Crawl Ordering by Search Impact

Sandeep Pandey*
Carnegie Mellon University
spandey@cs.cmu.edu

Christopher Olston
Yahoo! Research
olston@yahoo-inc.com

ABSTRACT

We study how to prioritize the fetching of new pages under the objective of maximizing the quality of search results. In particular, our objective is to fetch new pages that have the most *impact*, where the impact of a page is equal to the number of times the page appears in the top K search results for queries, for some constant K, e.g., $K = 10$. Since the impact of a page depends on its relevance score for queries, which in turn depends on the page content, the main difficulty lies in estimating the impact of the page before actually fetching it. Hence, impact must be estimated based on the limited information that is available prior to fetching page content, e.g., the URL string, number of in-links, referring anchortext.

We formally characterize this problem and study its hardness. We leverage our formalism to design a new *impact-driven* crawling policy, and demonstrate its effectiveness using real world data. Our technique ensures that the crawler acquires content relevant to "tail topics" that are obscure but of interest to some users, rather than just redundantly accumulating content on popular topics.

Categories and Subject Descriptors

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

General Terms

Algorithms, Performance, Design, Experimentation

Keywords

web crawling, crawl ordering, impact-driven crawling

1. INTRODUCTION

The main task of a Web crawler is to fetch pages from among the "frontier" of discovered URLs (Figure 1), in order to supply more pages to the ranking engine and in turn improve result quality. (There are other reasons to fetch new pages, such as discovery of additional pages or improving the accuracy of link-based importance measurements, but these aspects are not the focus of this paper.) For a Web-scale crawler, the frontier is gigantic (typically many times the number of crawled pages [11]) and many, if not most, of these pages are of no interest to search engine users for reasons related to quality, spam and relevance.

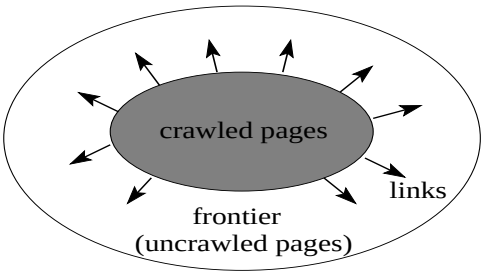


Figure 1: A search engine crawler's view of the Web.

	Query: "NCAA Football"		"Judy Inklesmurf"	
Rank	Page	Query-independent score	Page	Query-independent score
1	P_1	0.82	P_2	0.40
2	P_3	0.95	P_4	0.12
.	.	.	.	.
.	.	.	.	.
100	P_{201}	0.80	P_{202}	0.01

Table 1: Top results for queries "NCAA Football" and "Judy Inklesmurf."

On the other hand, certain frontier pages are highly relevant to topics of present interest, and users expect to see them in query results right away. Given that a crawler has limited resources relative to the enormity of the uncrawled frontier, it is crucial to fetch a well-chosen subset in a well-chosen order.

The current state of the art is to prioritize page fetching by query-independent features such as link-based importance (e.g., PageRank, or an estimate thereof) [1, 7]. Unfortunately, query-independent importance measures do not provide the best prioritization policy for a search engine crawler. Consider the two queries "NCAA Football" and "Judy Inklesmurf" (a hypothetical actor who stars in an obscure television series with a growing cult following). There

*This work was performed while the author was visiting Yahoo! Research.

WSDM'08, February 11–12, 2008, Palo Alto, California, USA.

are plenty of high-quality pages on the Web for the query "NCAA Football," but very few about "Judy Inklesmurf." For the sake of this example, say the pages relevant to these queries on the Web are as shown in Table 1, when arranged in decreasing order of rank position in the query result list. (Observe that the pages are not ranked in strictly decreasing order of query-independent importance scores because a search engine scoring function considers many other features such as referring anchortext, page content, URL string [16].)

Now suppose that we are given the choice between fetching page P_{201} about NCAA Football or fetching page P_2, the most linked-to page about Judy Inklesmurf. Though page P_{201} has a higher importance score than P_2, we prefer to fetch page P_2. Fetching page P_2 would make a huge improvement in the search results quality for the query Judy Inklesmurf. On the other hand, fetching page P_{201} would not lead to any significant improvement because there are many other superior quality pages for the NCAA football query (we assume P_{201} has no substantial impact for other queries).

The problem with using a query-independent importance measure to do crawl prioritization is that it only accumulates content on well-established topics whose pages have many links. However, as mentioned in [17], the number of tail queries (i.e., queries that lie in the tail of the query frequency distribution) seen by search engines today is too large to ignore.

In this paper we propose a new *impact-driven* approach whereby we focus directly on what topics the search engine users are interested in, and on how much *impact* a page would have on the search engine's ability to serve those interests. The impact of fetching a page depends on the following factors: (a) the queries for which the page is relevant and how often those queries are issued by users (b) the ranks that the page would receive in the result lists of those queries, and (c) the attention paid by users to the results displayed at those ranks. (We give a formal definition of impact in Section 3.)

In the remainder of this section, we measure the (lack of) correlation between query-independent importance score and impact (Section 1.1), and then give an overview of our impact-driven approach (Section 1.2).

1.1 Weak Correlation Between Impact and Query-Independent Importance

On the real Web, it is not the case that ordering pages by query-independent importance is similar to ordering them by search impact. We support this claim using real data. Figure 2 shows the query-independent importance scores and impact values of a random sample of pages.[1] Each point in the figure denotes a page. We divided the points with similar query-independent importance scores into buckets and computed the average impact for each bucket, as shown by the wide bars in Figure 2. (The two left-most bars are not very noticeable due to the small average impact values.)

As expected, as the average query-independent importance score of a bucket increases, the average impact increases, but observe that the points in the graph are significantly scattered—the Pearson correlation coefficient is only 0.09. In other words, there are a significant number of pages that have low query-independent importance scores but high impact, and vice versa.

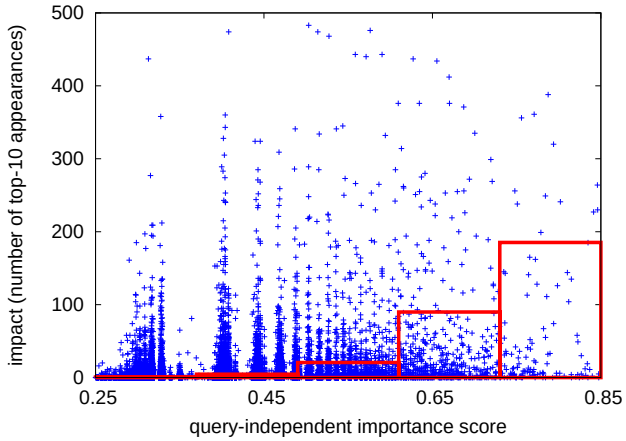


Figure 2: Impact and query-independent importance scores of a random sample of pages.

For example, the URL `www.silverscape.com/index.cfm/Strategy/Product_Positioning` is in the bottom 20% of all pages based on the query-independent importance score, but it falls in the top 1% in terms of impact. The reason is that many users issue the query "product positioning" on the search engine and this page is ranked among the top 10 results for this query due to its matching URL, referring anchortext, and other attributes, despite its weak importance score. (Additional real-world examples are given in Section 6.2.)

Conversely, the URL `www.pc2sms.eu` is in the top 1% based on the query-independent importance score, but below the top 25% based on impact. The reason for the relatively low impact is that the page is only relevant to queries such as "send free SMS," which is a popular topic on the Web and has a large number of relevant pages with higher scores.

1.2 Our Approach

As stated above, our goal is to fetch uncrawled pages that have the most impact. Measuring a page's impact requires computing its rank in the result list of a query, as determined by the search engine's scoring function.

Typically a scoring function takes many features about a page as its input, e.g., page content, URL string, referring anchortext [16]. We can divide the set of features into two groups: *content-dependent* (e.g., page title, words on the page) and *content-independent* (e.g., URL string, inlinks, host affiliation, referring anchortext). For an uncrawled page, the crawler has access to the content-independent features of the page only. Hence, the challenge is in estimating its rank for queries, while only knowing a subset of its scoring features (in particular the content-independent ones). Fortunately, content-independent features (especially inlinks/PageRank and referring anchortext) tend to be heavily weighted in the overall scoring procedure [2], so there is hope of being able to do this estimation reasonably well.

Our basic approach is the following: given scoring function

[1]More precisely, these are the uncrawled pages of Dataset 1 described in Section 6.1. For the purpose of this experiment the query-independent importance score was computed as done by a major search engine. Impact was measured as the number of times a page appears among the top 10 search results of queries submitted to the the search engine during a five-day period.

$S(p, q)$ over page p and query q, we define a new scoring function $S'(p, q)$ which takes content-independent features of page p as input and outputs a probability distribution over $S(p, q)$. Then we estimate the impact of each uncrawled page on a representative query workload. (The workload may be constructed from past queries that we expect to see again, and perhaps also anticipated future queries forecasted from news, blogs, or other early indicators of hot topics.)

There are two important refinements to the basic approach. The first is that, due to the fact that it is impossible to predict the future workload with full accuracy, we must supplement our workload-based calculations with query-independent cues. Second, to avoid per-query state we focus on a small subset of the query workload when driving the crawler. In particular, we focus on queries whose results are likely to be improved by crawling new pages. (Details on how we identify and exploit such queries are given in the body of the paper.)

1.3 Contributions

The main contributions of this paper are:

- We formulate the problem of ordering pages to crawl based on the impact on search result quality (Section 3).
- We give the computational complexity of the problem (Section 4).
- We propose a practical impact-driven crawling policy (Section 5).
- We demonstrate the effectiveness of our crawling policy via experiments over real world data (Section 6).

Before proceeding, we discuss related work.

2. RELATED WORK

Web crawling is a well-studied problem. The crawling problem has three main aspects: (1) *discovery* of new URLs, typically by monitoring pages that link to new pages (2) *acquisition* of the content associated with a subset of the discovered URLs, and (3) periodic *synchronization* of previously acquired pages to maintain freshness. This paper focuses on acquisition; the discovery and synchronization aspects are largely orthogonal and have been studied elsewhere, e.g. [6, 9, 10, 15, 20]. We leave the problem of dividing crawling resources among these three tasks as future work.

Prior work on choosing the order in which to acquire new content (also known as "prioritizing the crawling frontier") focused on ordering pages according to a query-independent notion of page importance [1, 7, 14]. In this paper we argue that the crawl ordering problem should instead be viewed through the lens of queries. Viewed in this manner, pages that would receive a good rank position for users' queries should be crawled, even if they have a relatively low page importance score (e.g. because they pertain to an obscure "tail topic," or are new and have not yet accumulated many in-links [8]).

Our query-centric approach is reminiscent of *focused crawling* [5]. Focused crawling scours the Web in search of pages relevant to a particular topic (or small set of topics). In contrast, our approach biases a "full-Web" crawler toward picking up pages that match any topic for which the search engine currently does not have enough relevant, high-quality content. Also, our optimization objective (maximizing *impact*) differs from the one studied in focused crawling (maximizing the *harvest rate*); optimizing for impact requires techniques that are driven by search result ranking, rather than topic classification as used in focused crawling.

3. PROBLEM STATEMENT

We give a formal definition of impact (Section 3.1), and use this definition to formalize the problem of impact-driven crawl ordering (Section 3.2).

3.1 Impact Definition

Let $S(p, q)$ denote the search engine scoring function, where p is a page and q is a query. Let $R(p, q)$ denote the rank of page p in the ranked result list of query q, as computed using $S(p, q)$ over all crawled and uncrawled pages. We define the impact of page p with respect to query q as:

$$I(p, q) = V(R(p, q))$$

where $V(r)$ denotes the *visibility* of rank r in the result list of a query. Formally, the visibility of rank r is the probability of an average user to view a page when displayed at rank r in a result list. Since users mostly pay attention to the top-ranked pages [13], we expect V to be larger for smaller ranks (i.e., ranks closer to 1).

Given a query workload $\mathcal{Q}$ consisting of queries and their associated frequencies, we define the total impact of page p as:

$$\begin{aligned} I(p, \mathcal{Q}) &= \sum_{q \in \mathcal{Q}} f(q) \cdot I(p, q) \\ &= \sum_{q \in \mathcal{Q}} f(q) \cdot V(R(p, q)) \end{aligned}$$

where $f(q)$ is the frequency of query q in workload $\mathcal{Q}$.

3.2 Crawl Ordering Optimization Problem

Our goal is to fetch pages in order of impact. Since crawling is generally performed in batches or *cycles*, the problem then becomes that of selecting pages to fetch in the next cycle.

As stated in Section 1.2, we propose the following approach for estimating impact. We define a new scoring function $S'(p, q)$ which takes the content-independent features of page p and query q as input and outputs a probability distribution of $S(p, q)$, i.e., values that $S(p, q)$ can take along with their probabilities.

A *query sketch* consists of the set of pages relevant to a query and their associated score (for crawled pages) or score distribution (for uncrawled pages), as illustrated in Figure 3 (we depict distributions as intervals). From the set of query sketches for queries in a workload, it is possible to derive bounds or construct probability distributions for rank $R(p, q)$ and impact $I(p, q)$. For example, from Figure 3 it is evident that uncrawled page P_3 will receive a rank between 3 and 5, and hence in the worst case its impact for query q is $I(p, q) = V(5)$.

Given the above framework we state the crawl selection problem. We are given n crawled and m uncrawled pages, and the sketches of all queries in the workload $\mathcal{Q}$. The objective is to select the c pages of maximum total impact (in either the expected sense or the worst-case sense), where $c \ll m$.

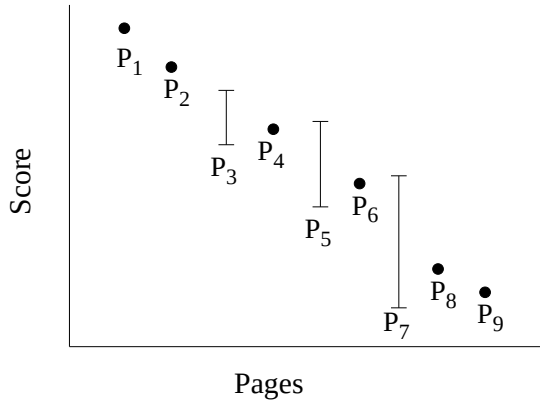


Figure 3: A query sketch.

Formally, let indicator variable $X_p \in \{0, 1\}$ denote the event of fetching uncrawled page p, i.e., $X_p = 1$ if p is fetched and 0 otherwise. Then, the crawl selection optimization problem for the expected case objective can be stated as follows (a similar formulation can be given for the worst case objective):

$$\text{maximize } \mathbb{E}\Big(\sum_p \big(X_p \cdot I(p, \mathcal{Q})\big)\Big)$$

where

$$\sum_p X_p = c$$

4. PROBLEM COMPLEXITY

In this section we analyze the complexity of the crawl selection optimization problem under the worst case and expected case objectives. Readers interested only in our proposed crawling policy may skip to Section 5.

4.1 Worst Case Impact Maximization

Under the objective of maximizing impact in the worst case, our problem is NP-hard to solve exactly and does not admit an *FPTAS* approximation scheme, for any form of score distributions $S'(p, q)$ (except point distributions) and any nonconstant monotonically nonincreasing visibility function $V(\cdot)$.

We illustrate the difficulty of this problem via an example. Suppose the query workload consists of only one query q, and say that the sketch of q has no crawled pages and exactly two uncrawled pages, P_1 and P_2, with overlapping score distributions (the shapes of the distributions do not matter for the worst case reasoning). The worst case impact of fetching only page P_1 is $f(q) \cdot V(2)$, because in the worst case, page P_1 is of lower score ($S(p, q)$) than P_2 and thus its rank $R(p, q) = 2$. Similarly, the worst case impact of fetching page P_2 is $f(q) \cdot V(2)$. However, if both pages P_1 and P_2 are fetched, then the total worst case impact is $f(q) \cdot (V(1) + V(2))$, because of the two pages one has to be at rank 1 and the other at rank 2. If $V(1) > V(2)$, then the total worst case impact of fetching the two pages is greater than the sum of their individual worst case impact values computed in isolation. Hence, the worst case impact ofTHM:1 fetching a page depends on which other pages are fetched at the same time. This property makes the impact maximization problem non-trivial to solve. In fact, the problem is NP-hard.

Theorem 1 *The problem of finding K pages that have the highest total impact in the worst case, is NP-hard for any nonconstant monotonically nonincreasing function $V(\cdot)$.*

Proof See Appendix A. ■

In addition to being NP-hard, it turns out that the problem does not admit a *fully polynomial-time approximation scheme* (FPTAS) [18], assuming $NP \nsubseteq \cap_{\epsilon>0} BPTIME(2^{n^\epsilon})$. Moreover, the problem does not have a *polynomial-time approximation scheme* (PTAS) if the visibility function goes to zero (i.e., $\exists\ r$ such that $V(r) = 0$). The intuition behind this result is similar to that of NP-hardness proof given in Appendix A, where we show that the number of edges in a subgraph, n_s, is related to the total worst-case impact of the corresponding set of pages. An approximation for the worst-case impact translates to an approximation for n_s, which is shown in [12] not to be possible, assuming $NP \nsubseteq \cap_{\epsilon>0} BPTIME(2^{n^\epsilon})$.

Theorem 2 *The problem of finding K pages that have the highest total impact in the worst case, does not admit FPTAS for any nonconstant monotonically nonincreasing function, assuming $NP \nsubseteq \cap_{\epsilon>0} BPTIME(2^{n^\epsilon})$. Moreover, the problem does not admit PTAS if there exists an integer constant R such that $V(R) = 0$.*

Proof See Appendix B. ■

4.2 Expected Case Impact Maximization

We give an optimal algorithm for the expected case impact maximization problem. The running time of the algorithm depends largely on the complexity of computing the rank distributions of random variables (in the vein of order statistics [3]), which can be expensive in practice as it involves solving convoluted integrals.

The key property we exploit in the algorithm is that expectation distributes over the sum of random variables. Hence, the total expected impact of fetching c pages is equal to the sum of their individual expected impact values.

Algorithm to Maximize Expected Impact

- *For each page p and query q, compute the probability distribution of $R(p, q)$, denoted by $\mathbb{P}(p, q, r)$, in the manner described below.*
- *Compute the expected impact of page p as:*

$$\mathbb{E}(I(p)) = \sum_{q \in \mathcal{Q}} \sum_{r=1}^{\infty} f(q) \cdot V(r) \cdot \mathbb{P}(p, q, r)$$

- *Select the c pages of highest expected impact.*

The rank distribution $\mathbb{P}(p, q, r)$ can be computed using the methods of order statistics [3], if the sketch of query q consists of independent identically distributed score distributions. In case that is not true, as is predominantly the case in search result ranking, there are more expensive monte-carlo methods that can be used.

This optimal algorithm requires computing a large number of rank distributions ($m \cdot |\mathcal{Q}|$ of them, where m is the number of uncrawled pages and $|\mathcal{Q}|$ is the number of queries in the workload). Performing this computation on Web-scale data, in the inner loop of a crawler, is not realistic.

5. IMPACT-DRIVEN CRAWLING POLICY

In Section 4 we showed that solving the crawl selection problem exactly (for the worst case or the expected case) is prohibitively expensive. Hence, in the remainder of this paper we focus on developing practical approximate methods. Due to the especially bad complexity of the worst case variant (it is hard even to approximate), we focus on approximate methods for expected impact.

Most of the complexity of the expected case variant is due to considering the score distributions of uncrawled pages. Hence we make the following simplification: function $S'(\cdot)$ outputs an expected score value instead of a score distribution.

We also consider a restricted visibility function $V(\cdot)$. Following [4] we let $V(\cdot)$ be a step function where $V(r) = 1$ for $r \leq K$ and $V(r) = 0$ otherwise, for some $K \geq 1$. This form models the steep drop in attention between the top results which are immediately visible on the user's screen and the subsequent results that come into view if the user scrolls or clicks.

Under the above simplifications the impact maximization problem can be stated as follows: given the query sketches[2] find c pages of maximal total impact, where impact is:

$$\begin{aligned} I(p, \mathcal{Q}) &= \sum_{q \in \mathcal{Q}} f(q) \cdot V(R(p,q)) \\ &= \sum_{q \in \mathcal{Q}} f(q) \cdot T(p,q) \end{aligned} \quad (1)$$

where

$$T(p,q) = \begin{cases} 1 & \text{if } p \text{ is in the top K results in the sketch of } q \\ 0 & \text{otherwise} \end{cases}$$

In other words, the impact of page p is equal to the sum of the frequencies of the queries for which page p is among the top K results. This number is easy to obtain from the query sketches. (Note that the query sketches need only contain the top K pages.)

To speed up the impact computation, we only build and use sketches for a small subset of queries, in particular those that (1) occur with non-negligible frequency, and (2) can potentially have their results improved by crawling new pages. In steady state, most frequently-occurring queries have already been supplied with plenty of high-quality relevant pages, and queries that do require special attention from the crawler typically constitute a small minority. We refer to such queries as *needy queries*.

In the remainder of this section we describe our crawl selection technique in detail, including our method of classifying queries as needy or non-needy.

[2]Since the output of $S'(\cdot)$ is now a scalar value, a query sketch consists of scalar score values only, rather than a mixture of scalar values and distributions.

5.1 Overview of Crawl Selection Technique

Figure 4 shows the process of selecting pages to fetch in the next crawl cycle.[3] Cylinders represent data; boxes represent computation. The key components are:

- *Queries + top K results*: A representative workload of search queries with associated frequencies (perhaps obtained from historical logs combined with forecasting methods), and the scores of the current top K search results.
- *Identify needy queries*: Given the query workload and top K result scores, classify queries as either *needy* or *non-needy* using the method described below in Section 5.2.
- *Uncrawled page metadata*: Known, content-independent information about each uncrawled page, e.g. URL, current in-link count or PageRank, known referring anchortext. (This metadata can be collected during previous crawl cycles.)
- *Match + assign scores*: For each needy query q, identify matching uncrawled pages p and compute the expected score $S'(p,q)$. (A page "matches" a query if it receives a nonzero (expected) score.) Matching is performed in sublinear time using an index built over the URLs and referring anchortext of the uncrawled pages; see Section 5.3 for details.
- *Create needy query sketches*: For each needy query create its query sketch, i.e. the top K (expected) scores, among all crawled and matching uncrawled pages.
- *Estimate impact*: Estimate the impact of fetching each uncrawled page from the needy query sketches using Equation 1. Then, supplement this query-based impact estimate with a query-independent estimate computed from query-independent features such as in-link count or PageRank. (This *hybrid* impact estimation procedure is necessary in practice due to imperfect query workload models.) Details of our hybrid estimation procedure are given in Section 5.4 below.
- *Select pages to fetch*: Select the c pages of highest estimated impact.

5.2 Identifying Needy Queries

Recall that a query is *needy* if impact can be had by biasing the crawler toward pages estimated to be relevant to it. If $\mathcal{C}$ is the set of pages fetched in a given crawl cycle, the portion of impact achieved for a given query q is:

$$\begin{aligned} I(\mathcal{C}, q) &= f(q) \cdot \big(\sum_{p \in \mathcal{C}} I(p,q)\big) \\ &= f(q) \cdot \big(\sum_{p \in \mathcal{C}} T(p,q)\big) \end{aligned}$$

If we are to select a subset of queries to use in biasing the crawler, a good strategy is to select the subset with maximal total I according to the above expression. In other words, if we define the *neediness score* of query q to be $neediness(q) =$

[3]In practice, rather than running this process from scratch for each new crawl cycle, one would of course maintain key data structures such as the query sketches incrementally. Incremental maintenance of top-K structures such as our query sketches can be done using known techniques [21].

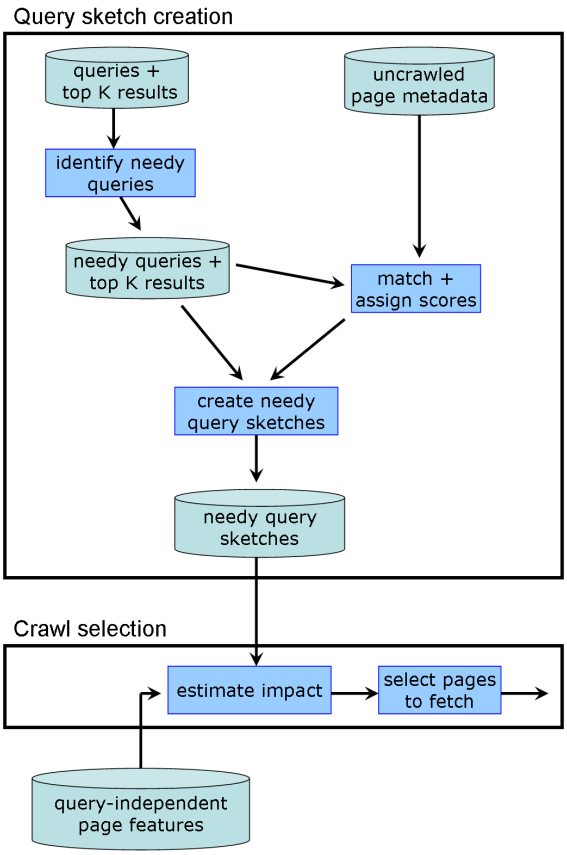


Figure 4: Impact-driven crawl selection steps.

$I(\mathcal{C}, q)$, we should select the queries with highest neediness scores.

The neediness score has two components: the query frequency $f(q)$, and a term that represents the *improvement* to the result set of query q, which depends on the set $\mathcal{C}$ of pages we fetch in the next cycle. The astute reader will observe that we are faced with a circularity: in order to identify needy queries according our definition we need to know which pages we will fetch next, yet our selection of pages to fetch is based on needy queries. To eliminate the circularity we require the ability to estimate the *improvability* (expected improvement) of a query based on some features of the query such as its current score distribution. Given data on query result improvement achieved in previous crawl cycles, a function from query features to improvability can be fit using regression.

There are many ways to learn such a function. One simple method that works well in our experience is to use the average score of the current top K results for a query as a feature, and use log-linear regression to fit a function from this feature to improvability (we validate this method in Section 6.3). The intuition is that queries with low-score results (e.g. "tail queries" on nascent or obscure topics like "Judy Inklesmurf") are more likely to be improvable than ones with high-score results (e.g. "head queries" such as "NCAA Football" whose result pages are highly entrenched and are unlikely to be displaced by newcomers), as discussed in Section 1. Of course, improvability does not only depend on the result scores, but also on the scores of uncrawled pages; however this information is not available while selecting needy queries, so we make due without it.

5.3 Matching Uncrawled Pages With Queries

Given a query q, we are to identify uncrawled pages p that "match" q, i.e. have a nonzero score $S(p, q)$. Since we are dealing with uncrawled pages, the only information available for matching is content-independent metadata such as URL strings and referring anchortext strings, and we of course cannot determine matches with full accuracy.

Instead, we label page p as a match for query q if the amount of textual overlap between the query string and p's URL and referring anchortext strings is above some threshold. In particular, we convert each of these strings into word-level n-grams for all $n \in [1, g]$ where g is a constant giving the maximum n-gram length, and declare a match if at least ρ fraction of the query n-grams match one of the page n-grams, for some $\rho \in [0, 1]$. Using a smaller value of ρ results in greater accuracy in the subsequent impact estimation step but also greater overhead, and vice-versa. (In our experiments, to make our results conservative we set $\rho = 1$, which favors efficiency over accuracy; see Section 6.) To identify matches efficiently, we maintain an index over the uncrawled page n-grams and perform lookups with each needy query n-gram.

5.4 Hybrid Impact Estimation

Selecting pages to fetch based solely on matching URL and anchortext strings with needy queries, has some fundamental limitations. Of course one problem is that some pages have little or no referring anchortext and lack a meaningful URL, yet still turn out to be impactful for other reasons such as high PageRank, and therefore worth fetching. Perhaps a more significant concern is the assumption that the query workload model covers all important future queries, which is of course dubious. For these reasons we propose a hybrid impact estimation scheme, which combines a *query-based* estimate that takes into account query neediness and relevance considerations, with a *query-independent* estimate that is not vulnerable to the problems just mentioned.

We learn a function from query-independent page features such as PageRank to impact, using a training set of previously-crawled pages. In our experiments we used regression to fit a quadratic function. (As expected the fit is not very good, due to the lack of strong correlation between query-independent importance and impact, as illustrated in Figure 2. The inability to translate importance to impact reliably is precisely the motivation behind our query-based approach. In Section 6.2 we show that by combining this query-independent impact prediction with a query-based prediction we get a good overall results.)

To combine this query-independent impact estimate with our query-based estimate we simply take a weighted average. In our experiments we set the weight of the query-based estimate to 0.9 and the weight of the query-independent estimate to 0.1, which yielded good results. (The results were not especially sensitive to the exact weights used.)

6. EVALUATION

In this section we evaluate the effectiveness of our impact-based crawl ordering technique. Throughout our experiments we set parameter K (the bottom-most rank position

in a query result list that is considered likely to be viewed) to $K = 10$.

6.1 Data

Our experiments simulate crawl ordering policies over real web data, guided by real search queries.

Query workload. We used the complete query log of a major search engine for a five-day period (9/16/2006 to 9/20/2006).

Web pages (*Dataset 1*). We took a random sample of approximately 110,000 pages from the crawl of a major search engine, and for the purpose of our experiments treated them as "uncrawled." The remaining pages (many billions) were treated as "crawled." [4] We refer to this data set as *Dataset 1*.

Web pages (*Dataset 2*). Search engine crawlers (including a hypothetical crawler that uses our technique) are generally biased toward acquiring pages that have a high query-independent importance score (e.g., high PageRank). So, one would expect uncrawled pages to have lower average importance score than crawled ones, which is not the case in Dataset 1. To simulate this situation we took Dataset 1 and promoted the top 20% pages of highest importance from the "uncrawled" set to the "crawled" set. We refer to this new data set as *Dataset 2*.

Scoring function. We used the query result scoring function employed by a major search engine for $S(\cdot)$. We then learned the content-independent $S'(\cdot)$ function as a regression tree, using a publicly available machine learning tool called WEKA [19].

Query-page matching. Recall from Section 5.3 that parameter $\rho \in [0, 1]$ controls how conservative the query-page matching process is: small ρ leads to many matches, and large ρ leads to few matches. This parameter governs a tradeoff between overhead (many query-page matches means slower sketch construction) and accuracy (few query-page matches means sketches may be incomplete). We desire to measure the worst-case accuracy, i.e., the accuracy a system would have if tuned for least overhead. Hence in all of our experiments we set $\rho = 1$.

We used n-grams of length up to $g = 3$.

6.2 Effectiveness of Page Ordering Policies

Our first experiment compares the end-to-end performance of various policies for ordering pages for fetching, where the goal is to fetch the pages of highest (actual) impact first. For now we assume that sketches for all queries are available (we study the case in which only a select subset of query sketches are used in Section 6.4). The policies we compare are:

- **random:** orders pages according to a random permutation.
- **query-independent:** orders pages in descending order of query-independent importance score.
- **query-driven:** orders pages in descending order of impact, as estimated from query sketches.
- **hybrid:** orders pages according to an impact estimate that combines a strictly query-independent estimate with a strictly query-based estimate, as discussed in Section 5.4.

[4] In a real crawling scenario, the uncrawled pages would outnumber the crawled ones. Our simulation chooses an order for a small subset of the uncrawled pages—in particular the ones that were acquired by the real search engine's crawler.

Figure 5 shows the performance of these policies on our two Web page data sets. In each graph the x-axis plots the fraction of uncrawled pages fetched, and the y-axis plots the cumulative impact accrued by a given policy.

The query-driven policy performs very well at first, because the query-page matching and scoring yields excellent suggestions of high-impact pages to fetch first. However, as the query relevance clues "dry up," so to speak, the purely query-driven policy starts to perform less well than the query-independent policy. The hybrid policy, which incorporates both types of information, does the best. The superiority of the hybrid policy is especially pronounced on Dataset 2, which we consider to be the more realistic of the two, as discussed in Section 6.1.

Under the hybrid crawl ordering policy, pages with a relatively low importance score but that are expected to have high impact can be said to be *promoted* to an earlier position, compared with query-independent ordering. An example is `www.texhoma.us/history.htm` (shown in Figure 6(a)), which is about Texhoma, a town on the Oklahoma-Texas state line in the United States that has generated some publicity due to its intriguing geographical location. The major search engines Google, MSN and Yahoo! all rank this page in the top 10 results for the query "texhoma," which occurred with moderate frequency in our query log. Yet this page does not have a particularly high link-based importance score. The hybrid policy places it much earlier in the crawl order than the query-independent policy, due to textual matching between the query string and the page's URL and referring anchortext.

Texahoma is an example of a "tail topic," i.e. one that garners some interest among the user population, but not enough to generate a large number of in-links to pages on the topic. Another tail topic example is "toyota forums," for which the page `www.topix.net/forum/autos/toyota` (shown in Figure 6(b)) is one of the top 10 results. The query "toyota forums" is somewhat frequent and hence this page has more impact than most pages, yet this page has a low link-based importance score. Again, the hybrid policy treats this page preferentially despite its weak importance score, due to matches of its URL and referring anchortext with the query. Our data set contains many examples of this form, which collectively account for the difference in performance between the hybrid and query-independent policies.

6.3 Ability to Learn Query Improvability

Next we study the ability to prune the state required by the query-driven and hybrid policies. In particular we wish to limit the number of queries for which we must compute, maintain and read sketches, without sacrificing effectiveness. Recall from Section 5.2 that our method of selecting queries is to rank them according to *neediness*, which is the product of query frequency with *improvability*. Improvability is the extent to which we expect to be able to alter the top K query results by crawling new pages. The key issue is whether we

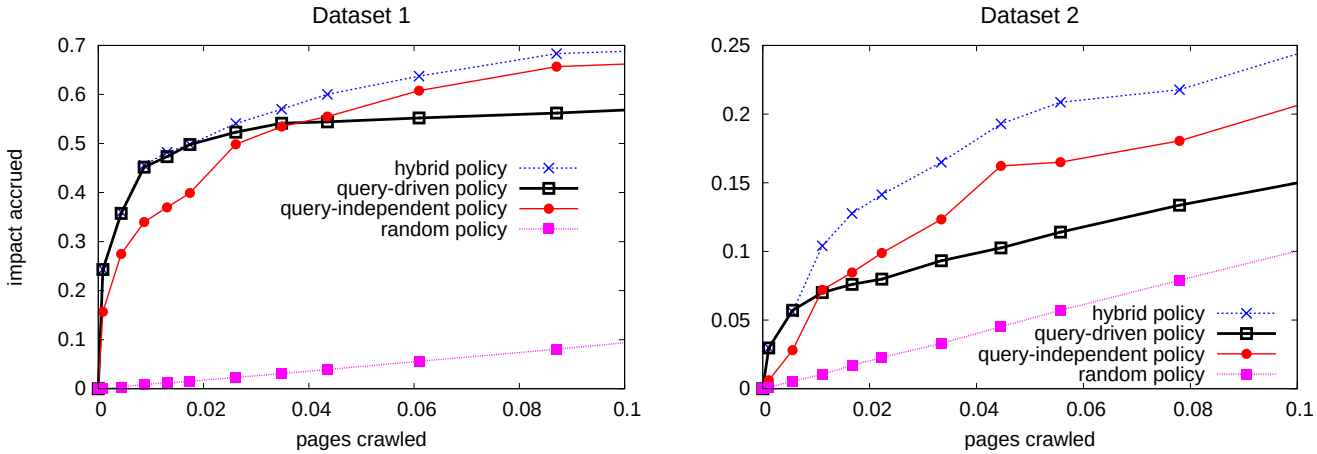
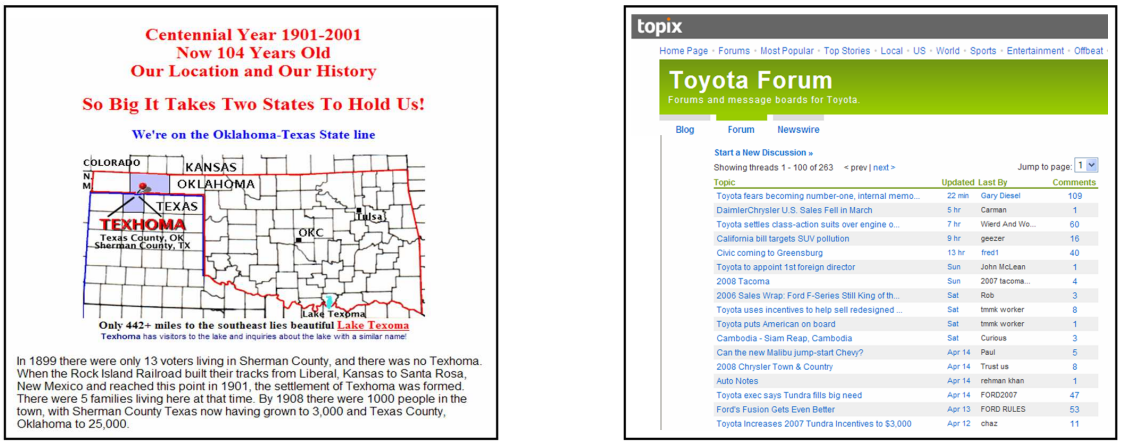


Figure 5: Effectiveness of different crawl ordering policies.



(a) (b)

Figure 6: Examples of pages crawled earlier by our impact-driven approach, compared with a query-independent approach.

can learn query improvability as a function of features of the query result score distribution prior to crawling.

To test our ability to learn query improvability, we randomly divided our query workload in half, into a training set and a test set. We then fed the training set to our hybrid impact-based crawl ordering policy, which selected c uncrawled pages to fetch, and we measured the resulting improvement to each query (in terms of number of new top K results fetched). Next we used the training data to fit a function from average top-K score (prior to crawling), to improvement. (To fit this function, we first grouped queries into buckets according to average top K score, and computed the average improvement for each bucket; then we used log-linear regression to fit a curve.) Lastly we compared the function learned on the training data to the test data, as shown in Figure 7.

There are two graphs: one for each of our Web page data sets. Each graph plots average top K score on the x-axis and improvability on the y-axis. The '$\times$' symbols show data points from the test set (one point per query bucket), and the smooth curve shows the function learned over the training set. As we can see, the learned function predicts the behavior of the test set with fairly high accuracy. (We used $c = 1000$ for these graphs; similar results were observed for other values of c.)

This result demonstrates that the relationship between average top K score and improvability remains stable across two disjoint query samples, and hence it is reasonable to model query improvability as a function of the average top K score feature. Of course, since we are using average improvability values, the prediction for a given query may be off due to variance around the mean. The main source of variance is the variation in relevance of uncrawled pages—given two queries whose crawled relevant pages have roughly the same scores, it may still be the case that the scores of the uncrawled relevant pages differ between the two queries. This variation cannot be modeled without incurring the unacceptable additional overhead of matching all queries (not

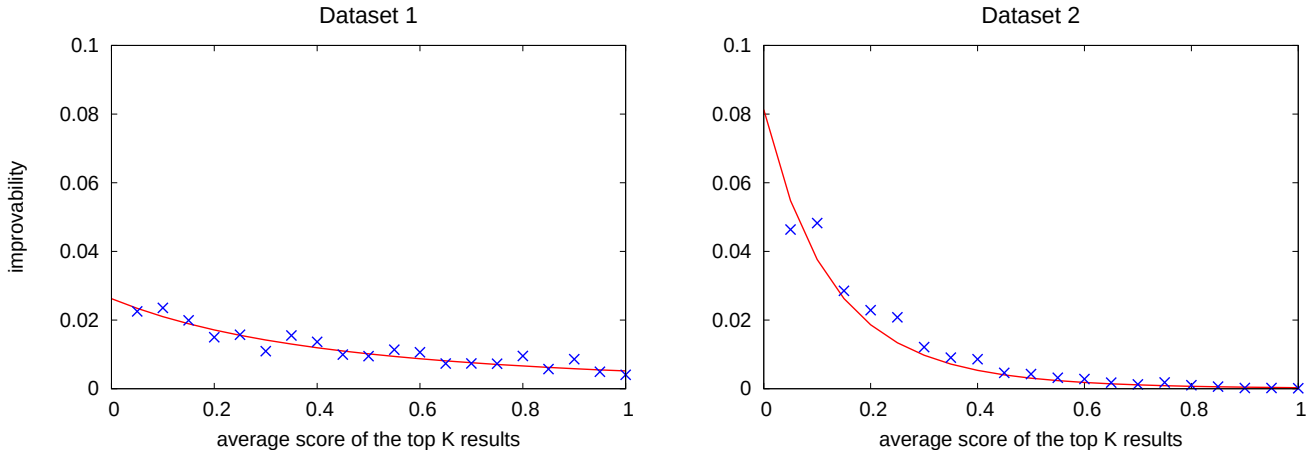


Figure 7: Learning *query improvability* as a function of the average score of the top K results.

just needy ones) with uncrawled pages, so we do not attempt to do so. Fortunately it turns out that nonetheless the improvability predictor we learn just from features of crawled pages serves us well, as we demonstrate next.

6.4 Effectiveness with Limited Query Sketches

Now we are ready to evaluate the effectiveness of our impact-based crawl ordering technique when we use only selected query sketches rather than all of them. For this experiment we learned a query improvability function from the training query set (as described in Section 6.3), and then used it on the test query set, along with query frequency data, to select the most *needy* queries. We then simulated our hybrid crawl ordering policy, and only allowed it to access sketches for the selected queries.

Figure 8 shows the result, over our two Web page data sets as usual. The x-axis of each graph plots the fraction of query sketches retained, and the y-axis plots total impact yielded. The different curves correspond to different crawl budgets c. Most of the impact can be had even if we only retain a small fraction (roughly 0.0007) of query sketches.

Figure 9 plots performance as a function of the crawl budget c for (a) the hybrid policy using all sketches and (b) the hybrid policy with 0.0007 fraction of sketches (for the queries estimated to be the most needy). (The performance curves differ from the ones plotted in Figure 5 because the evaluation was done on different query sets: Figure 5 is on the full query set and Figure 9 is on the test set only.) From Figure 9 we can see that the hybrid policy with only a small fraction of sketches performs almost as well as when using all sketches.

Our method of selecting "needy" queries succeeds in significantly shrinking the space and time requirements of a query-driven crawler. Only a minor reduction in effectiveness is felt.

7. SUMMARY

The job of a crawler is to narrow the gap between the pages the search engine currently returns in response to user queries, and the ones it could return if the appropriate content was crawled. We approached the crawling problem from this standpoint, and derived a new query-centric crawl ordering technique. The key components are: (1) identifying queries that can potentially have their results improved via crawling, and (2) selecting pages to fetch given these queries, the search engine's scoring function, and features of a page that are available prior to fetching it. We demonstrated that our approach achieves substantially greater impact on search results than the conventional query-independent approach. Our approach is especially impactful for "tail queries," which in aggregate represent a substantial fraction of all queries, yet are not necessarily well served by conventional query-independent techniques.

8. REFERENCES

[1] S. Abiteboul, M. Preda, and G. Cobena. Adaptive On-line Page Importance Computation. In *Proc. WWW*, 2003.

[2] S. Brin and L. Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. In *Proc. WWW*, 1998.

[3] G. Casella and R. L. Berger. *Statistical Inference*. Duxbury, 2001.

[4] S. Chakrabarti, A. M. Frieze, and J. Vera. The influence of search engines on preferential attachment. In *Proc. Symposium on Discrete Algorithms*, 2005.

[5] S. Chakrabarti, M. van den Berg, and B. Dom. Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery. In *Proc. WWW*, 1999.

[6] J. Cho and H. Garcia-Molina. Synchronizing a Database to Improve Freshness. In *Proc. ACM SIGMOD*, 2000.

[7] J. Cho, H. García-Molina, and L. Page. Efficient Crawling Through URL Ordering. *Computer Networks and ISDN Systems*, 30(1–7):161–172, 1998.

[8] J. Cho and S. Roy. Impact of Search Engines on Page Popularity. In *Proc. WWW*, 2004.

[9] A. Dasgupta, A. Ghosh, R. Kumar, C. Olston, S. Pandey, and A. Tomkins. The Discoverability of the Web. In *Proc. WWW*, 2007.

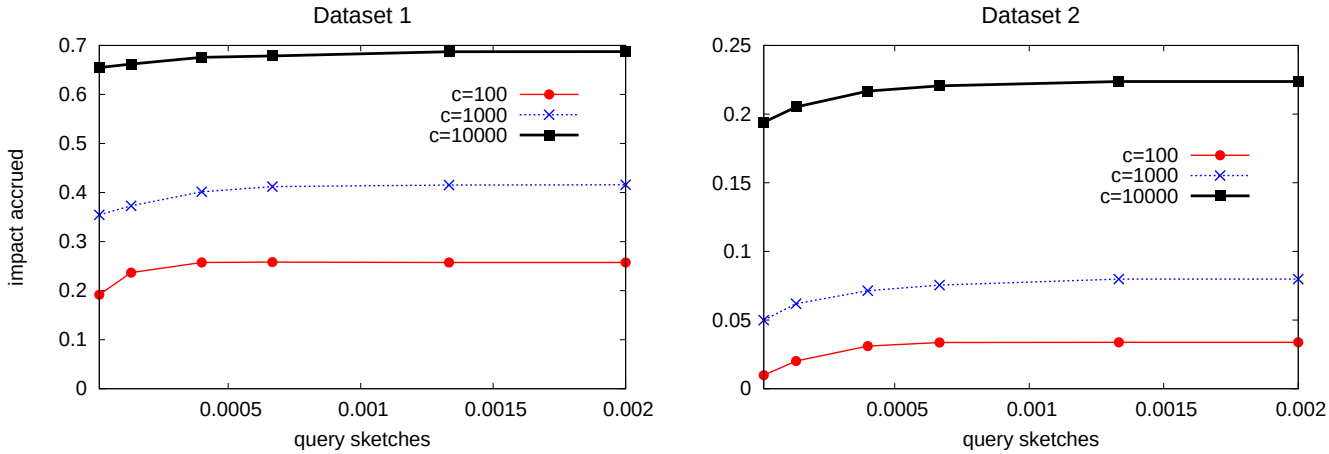


Figure 8: Performance of the hybrid policy on the test query set while varying the number of query sketches. The different curves are for different values of the crawl budget c.

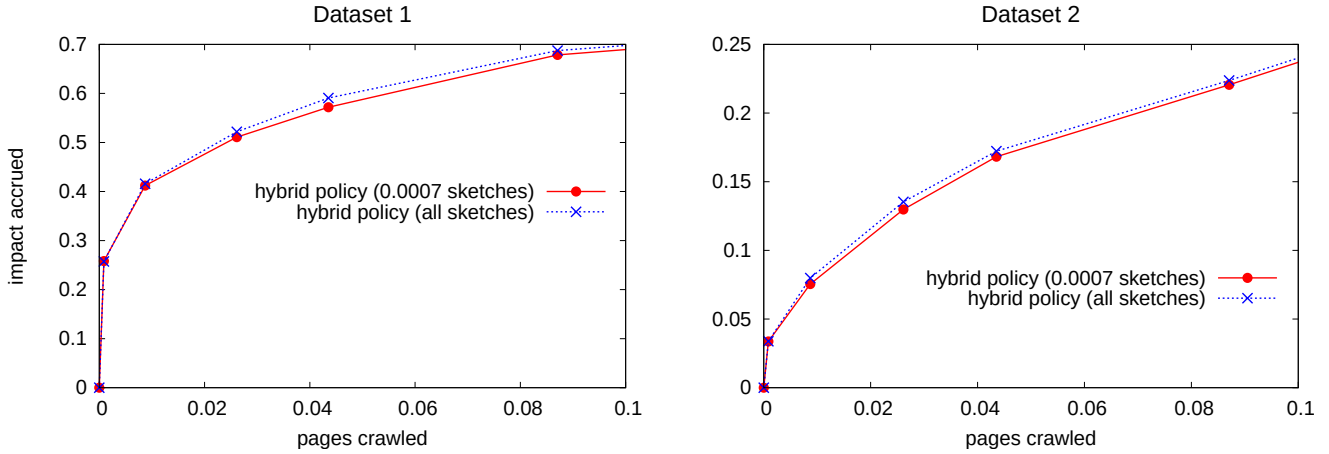


Figure 9: Effectiveness of hybrid crawl policy when only using a small fraction of query sketches (for the queries estimated to be the "neediest" based on the training query set), evaluated over the test query set.

[10] J. Edwards, K. S. McCurley, and J. A. Tomlin. An Adaptive Model for Optimizing Performance of an Incremental Web Crawler. In *Proc. WWW*, 2001.

[11] N. Eiron, K. S. McCurley, and J. A. Tomlin. Ranking the Web Frontier. In *Proc. WWW*, 2004.

[12] S. Khot. Ruling Out PTAS for Graph Min-Bisection, Densest Subgraph and Bipartite Clique. In *Proc. IEEE Symposium on Foundations of Computer Science*, 2004.

[13] R. Lempel and S. Moran. Predictive Caching and Prefetching of Query Results in Search Engines. In *Proc. WWW*, 2003.

[14] M. Najork and J. L. Wiener. Breadth-First Search Crawling Yields High-Quality Pages. In *Proc. WWW*, 2001.

[15] S. Pandey and C. Olston. User-centric Web Crawling. In *Proc. WWW*, 2005.

[16] M. Richardson, A. Prakash, and E. Brill. Beyond PageRank: Machine Learning for Static Ranking. In *Proc. WWW*, 2006.

[17] Search's Long Tail. `http://blog.searchenginewatch.com/blog/050314-164653`.

[18] V. Vazirani. *Approximation Algorithms*. Springer-Verlag, 2001.

[19] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[20] J. Wolf, M. Squillante, P. Yu, J. Sethuraman, and L. Ozsen. Optimal Crawling Strategies for Web Search Engines. In *Proc. WWW*, 2002.

[21] K. Yi, H. Yu, J. Yang, G. Xia, and Y. Chen. Efficient Maintenance of Materialized Top-k Views. In *Proc. International Conference on Data Engineering*, 2003.

APPENDIX

A. PROOF OF THEOREM 1

Our proof consists of a reduction from the densest K-vertex subgraph problem, which is known to be NP-hard [18]. (The densest K-vertex subgraph problem is the following: given graph G and a parameter K, find a subgraph containing K vertices that has the maximum number of edges.)

Let G denote the input graph instance of the densest K-vertex subgraph problem. Let U and E denote the set of vertices and edges of G respectively. Let r denote an integer constant such that $V(r) > V(r+1)$. For each $u_i \in U$ we create a page, denoted by $P_{i,i}$, in our impact maximization problem instance. Then we create three *kinds* of queries (let d denote the largest degree in G):

- **Kind A**: For each edge $e_{i,j} = (u_i, u_j)$ we create a query denoted by $A_{i,j}$. In the sketch of query $A_{i,j}$ we put pages $P_{i,i}$, $P_{j,j}$ and $r-1$ dummy pages denoted by $P^1_{i,j}, P^2_{i,j} \ldots P^{r-1}_{i,j}$, and then we assign the same score distribution to each of them.
- **Kind B**: For each $u_i \in U$ we create two queries $B^1_{i,i}$ and $B^2_{i,i}$ and put only page $P_{i,i}$ in their sketches.
- **Kind C**: For each $u_i \in U$ of degree d_i, we create another $d - d_i$ queries denoted by $C^1_{i,i}, C^2_{i,i} \ldots C^{d-d_i}_{i,i}$, where d is the largest degree in G. In the sketch of query $C^k_{i,i}$ we put page $P_{i,i}$ and r dummy pages denoted by $P^{k,1}_{i,i}, P^{k,2}_{i,i} \ldots P^{k,r}_{i,i}$, and we assign the same score distribution to each of them.

The frequency count is set to 1 for each query. Let $\mathcal{P}^*$ denote the set of K pages that maximize the worst case impact of the above instance. First, we claim that a dummy page can never be present in $\mathcal{P}^*$. This is true because a dummy page is present in one and only one query sketch and so the highest impact it can have is $V(1)$. On the other hand, each non-dummy page $P_{i,i}$ contributes at least $2 \cdot V(1)$ due to queries $B^1_{i,i}$ and $B^2_{i,i}$ since it is the only page present in their sketches.

Next we show that the vertices that corresponds to the pages in $\mathcal{P}^*$ form the densest K-vertex subgraph. We prove it by showing that in the aforementioned impact maximization problem instance, the worst case impact of fetching any set of K non-dummy pages, say $\mathcal{C}$, is directly proportional to the density of the corresponding K-vertex subgraph in G, say G_s. Next we compute the worst case impact of set $\mathcal{C}$.

As mentioned before, for each page $P_{i,i}$ in the set, we get the worst case impact of $\big(2 \cdot V(1)\big)$ due to queries $B^1_{i,i}$ and $B^2_{i,i}$. Also, page $P_{i,i}$ has $\big(d \cdot V(r+1)\big)$ worst case impact due to d_i number of $A_{i,*}$ and $d - d_i$ number of $C^*_{i,i}$ queries because in the worst case the page is ranked last (i.e., $r+1$) in all these sketches. However, note that for edge $e_{i,j} = (u_i, u_j)$ in G_s both pages $P_{i,i}$ and $P_{j,j}$ are present in $\mathcal{C}$, and so in the worst case one of them is ranked $r+1$ but the other one has to be ranked r for query $A_{i,j}$. Hence, due to each query sketch $A_{i,j}$ that corresponds to edge $e_{i,j} = (u_i, u_j)$ in G_s we get an additional worst case impact of $V(r) - V(r+1)$. Hence, the total worst case impact is equal to $(2 \cdot K \cdot V(1)) + (d \cdot K \cdot V(r+1)) + (n_s \cdot (V(r) - V(r+1)))$ where n_s denotes the number of edges in G_s. Hence, it proves that the worst case impact of set $\mathcal{C}$ is proportional to the density of subgraph G_s.

B. PROOF OF THEOREM 2

We show that an FPTAS for our problem would allow us to derive a PTAS for the densest K-vertex subgraph problem, which is not possible if $NP \nsubseteq \cap_{\epsilon>0} BPTIME(2^{n^\epsilon})$[12].

Let G denote the input graph instance of the densest K-vertex subgraph problem. Let U and E denote the set of vertices and edges of G respectively. Let r denote an integer constant such that $V(r) > V(r+1)$. We create an instance of our impact maximization problem as done in our proof of Theorem 1 (Appendix A). Let $\mathcal{P}^\epsilon$ denote the set of K pages output for this problem instance by an FPTAS scheme for a given ϵ. By definition, it means that the worst case impact of $\mathcal{P}^\epsilon$, denoted by I^ϵ, is within $1-\epsilon$ factor of the optimal worst case impact I^* achieved by, say, $\mathcal{P}^*$.

First, we claim that a dummy page in $\mathcal{P}^\epsilon$ can be replaced by a non-dummy page without decreasing the worst case impact of $\mathcal{P}^\epsilon$. This claim is true because, as shown in Appendix A, a dummy page cannot have more impact than $V(1)$, while the worst case impact of each non-dummy page is at least $\big(2 \cdot V(1)\big) + \big(d \cdot V(r+1)\big)$. Hence, we regard $\mathcal{P}^\epsilon$ to consist of non-dummy pages henceforth.

Suppose G_s denote the subgraph of G corresponding to the K pages in $\mathcal{P}^\epsilon$. Then, as shown in Appendix A, the total worst case impact of set $\mathcal{P}^\epsilon$, I^ϵ, is $(2 \cdot K \cdot V(1)) + (d \cdot K \cdot V(r+1)) + (n_s \cdot (V(r) - V(r+1)))$ where n_s denotes the number of edges in G_s. Similarly,

$$\begin{aligned} I^* &= \big(2 \cdot K \cdot V(1)\big) + \big(d \cdot K \cdot V(r+1)\big) \\ &\quad + \big(n^* \cdot (V(r) - V(r+1))\big) \end{aligned}$$

where n^* denote the number of edges in the densest K-vertex subgraph of G.

Since $I^\epsilon \geq (1-\epsilon) \cdot I^*$, we can derive the following inequality:

$$n^\epsilon \geq n^* \cdot \left(1 - \epsilon - \epsilon \cdot K \cdot \frac{\big(2 \cdot V(1)\big) + \big(d \cdot V(r+1)\big)}{n^* \cdot \big(V(r) - V(r+1)\big)}\right)$$

Without loss of generality we assume that n^* is at least $K/2$ (it is easy to see that deciding and solving the subgraph problem is poly-time for graphs where $n^* < \frac{K}{2}$). Hence,

$$\begin{aligned} n^\epsilon &\geq n^* \cdot \left(1 - \epsilon - 2 \cdot \epsilon \cdot \frac{\big(2 \cdot V(1)\big) + \big(d \cdot V(r+1)\big)}{\big(V(r) - V(r+1)\big)}\right) \\ &= n^* \cdot \big(1 - \epsilon \cdot O(d)\big) \end{aligned}$$

Hence, subgraph G_s is an ϵ' optimal solution for the densest K-vertex subgraph problem if we set $\epsilon = \frac{\epsilon'}{O(d)}$. Since, by definition, FPTAS runs in time polynomial in $1/\epsilon$, the ϵ' optimal solution G_s that we just derived for the subgraph problem is poly-time in its input. Hence, we have a PTAS for the densest K-vertex subgraph problem which is a contradiction [12]. Hence, our problem does not admit FPTAS.

To see why a PTAS cannot exist when $V(R) = 0$ for some integer R, we set $r = R-1$ in the above inequality. Without loss of generality we assume R to be the smallest integer where $V(R) = 0$. Hence, we get $n^\epsilon \geq n^* \cdot (1 - \epsilon \cdot O(1))$. This result implies that a PTAS for our problem allows us to construct a PTAS for the subgraph problem which is a contradiction.