

$$\sum_{q \in \mathcal{Q}} f_q \cdot k \cdot \sum_{p \in A(q, \mathcal{W}^L)} V(p, A(q, \mathcal{W}^L)) \cdot S(\mathcal{W}[p], q)$$

We model the *quality* of a search repository $\mathcal{W}^L$ with respect to a particular scoring method $S()$ and an expected usage pattern (query workload $\mathcal{Q}$ and viewing likelihood function $V()$) as a scalar value $Q(\mathcal{W}^L)$. In particular we define $Q(\mathcal{W}^L)$ to be directly proportional to expected average usefulness:

$$Q(\mathcal{W}^L) \propto \sum_{q \in \mathcal{Q}} f_q \cdot \sum_{p \in \mathcal{W}} V(p, A(q, \mathcal{W}^L)) \cdot S(\mathcal{W}[p], q)$$

(Assume $V(p, a) = 0$ for pages p not present in result list a.)

Note that assessing repository quality via application of a scoring function places more stringent requirements on the choice of scoring function than if it were to be used solely for result ranking purposes (as is the traditional usage). Tuning the scoring function is the subject of research on Factor 1 stated above, which is beyond the scope of this paper.

1.3 Web Page (Re)download Prioritization

Based on the above *user-centric* model of search repository quality we propose a new Web crawler scheduling policy that prioritizes (re)downloading of Web pages based on the expected gain in repository quality. The main difficulty is that the benefit of downloading a Web page can only be measured after it has been downloaded. Hence the principal challenge is how to estimate the expected improvement in repository quality if a particular page were to be downloaded, without downloading it. In this paper we focus on estimating the benefit of *re*-downloading pages already present in the repository. We show that the benefit of redownloading a page can be estimated fairly accurately from the measured improvement in repository quality due to past downloads of the same page. However, naïve methods of measuring the improvement in repository quality due to downloading a new or updated page are extremely inefficient—in practice they would cripple a Web crawler. We propose a novel approximation scheme for this purpose, coupled with an implementation technique in which measurements are taken in conjunction with index maintenance operations. Our technique is efficient enough to use in an operational Web crawler.

1.4 Contributions

The specific contributions of this paper are as follows:

- We propose a new metric of the quality of a search engine's local repository of Web pages, defined with respect to the quality of the user experience. (Section 3)
- We establish a new incremental Web crawling paradigm, called user-centric crawling, in which the objective is to maximize the quality of average user's search experience directly. (Section 4)
- We provide an efficient method of measuring the approximate impact of (re)downloading a Web page into the local repository, in terms of improvement in the quality of the user experience. Our method is tightly integrated with the process of updating an inverted index that is maintained over the repository, and incurs little additional overhead. (Section 5)
- We evaluate the effectiveness of our user-centric Web page refresh scheduling policy empirically using real Web data. In particular we show that the improvement in quality yielded by downloading a particular page is fairly consistent across time, making our approach feasible. We also compare our policy against prior Web page refreshing schemes, and show that our policy makes much more effective use of resources when measured according to a user-centric notion of repository quality. (Section 6)

2. RELATED WORK

Web crawling is a well-studied research problem. The subproblem of scheduling page refreshing under resource constraints has been studied in [9, 21]. In [9], the optimization objective is to maximize the average *freshness* or minimize the average *age* of pages in the repository, treating all pages and changes to pages with uniform importance. Unlike in our work, neither the manner in which pages change nor the way in which users query and view results are considered.

In [21] a metric that assesses the level of "embarrassment" to the search engine was proposed, along with a corresponding page refreshing policy. In the model of [21], embarrassment accrues whenever a user clicks on a search result link, only to discover that the destination page is not, in fact, relevant to the query she had issued. While a search engine with a high embarrassment level clearly does not provide quality service to its users, minimizing (or even eliminating) embarrassment is not all that is needed to ensure a good user experience. Consider that the omission of high-quality, relevant documents from search results generates no embarrassment, although it can degrade the quality of the user experience substantially (of course the user may not "know what she is missing"). This point illustrates the difference in philosophy between embarrassment-based crawling and our user-centric crawling paradigm. We provide a thorough empirical comparison of our page refresh scheme with those of [9] and [21] in Section 6.

Work on *focused crawling* [8] concentrates on how to obtain an initial crawl of the portion of the Web likely to be of interest to a particular community of users. Our work is complementary. Our user-centric approach to incremental crawling can be used to keep the repository of a focused search engine up-to-date as the Web evolves.

3. SEARCH REPOSITORY QUALITY

Recall from Section 1.2 that in our user-centric model, the *quality* of search repository $\mathcal{W}^L$ is expressed as:

$$Q(\mathcal{W}^L) \propto \sum_{q \in \mathcal{Q}} f_q \cdot \sum_{p \in \mathcal{W}} V(p, A(q, \mathcal{W}^L)) \cdot S(\mathcal{W}[p], q) \qquad (1)$$

where $V(p, a)$ denotes the likelihood of a user viewing page p when presented with result list a. Empirical measurements taken during an extensive user study [13] indicate that the expected viewing likelihood $V(p, a)$ depends primarily on the rank of p in a, denoted $R(p, a)$. This property appears to stem from the tendency of users to scan search result lists linearly starting from the top, regardless of the content of the list [13]. Furthermore, users typically cease exploration well before reaching the end of the list, especially for very large result sets. In light of these observations we model viewing likelihood purely as a function of rank, so that $V(p, a) = I(R(p, a))$ for some function $I(r)$. We believe this model serves as a reasonable first-order approximation of true user behavior (the same model was adopted in [11]). The function $I(r)$ can be estimated by monitoring user behavior and fitting a curve. For example, AltaVista usage logs analyzed in [11, 15] reveal that the following relationship holds quite closely:

$$I(r) = c \cdot r^{-3/2} \qquad (2)$$

where c is a normalization constant.[2] By substituting into Equation 1 we obtain:

$$Q(\mathcal{W}^L) \propto \sum_{q \in \mathcal{Q}} f_q \cdot \sum_{p \in \mathcal{W}} I(R(p, A(q, \mathcal{W}^L))) \cdot S(\mathcal{W}[p], q) \quad (3)$$

(The rank of a page not present in a result list is taken to be ∞, with $I(\infty) = 0$.)

3.1 Ideal Repository Quality

It is instructive to formulate an expression for the upper bound on search repository quality. As long as the inspection likelihood function $I(r)$ is monotonically nonincreasing, the expected cumulative score of visited pages is maximized when pages are always presented to users in descending order of their true score $S(\mathcal{W}[p], q)$. This ideal situation occurs when a search engine's repository is exactly synchronized with the Web at all times, such that $\mathcal{W}^L = \mathcal{W}$. Hence, we denote the highest possible search repository quality as $Q(\mathcal{W})$, where:

$$Q(\mathcal{W}) \propto \sum_{q \in \mathcal{Q}} f_q \cdot \sum_{p \in \mathcal{W}} I(R(p, A(q, \mathcal{W}))) \cdot S(\mathcal{W}[p], q) \quad (4)$$

It is not difficult to construct a formal proof that presenting search results in descending order of true score (based on the live Web copy) does indeed achieve a tight upper bound on quality. To understand intuitively why it is the case that ranking results in any other order results in a lower quality score, consider the following two cases. First, if a page is assigned a worse rank than its true score reflects, users will reach that page less often, statistically, than they would had the page been ranked correctly. Second, if a page is assigned a better rank than it merits based on its true score, users will tend to visit that page at the expense of not visiting other pages with higher scores. Presenting results in descending order of true score makes most effective use of users' limited attention span.

3.2 Normalized Quality Metric

It is convenient to represent search repository quality on a known, bounded scale. Hence we define the quality of repository $\mathcal{W}^L$ relative to the upper bound on quality corresponding to the case in which $\mathcal{W}^L = \mathcal{W}$, such that $Q(\mathcal{W}^L) \in [0, 1]$. In this way we arrive at our final, normalized expression for $Q(\mathcal{W}^L)$:

$$Q(\mathcal{W}^L) = \frac{\sum_{q \in \mathcal{Q}} f_q \cdot \sum_{p \in \mathcal{W}} I(R(p, A(q, \mathcal{W}^L))) \cdot S(\mathcal{W}[p], q)}{\sum_{q \in \mathcal{Q}} f_q \cdot \sum_{p \in \mathcal{W}} I(R(p, A(q, \mathcal{W}))) \cdot S(\mathcal{W}[p], q)} \quad (5)$$

Observe that in practice it is effectively impossible to compute the exact quality value of a large repository of Web pages. Measuring $Q(\mathcal{W}^L)$ exactly would require access to a fully up-to-date snapshot of the corresponding pages on the live Web, and obtaining such a snapshot is precisely the problem we are trying to solve. Our quality metric serves primarily as a conceptual tool for now; we will explain how to translate it into a practical implement later in Section 5.

[2]User views were measured at the granularity of groups of ten results in [15], and later extrapolated to individual pages in [11].

3.3 Change in Quality

To motivate and describe our user-centric Web crawling scheme we require a metric for the change in repository quality upon (re)downloading the latest copy of a particular Web page into the repository. We extend our notation to incorporate time as follows. Let $\mathcal{W}_t$ and $\mathcal{W}_t^L$ refer to the state of the live Web and of the local repository, respectively, at time t. Now consider a page p and let $\mathcal{W}_t^{L+p}$ refer to the state of the repository if it is altered by incorporating the latest version of p, such that $\mathcal{W}_t^{L+p}[p] = \mathcal{W}_t[p]$. (We assume for simplicity of our formal notation that the process of downloading a page and incorporating it into the repository occurs instantaneously.) We define the *change in repository quality* $\Delta Q(p, t)$ due to downloading page p at time t as:

$$\begin{aligned} \Delta Q(p, t) &= Q(W_t^{L+p}) - Q(W_t^L) \\ &= \frac{\sum_{q \in \mathcal{Q}} f_q \cdot \sum_{p' \in \mathcal{W}} \Delta I(p', q, \mathcal{W}_t^L, \mathcal{W}_t^{L+p}) \cdot S(\mathcal{W}_t[p'], q)}{\sum_{q \in \mathcal{Q}} f_q \cdot \sum_{p' \in \mathcal{W}} I(R(p', A(q, \mathcal{W}_t))) \cdot S(\mathcal{W}_t[p'], q)} \end{aligned} \quad (6)$$

where $\Delta I(p, q, \mathcal{W}_1, \mathcal{W}_2)$ denotes the change in the expected frequency with which users inspect page p as a consequence of issuing query q, if repository $\mathcal{W}_2$ is used instead of $\mathcal{W}_1$ to construct query answers. Formally:

$$\Delta I(p, q, \mathcal{W}_1, \mathcal{W}_2) = I(R(p, A(q, \mathcal{W}_2))) - I(R(p, A(q, \mathcal{W}_1))) \quad (7)$$

As an aside, we highlight two important yet subtle characteristics of $\Delta Q(p, t)$. First, the value of $\Delta Q(p, t)$ for a given page p depends on the current state of the Web at large ($\mathcal{W}_t$), because our quality metric is normalized relative to the quality of a hypothetical ideal search engine that has perfect and instantaneous access to the live Web. Second, $\Delta Q(p, t)$ also depends on the current state of pages other than p in the search engine repository $\mathcal{W}_t^L$. Consequently, if we consider two pages p_1 and p_2 that are downloaded nearly simultaneously although in some serial order, the improvement in quality attributed to the action of downloading each page may depend on the order in which they are downloaded. Both of these characteristics imply the following property: Given a page p and two moments of time t_1 and t_2 such that page p is never updated or downloaded during the interval $[t_1, t_2]$ (i.e., both $\mathcal{W}_t[p]$ and $\mathcal{W}_t^L[p]$ remain unchanged for all $t \in [t_1, t_2]$), it is not necessarily the case that $\Delta Q(p, t_1) = \Delta Q(p, t_2)$.

4. USER-CENTRIC WEB CRAWLING

Our *user-centric* Web crawling scheme is driven directly by our user-centric metric of search repository quality introduced in Section 3. Given limited resources available for downloading pages (but unlimited space for storing copies of Web pages), the objective of user-centric crawling is to schedule page downloading in such a way as to maximize repository quality.

Suppose that, due to resource limitations, it is only possible to download and (re)index up to B pages per time unit. (As in [12, 21], we assume uniform resource cost across all pages, since the fixed overhead of the operations required typically constitutes the dominant factor.) With user-centric crawling, page downloading is scheduled on the

basis of priorities. Each page p is assigned a numeric priority $P(p,t)$ proportional to the expected improvement in repository quality if p is (re)downloaded into the repository at time t. Page priorities may change with time. At the beginning of each time unit, the B pages of highest current priority are scheduled for downloading, along with the operations necessary to maintain the index up to date.

Ideally, we would set $P(p,t) = \Delta Q(p,t)$. However, since it is generally far from feasible to determine the precise value of this expression, we substitute the best available estimate of the expected change in repository quality due to (re)downloading page p. Stated formally, we set $P(p,t) = E(\Delta Q(p,t))$, where $E()$ denotes our estimation procedure. For pages that are already present in the repository, and hence have been downloaded at least once in the past, the expected benefit in terms of repository quality of downloading the page again in the future can be estimated using information observed during previous downloads of the same page. For such pages we propose to estimate $\Delta Q(p,t)$ for present or future values of t based on the value of $\Delta Q(p,t')$ measured at one or more times t' at which page p was downloaded in the past ($t' < t$).

We expect that for many (although not all) Web pages, a reasonable ballpark estimate of ΔQ can be made based on past measurements of the same quantity. We provide empirical evidence to justify this expectation later in Section 6; here we give some intuition. Many Web pages contain a set of "identifying" terms, which constitute the keywords used by most people to locate the page. These terms usually persist over a long period of time, even if other aspects of the page are volatile. For example, consider a page listing open access hours, rules and regulations, etc. for the Carnegie Mellon gymnasium. The terms "Carnegie Mellon gymnasium" are likely to remain on the page at all times, allowing people to locate it consistently by querying for these terms. Other terms such as "1:30pm" may come and go, but these changes are unlikely to impact user searching significantly. For such a page, ΔQ would consistently be small. For some pages, however, the volatile content plays a large role in user search. Consider, for example, a page listing weekly seminar announcements. Assuming the seminar topics generally align with the interests of many search engine users, ΔQ would tend to be fairly large, consistently. Of course there are bound to be Web pages that do not behave in any consistent way with respect to our metric, as is the case with any metric of content evolution.

As for newly-discovered pages that have not been downloaded even once, we leave the problem of how to estimate ΔQ as future work. Observe that given a suitable method of doing so, downloading of new pages can be scheduled jointly with redownloading of old ones, with resource allocation between these two tasks taken care of automatically.

In the remainder of this paper we focus on the important subproblem of how to prioritize redownloading, or *refreshing* of Web pages already present in a search engine repository. The main challenge is how to estimate the change in repository quality each time a page is downloaded, without incurring substantial additional overhead. We address this issue next.

5. ESTIMATING CHANGES IN QUALITY DURING CRAWLER OPERATION

Our approach to page refresh scheduling hinges on the ability to measure the change in repository quality, $\Delta Q(p,t)$, each time page p is downloaded. Clearly, a highly efficient method of measuring this quantity is needed. We focus on measuring the numerator of our expression for $\Delta Q(p,t)$ (Equation 6), since the denominator is the same across all pages and does not affect relative differences in priorities. Hence our goal is to measure the *absolute change in quality*, $\Delta Q_A(p,t)$, defined as:

$$\Delta Q_A(p,t) = \sum_{q \in \mathcal{Q}} f_q \cdot \sum_{p' \in \mathcal{W}} \Delta I(p', q, \mathcal{W}_t^L, \mathcal{W}_t^{L+p}) \cdot S(\mathcal{W}_t[p'], q)$$

where $\mathcal{W}_t^L$ denotes the contents of the search engine repository before page p is refreshed, and $\mathcal{W}_t^{L+p}$ denotes its contents afterward. From Equation 7, $\Delta I(p,q,\mathcal{W}_1,\mathcal{W}_2) = I(R(p,A(q,\mathcal{W}_2))) - I(R(p,A(q,\mathcal{W}_1)))$.

The anticipated workload $\mathcal{Q}$ can be estimated using recent query logs, and the function $I(r)$ can be determined from usage logs. For the remainder of this paper we assume $I(r) = c \cdot r^{-3/2}$, following [11]. Furthermore in the remainder of this paper we restrict the scoring function $S()$ to be one in which the score of a page depends on the content of that page only. (The score may also incorporate a global notion of "importance," e.g., PageRank [17], that is recomputed on occasion, at a time-scale that is large relative to the rate of downloading pages.) We also assume the score of a page is zero if it does not contain at least one instance of every term in a query.

Even if we sidestep the difficulty that true scores $S(\mathcal{W}_t[p'], q)$ of pages $p' \neq p$ are unavailable, say by substituting estimates, it is still very expensive to compute $\Delta Q_A(p,t)$ directly. Doing so requires materializing the result list of every query affected by the change in content of page p, and for each list examining the scores and ranks of every page whose rank has changed. Therefore we seek an efficient approximation scheme.

5.1 Approximation Scheme

5.1.1 Approximating the Workload

Since most search engine queries consist of only one or two terms, we approximate the query workload by breaking each multiple-term query into a set of single-term queries. (We leave more sophisticated treatment of multiple-term queries as future work.) The resulting simplified workload, $\mathcal{Q}'$, consists of only single-term queries and their frequencies, where the frequency $f_{q'}$ of a single-term query $q' \in \mathcal{Q}'$ is set equal to the sum of the frequencies of the queries in $\mathcal{Q}$ in which q' occurs. Now, observe that for any single-term query q consisting of a term that occurs in neither $\mathcal{W}_t^L[p]$ nor $\mathcal{W}_t^{L+p}[p]$, $S(\mathcal{W}_t^{L+p}[p], q) = S(\mathcal{W}_t^L[p], q) = 0$ so the result of q remains unchanged by the update to page p. Hence we arrive at the following approximate expression for $\Delta Q_A(p,t)$:

$$\Delta Q_A(p,t) \approx \sum_{q \in \mathcal{S}} f_q \cdot \sum_{p' \in \mathcal{W}} \Delta I(p', q, \mathcal{W}_t^L, \mathcal{W}_t^{L+p}) \cdot S(\mathcal{W}_t[p'], q)$$

where $\mathcal{S} = \mathcal{Q}' \cap (\mathcal{W}_t^L[p] \cup \mathcal{W}_t^{L+p}[p])$.

5.1.2 Approximating the Score-Rank Correspondence

To avoid computing result lists directly, we use precomputed functions, each of which provides an approximate mapping between score and rank among the results of a particular query. In particular, for each query $q \in \mathcal{Q}'$ we maintain an invertible piecewise linear function F_q from result scores to ranks, on log-log scale. The function pieces are of equal length on the log scale, so that scores corresponding to small rank values are approximated most accurately. Since they are only intended to provide an approximate mapping between score and rank, these functions need only be updated periodically, and can be made very small so as to fit in main memory (in our experiments described in Section 6, we used three pieces per function; the space requirement is just 20 bytes per query).

Let $F_q(s)$ denote the result of using function F_q to estimate the rank in the result of query q of a page whose score is s. Conversely let $F_q^{-1}(r)$ denote the result of using the inverse of F_q to estimate the score of a page appearing at rank position r in the result of query q. Using our technique of approximating the relationship between score and rank for a particular query using a piecewise function, we estimate $\Delta Q_A(p,t)$ as follows.

At the time page p is refreshed, suppose we are able to determine the set $\mathcal{S}$ of queries affected by the changes in p, as well as for each $q \in \mathcal{S}$ the scores for p both before and after the refresh is applied, i.e., $S(\mathcal{W}_t^L[p], q)$ and $S(\mathcal{W}_t^{L+p}, q)$. (We describe how to obtain these quantities efficiently later in Section 5.2.) For notational ease let $s_1 = S(\mathcal{W}_t^L[p], q)$ and $s_2 = S(\mathcal{W}_t^{L+p}[p], q)$. For each query $q \in \mathcal{S}$ we estimate $R(p, A(q, \mathcal{W}_t^L))$ and $R(p, A(q, \mathcal{W}_t^{L+p}))$ as $r_1 = F_q(s_1)$ and $r_2 = F_q(s_2)$, respectively. Our expression for the component of $\Delta Q_A(p,t)$ corresponding to query q becomes:

$$\begin{aligned}&\sum_{p' \in \mathcal{W}} \Delta I(p', q, \mathcal{W}_t^L, \mathcal{W}_t^{L+p}) \cdot S(\mathcal{W}_t[p'], q) \\ &\approx \big((I(r_2) - I(r_1)) \cdot s_2\big) + \\ &\quad \sum_{p' \in \mathcal{W}, p' \neq p} \Delta I(p', q, \mathcal{W}_t^L, \mathcal{W}_t^{L+p}) \cdot S(\mathcal{W}_t[p'], q)\end{aligned}$$

Now we focus on transforming the second term into a form that is amenable to efficient evaluation. Assume $r_1 < r_2$ (the case in which $r_1 > r_2$ is symmetric). We transform the summation over pages into a summation over rank positions affected by the shift in rank of page p, and invoke our piecewise function to obtain a ballpark estimate of true scores:

$$\begin{aligned}&\sum_{p' \in \mathcal{W}, p' \neq p} \Delta I(p', q, \mathcal{W}_t^L, \mathcal{W}_t^{L+p}) \cdot S(\mathcal{W}_t[p'], q) \\ &\qquad \approx \sum_{r=r_1+1}^{r_2} \big(I(r-1) - I(r)\big) \cdot F_q^{-1}(r)\end{aligned}$$

Assume now that r_1 and r_2 fall into the same piece P of piecewise function F_q (it is straightforward to extend our method to handle the case in which r_1 and r_2 span multiple pieces). Let δ_i denote the average difference between the scores for two consecutive rank positions in piece P. Simplifying the above expression, we obtain:

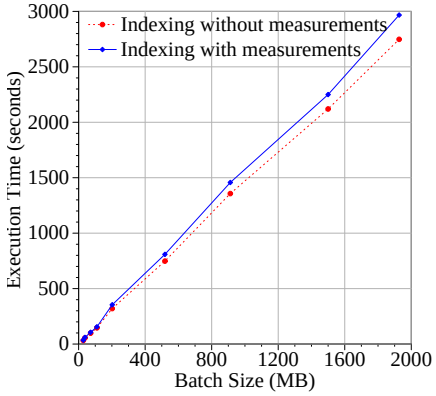

Figure 2: Overhead of our measurement scheme.

$$\big(I(r_1) \cdot F_q^{-1}(r_1 + 1)\big) - \sum_{r=r_1+1}^{r_2-1} \big(I(r) \cdot \delta_i\big) - \big(I(r_2) \cdot F_q^{-1}(r_2)\big)$$

For cases in which $(r_2 - r_1)$ is small we evaluate the above expression exactly, using $I(r) = c \cdot r^{-3/2}$. When $(r_2 - r_1)$ is large we use an approximate form of the middle term derived by substituting a definite integral in place of the summation. A closed-form solution for the integral is easy to obtain. The net result of applying the integral approximation is:

$$\sum_{k=i}^{j} k^{-3/2} \approx 2 \cdot \left(\frac{1}{\sqrt{i}} - \frac{1}{\sqrt{j-1}}\right)$$

We found our experimental results (Section 6) not to be very sensitive to the settings of our approximation parameters, such as the number of pieces to use in each F_q.

5.2 Taking Measurements During Index Maintenance

Our scheme for estimating the change in repository quality upon refreshing page p described in Section 5.1 takes as input the set $\mathcal{S} \subseteq \mathcal{Q}'$ of single-term queries (constructed from the original multiple-term query log) affected by the changes in p, and for each $q \in \mathcal{S}$ the scores for p both before and after the refresh is applied, i.e., $S(\mathcal{W}_t^L[p], q)$ and $S(\mathcal{W}_t^{L+p}, q)$. Conveniently, it is possible to compute these scores efficiently by coupling the measurement procedure closely with the process of updating the inverted index, which is a necessary operation that makes newly-downloaded content "searchable."

An inverted index contains lists of *postings* extracted from the repository. A posting corresponds to a unique term/page pair, and typically contains the number of times the term appears in the page, font sizes, and any other information required to evaluate the scoring function. Postings are typically updated in batches, after a set of pages have been (re)downloaded into the repository. During the index updating process, postings corresponding to terms no longer present in pages are removed, and new postings are added corresponding to new terms. With our measurement technique, whenever a posting corresponding to term $\mathcal{T}$ in page p is added or removed, the resulting shift (if any) in the score of p for query $q = \{\mathcal{T}\}$ is recorded in a special in-memory buffer. After processing of the batch of updates has completed, ΔQ_A estimates are computed using the procedure of Section 5.1.

5.3 Overhead of Measurement Scheme

We integrated our quality change measurement scheme with the indexing component of Lucene [2], a publicly-available document indexing and retrieval system. Figure 2 shows the time it takes to index a batch of HTML pages, both with and without our special measurement code. Batch size (in megabytes) is plotted on the x-axis. Total running time is plotted on the y-axis. Our measurement scheme incurs very modest overhead of $7 - 8\%$.

6. EXPERIMENTS

We compared our user-centric page refreshing scheme with other schemes proposed in the literature, using simulations over real Web evolution data. We used two different data sets (both from the UCLA WebArchive project data[4, 16]):

1. **B**oston **D**ata **Set** (BDS): A 48-week archive of a single Web site, www.boston.com. The complete Web site was crawled once every week. Since our focus is on refreshing the pages that persist over an extended period of time, pages not present in all 48 weekly snapshots were removed. The remaining Web pages number around $16,000$.

2. **M**ultiple site **D**ata **S**et (MDS): A 48-week archive of 15 different Web sites, each sampled from a different OpenDirectory topic area [3]. As with BDS, pages not present in every weekly snapshot were removed. Furthermore, in order to emphasize the role played by Web page refreshing in the relatively short duration of the Web evolution data we had access to, and also to reduce the time required to perform each run of our experiments, we only retained pages that changed in some way (as determined by a checksum) at least once during the 48-week period. The final data set consists of around $19,000$ pages.

To obtain query workloads for our experiments we used the publicly-available AltaVista query log [1]. It consists of around seven million single- and multi-term queries. Since our data sets are concentrated around fairly specific topics, whereas the topics represented in the query log are quite broad, we created workloads specific to each data set by filtering queries based on relevance to the pages in each data set. In particular, we eliminated queries for which the sum of TF-IDF scores across all pages in a data set was below a certain threshold. The threshold was chosen based on observing a knee in the distribution that we felt would serve as a natural cutoff point for query relevance.

Next we describe each of the three page refreshing strategies we evaluated in turn.

6.1 Web Page Refreshing Schemes Evaluated

6.1.1 Staleness-Based Refreshing

With staleness-based refreshing (*SBR*) [9], the objective is to minimize the number of *stale* pages in the search engine repository.[3] It is shown in [9] that under the staleness objective, when resources are limited it is best to abandon refreshing of frequently updated pages in favor of refreshing of other, less frequently updated pages.

In the simplest implementation of *SBR*, the repository copy of a page is considered stale if it is not identical to the current Web copy. Since Web pages are often updated in fairly minor ways (e.g., advertisements, timestamps) we used the standard method of *shingling* [5, 7] as a heuristic for discriminating between significant and insignificant updates. A page is considered stale if the fraction of shingles that differ between the repository copy and Web copy of the page exceeds a particular threshold $\tau_{SBR} \in [0, 1]$. In our experiments we tested values of τ_{SBR} throughout the range $[0, 1]$.

The work in [9] focuses uniquely on determining with which frequency to refresh each page. No algorithm is provided for scheduling refreshes in the presence of a hard resource constraint. We used the transportation algorithm suggested in [21] for this purpose.

6.1.2 Embarrassment-Based Refreshing

With embarrassment-based refreshing (*EBR*) [21], the objective is to minimize the level of "embarrassment" to a search engine provider. Embarrassment accrues whenever a user clicks on a search result link, only to discover that the destination page is not, in fact, relevant to the query she had issued. (A Boolean notion of relevance is assumed.)

The work in [21] applies to a wide variety of page update models, including the fairly general *quasi-deterministic* model. (In the quasi-deterministic model, time is divided into discrete slices, and the probability that a particular page undergoes an update may be different at each time-slice.) We did not feel that our 48-week data set contained a sufficient duration of data to fit a reliable quasi-deterministic model, so we used the simpler *Poisson* update model, as done in [9].

An important parameter in *EBR* is $d(p)$, which denotes the probability that if the repository copy of page p is out of date with respect to the current Web copy (i.e., $\mathcal{W}^L[p] \neq \mathcal{W}[p]$), whenever the search engine presents page p to a user, p turns out to be an irrelevant response for the query that was issued (note that $d(p)$ is a query-independent parameter). No method of estimating this parameter is provided in [21]. Since the shingling technique is a widely-accepted way of measuring the difference between two Web pages, or two copies of the same page, we apply it here. In particular, we assume that if a page undergoes an update, it becomes irrelevant to an average query if the fraction of shingles that change exceeds a configurable threshold τ_{EBR}. We compute $d(p)$ as the fraction of updates to page p that induce at least τ_{EBR} fraction of the shingles to change. In our experiments we tested values of τ_{EBR} throughout the range $[0, 1]$.

6.1.3 User-Centric Refreshing

Our user-centric page refreshing scheme is parameterized by a scoring function $S()$. While our approach is compatible with a wide variety of possible scoring functions, for our experiments we needed to use a specific scoring method. Since no standard exists, we used two well-accepted methods that we feel constitute two extremes among the spectrum of options: (1) the well-known **TF-IDF** metric [18], using the variant employed in the popular Lucene software [2], and (2) **inlink count** obtained by querying Google, which we used

[3] In [9] an alternative optimization objective, minimizing average *age*, is also proposed. Our preliminary experiments showed that age-based refreshing did not perform as well as staleness-based refreshing under our metric, so we did not consider it further.

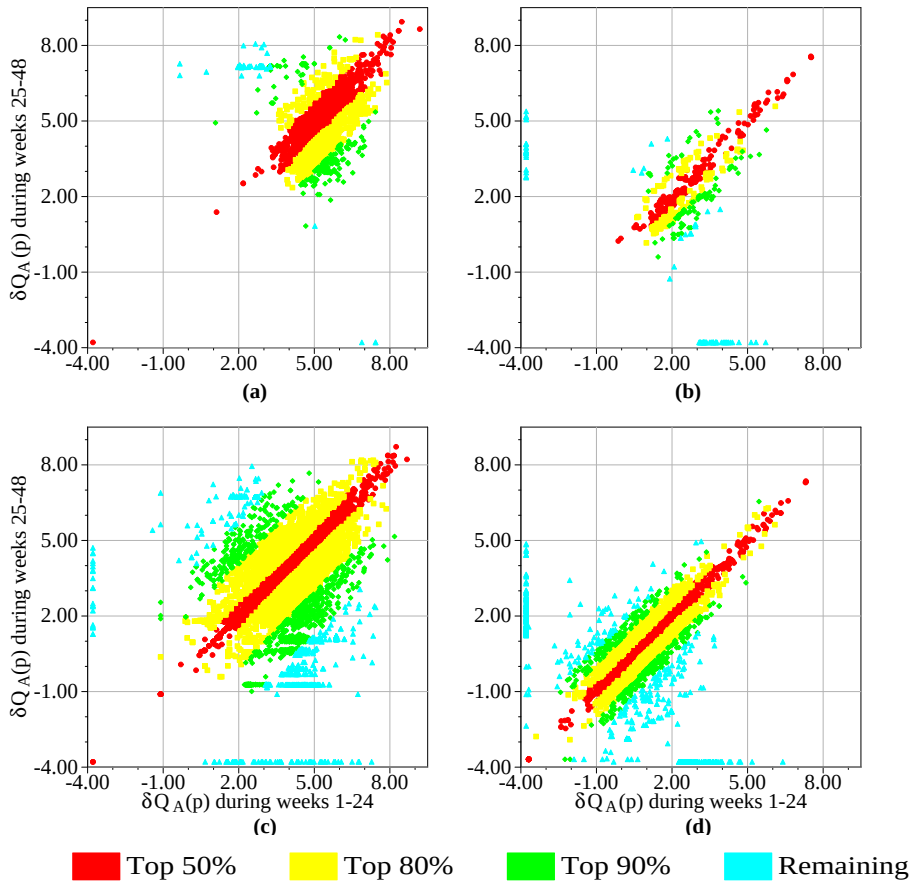

Figure 3: Amenability to forecasting of time-normalized change in quality ($\delta Q_A(p)$). The four graphs shown correspond to (a) BDS data set with TF-IDF scoring function, (b) BDS with inlink count scoring function, (c) MDS data set with TF-IDF, and (d) MDS with inlink count. All graphs are on a log-log scale.

as a surrogate for PageRank [17] due to lack of adequate data (it has been suggested that inlink count and PageRank yield similar results [20]). In both cases the result of a query consists of a list of all pages that contain every term in the query, arranged in descending order of score.

In our user-centric page refreshing scheme, each page p is assigned an associated priority value $P(p,t)$, which may vary over time. Page refreshing is scheduled according to priority. The priority of a page is set equal to the expected change in repository quality if that page is refreshed, as estimated by extrapolating from past measurements of this quantity taken during previous refreshes of the same page. These measurements are obtained using the estimation procedure of Section 5.

A variety of extrapolation methods can be used. The option we selected for our experiments is as follows. Given a set $\mathcal{R}(p)$[4] of time instants of past refreshes of page p, let:

$$\delta Q_A(p) = \frac{1}{|\mathcal{R}(p)|} \sum_{t \in \mathcal{R}(p)} \frac{\Delta Q_A(p,t)}{t - LR(p,t)}$$

where $LR(p,t)$ denotes the time of the most recent refresh of page p prior to t. Set $P(p,t) = \delta Q_A(p) \cdot (t - LR(p,t))$.

[4]We envision that in a real deployment the set $\mathcal{R}(p)$ would be determined based on a sliding window of recent refreshes of page p. (Other heuristics for favoring recent observations, such as exponentially-decayed averaging, warrant investigation as well; we leave this topic as future work.)

6.2 Estimation of Page Change Characteristics

Each of the page refreshing schemes we consider relies on forecasting of Web page change behavior based on behavior observed in the past. In particular, for each page p *SBR* requires a Poisson change rate parameter $\lambda(p)$, *EBR* requires a query irrelevance probability parameter $d(p)$, and user-centric refreshing requires a time-normalized quality change value $\delta Q_A(p)$. We opted against splitting our data sets to perform parameter fitting and evaluation over different portions (say, 24 weeks each), because shortening our somewhat short 48-week data any further would make it difficult to obtain reliable performance measurements. Plus, in this paper we do not focus on the forecasting problem, and we seek to compare all three methods on equal footing, independent of the forecasting method used. Therefore, for all three policies we used the entire 48-week data set to estimate the necessary parameter for each page p.[5]

Still, we wanted to check that quality change values $\delta Q_A(p)$ are amenable to forecasting based on past measurements. For this purpose we estimated $\delta Q_A(p)$ values (using our approximation method of Section 5.1) for each page, once over

[5]Note that for *SBR* and *EBR*, different settings for the shingles threshold τ_{SBR} (τ_{EBR}, respectively) result in potentially different $\lambda(p)$ ($d(p)$) values, which is precisely the purpose of varying the threshold.

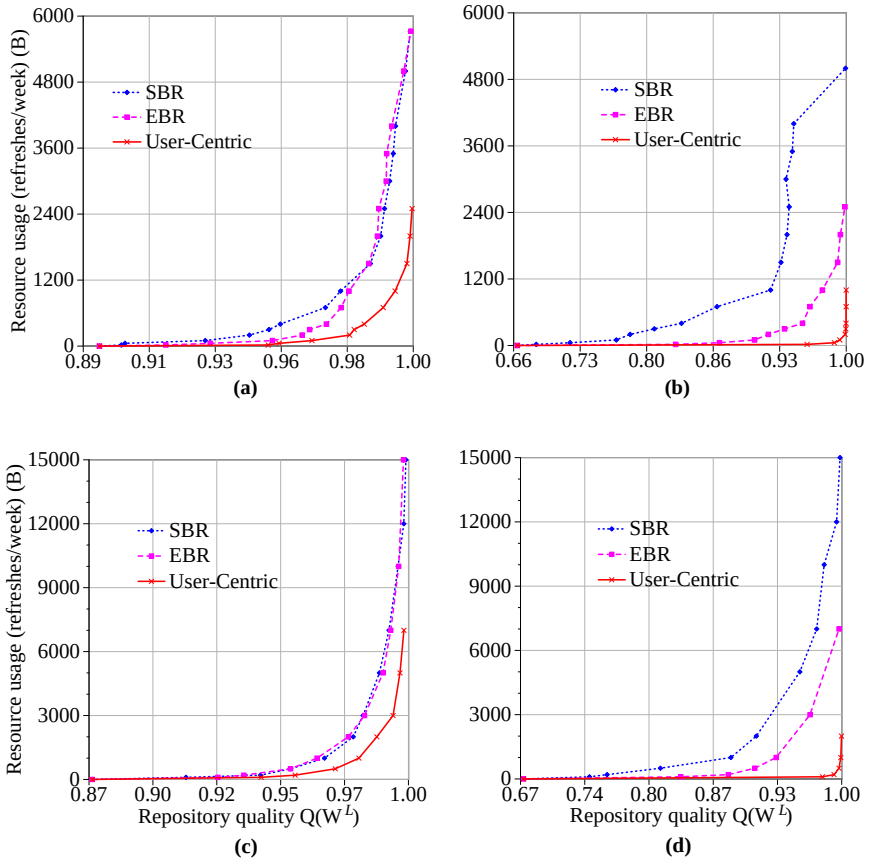

Figure 4: Repository quality versus resource usage. The different graphs are for (a) BDS data set with TF-IDF scoring function, (b) BDS with inlink count scoring function, (c) MDS data set with TF-IDF, and (d) MDS with inlink count.

the first 24 weeks of our data set, and again over the second 24 weeks, under the scenario in which every update to a page triggers an immediate refresh. We then compared the $\delta Q_A(p)$ estimates across the two 24-week periods. Figure 3 shows the outcome, for each of our two data sets under each of the two scoring functions we tested. In each graph, $\delta Q_A(p)$ over weeks $1-24$ is plotted on the x-axis, and $\delta Q_A(p)$ over weeks $25-48$ is plotted on the y-axis. Each dot in each graph corresponds to one page. When $\delta Q_A(p) = 0$, that indicates no change in repository quality due to updates to page p. Beyond that the scale of the axes is immaterial (since we are not measuring normalized quality). Each graph is plotted on a log-log scale, with pages with a value of 0 for one of the two $\delta Q_A(p)$ measurements inserted artificially along the edges. Pages with $\delta Q_A(p) = 0$ for weeks $1-24$ as well as weeks $25-48$ are not plotted (hence these graphs present a conservative view of amenability to forecasting). Dots are colored according to quantiles of proximity to the diagonal; see the key below the graphs. Points that are close to the diagonal ($y = x$ line) correspond to pages whose $\delta Q_A(p)$ values remain fairly consistent in both halves of the data set, implying that they can be forecasted accurately at this time-scale based on past measurements. These findings are in accord with those presented in [16], which assessed amenability to forecasting of Web page change characteristics as measured by TF-IDF cosine similarity directly.

6.3 Comparison of Page Refreshing Schemes

We compared the three page refreshing schemes (*SBR*, *EBR*, and user-centric crawling) using our user-centric repository quality metric which, as we have argued, we believe serves as a suitable metric for evaluating a crawler serving a search engine. Of course, crawling can also be used for other purposes (archival, mining, etc.), in which case our metric is not appropriate. For the purpose of evaluating the performance of a refreshing scheme we applied the precise formula for repository quality (Equation 5), and did not rely on any approximation techniques.

For this experiment we provided each refreshing scheme with a fully synchronized repository at week 1, and then allowed a fixed number of pages, B, to be refreshed every week for the remaining 47 weeks. We compared page refreshing schemes in terms of the resource requirement (B value) necessary to achieve a certain level of repository quality according to our user-centric metric, for two different scoring functions, TF-IDF and inlink count, over each of our two data sets, BDS and MDS. The results are plotted in Figure 4. In each graph, repository quality is plotted on the x-axis, and the resource requirement B is plotted on the y-axis. For each of *SBR* and *EBR*, for each B value the best repository quality level obtained using shingle threshold values $\tau \in \{0.1, 0.2, \ldots, 0.9, 1.0\}$ is plotted. For both data sets and both scoring functions, our user-centric page refresh-

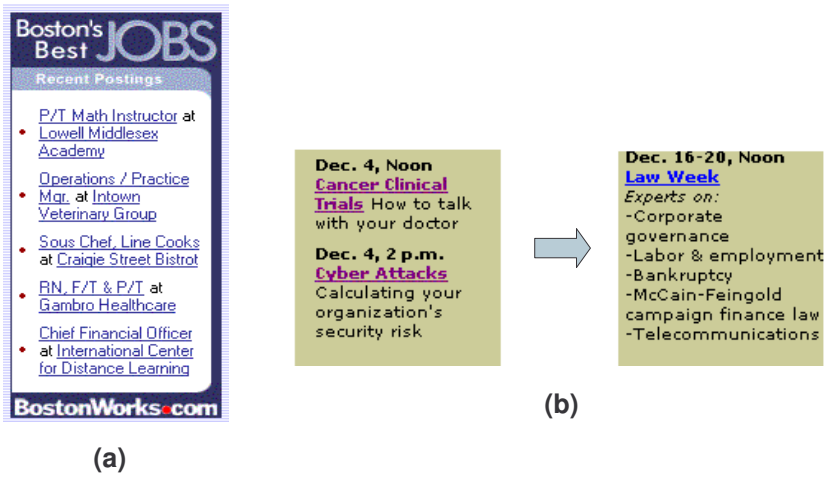

Figure 5: Examples drawn from our real-world data sets (boston.com and washingtonpost.com, respectively).

ing scheme requires substantially fewer resources to achieve the same level of repository quality than either of *SBR* and *EBR*.

We highlight the primary underlying reasons for this result using the following two examples taken from our data sets:

Example 1: Figure 5(a) shows an advertisement added to a Web page in the boston.com data set. As it turned out, although the new advertisement consists of a large textual segment, none of the terms in the advertisement match frequently-issued queries in the AltaVista query workload. Hence, from the perspective of our user-centric notion of repository quality it is not important to capture the content of the advertisement. Consequently our user-centric refreshing scheme did not devote resources to refreshing this page (which turned out not to be updated in any way other than changing of advertising material), leaving more resources available for other tasks. This example illustrates that heuristics for estimating the importance of an update based on the number of words that change do not always work well.

Example 2: Figure 5(b) shows a portion of a Web page containing seminar announcements, that was updated to remove outdated announcements and replace them with a new announcement of an upcoming law seminar series. If this page is not refreshed in a timely fashion, users querying for, say "Washington campaign finance", would not see this page among the query results even though it should appear (and be placed at a good rank position under at least some scoring functions). Our user-centric repository quality metric is particularly good at characterizing the importance of keeping this page up to date in the repository, by noting the high degree of match between frequent queries and evolving content (for example, the query "cancer" occurs frequently in the AltaVista query workload). This example illustrates (1) the importance of accounting for false negatives as well as false positives, and (2) that certain frequently-updated pages merit the devotion of precious refreshing resources, if it is the case that the updates tend to have a large impact on the user experience.

One may be inclined to suppose that, say, 95% repository quality is sufficient, and that there is no need to shoot for quality values very close to 100%. However, the difference between 95% and 99% repository quality can have a significant impact on the user experience. In fact, we came across Example 2 by examining a scenario in which *SBR* and *EBR* each achieved $\sim$ 95% quality, whereas our user-centric scheme attained over 99% quality under the same resource constraint. Both *SBR* and *EBR* neglected to refresh this important seminar announcement page, leading to a substantial degradation in the quality of search results for a large number of (simulated) users.

7. SUMMARY AND FUTURE WORK

The capability to query the content of the World Wide Web instantaneously and accurately using search engines is invaluable, so it is very important that we understand how to deploy highly effective Web crawlers. Given the sustained growth in size and dynamicity of the Web as a whole, it appears that Web crawling will remain, in relative terms, a resource-starved activity for the foreseeable future. This property is especially true for the growing number of topic-specific search engines, which are often sustained by modest budgets.

In this paper we introduced a new Web crawling paradigm designed specifically for search engines, in which the objective is to allocate resources to crawling tasks in such a way as to maximize the quality of the user experience, given a fixed resource allowance. Scheduling of crawler tasks is driven entirely by usage, in terms of which queries are issued, with what frequency, and which results are inspected by users, so our scheme does not rely on external tuning parameters.

After introducing and formalizing our overall user-centric crawler scheduling policy we focused on the important subproblem of scheduling refreshing of Web pages already present in a local search repository, in order to keep them up to date. We showed that the benefit of refreshing a particular page, measured in terms of impact on the user experience, is amenable to prediction based on measurements of the benefit of downloading the page in the past. We devised an efficient, yet approximate method for taking these measurements that is tightly integrated with the process of updating an inverted index maintained over the repository, and incurs little additional overhead. Lastly we compared our user-centric page refreshing scheme against prior schemes empirically using real Web data. Our results demonstrate that our scheme requires substantially fewer resources to achieve the same user experience quality, leaving more resources for other important tasks such as downloading new pages.

7.1 Future Work

Our user-centric page refreshing scheme can be extended to make it compatible with scoring functions in which the score of a page depends partially on the content of other pages (as with anchortext inclusion methods [6]). In principle such an extension can be made without compromising the crucial ability to estimate changes in repository quality in tandem with index maintenance operations (Section 5.2). Evaluating the viability of extending our techniques in this way is an important topic of future work. Another problem left as future work is to determine the most effective method of forecasting the change in quality due to refreshing a page, based on historical observations. Finally, the most significant topic of future work is to devise methods of gauging the benefit of downloading newly-discovered pages, to enable downloading of new pages to be scheduled jointly with refreshing of old ones (as done in [12] under a different optimization objective).

8. ACKNOWLEDGMENTS

We are grateful to Junghoo Cho and Alexandros Ntoulas for providing access to data from the UCLA WebArchive project.

9. REFERENCES

[1] AltaVista Query Log. `http://ftp.archive.org/AVLogs/`.

[2] Jakarta Lucene. `http://jakarta.apache.org/lucene/docs/index.html`.

[3] Open Directory Project. `http://www.dmoz.org/`.

[4] UCLA WebArchive. `http://webarchive.cs.ucla.edu/`.

[5] S. Brin, J. Davis, and H. Garcia-Molina. Copy Detection Mechanisms for Digital Documents. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, 1995.

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

[7] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic Clustering of the Web. In *Proceedings of the Sixth International World Wide Web Conference*, 1997.

[8] S. Chakrabarti, M. van den Berg, and B. Dom. Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery. In *Proceedings of the Eighth International World Wide Web Conference*, 1999.

[9] J. Cho and H. Garcia-Molina. Synchronizing a Database to Improve Freshness. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, 2000.

[10] J. Cho and H. Garcia-Molina. The Evolution of the Web and Implications for an Incremental Crawler. In *Proceedings of the Twenty-sixth International Conference on Very Large Databases*, 2000.

[11] J. Cho and S. Roy. Impact of Search Engines on Page Popularity. In *Proceedings of the Thirteenth International World Wide Web Conference*, 2004.

[12] J. Edwards, K. S. McCurley, and J. A. Tomlin. An Adaptive Model for Optimizing Performance of an Incremental Web Crawler. In *Proceedings of the Tenth International World Wide Web Conference*, 2001.

[13] T. Joachims. Optimizing Search Engines Using Clickthrough Data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002.

[14] J. M. Kleinberg. Authoritative Sources in a Hyperlinked Environment. *Journal of the ACM*, 46(5):604–632, 1999.

[15] R. Lempel and S. Moran. Predictive Caching and Prefetching of Query Results in Search Engines. In *Proceedings of the Twelfth International World Wide Web Conference*, 2003.

[16] A. Ntoulas, J. Cho, and C. Olston. What's New on the Web? The Evolution of the Web from a Search Engine Perspective. In *Proceedings of the Thirteenth International World Wide Web Conference*, 2004.

[17] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. *Stanford Digital Library Project Technical Report*, 1998.

[18] G. Salton and C. S. Yang. On the Specification of Term Values in Automatic Indexing. *Documentation*, 29(351–372), 1973.

[19] D. Sullivan. Searches Per Day, SearchEngineWatch. `http://searchenginewatch.com/reports/article.php/2156461`.

[20] T. Upstill, N. Craswell, and D. Hawking. Predicting Fame and Fortune: PageRank or Indegree? In *Proceedings of the Eighth Australasian Document Computing Symposium*, 2003.

[21] J. L. Wolf, M. S. Squillante, P. S. Yu, J. Sethuraman, and L. Ozsen. Optimal Crawling Strategies for Web Search Engines. In *Proceedings of the Eleventh International World Wide Web Conference*, 2002.